DEPARTMENT OF
STATISTICS

On Kernel and Feature Learning in Neural Networks

A thesis submitted for the degree of
Doctor of Philosophy

Michaelmas 2022

Bobby He
St Peter's College

Department of Statistics
University of Oxford

Abstract

Inspired by the theory of wide neural networks (NNs), kernel learning and feature learning have recently emerged as two paradigms through which we can understand the complex behaviours of large-scale deep learning systems in practice. In the literature, they are often portrayed as two opposing ends of a dichotomy, both with their own strengths and weaknesses: one, kernel learning, draws connections to well-studied machine learning techniques like kernel methods and Gaussian Processes, whereas the other, feature learning, promises to capture more of the rich, but yet unexplained, properties that are unique to NNs.

In this thesis, we present three works studying properties of NNs that combine insights from both perspectives, highlighting not only their differences but also shared similarities. We start by reviewing relevant literature on the theory of deep learning, with a focus on the study of wide NNs. This provides context for a discussion of kernel and feature learning, and against this backdrop, we proceed to describe our contributions. First, we examine the relationship between ensembles of wide NNs and Bayesian inference using connections from kernel learning to Gaussian Processes, and propose a modification that accounts for missing variance at initialisation in NN functions, resulting in a Bayesian interpretation to our trained deep ensembles. Next, we combine kernel and feature learning to demonstrate the suitability of the *feature kernel*, i.e. the kernel induced by inner products over final layer NN features, as a target for knowledge distillation, where one seeks to use a powerful teacher model to improve the performance of a weaker student model. Finally, we explore the gap between collapsed and whitened features in self-supervised learning, highlighting the decay rate of eigenvalues in the feature kernel as a key quantity that bridges between this gap and impacts downstream generalisation performance, especially in settings with scarce labelled data. We conclude with a discussion, including limitations and future outlook, of our contributions.

Acknowledgements

I'd like to start by thanking my supervisor Yee Whye, and co-supervisors, Arnaud and George, for their kindness and support during my PhD. To Yee Whye, I owe a great deal for starting me on the long and winding road that lead to much of this thesis, and for his guidance and patience when wrong paths were inevitably taken. I am grateful to Yee Whye also for the encouragement I received throughout my PhD to follow my interests, and for exemplifying qualities like openness, resilience and compassion, that I strive for, both within and outside of my research.

At the Department of Statistics in Oxford, I'd like to thank: Adam, Alan, Bryn, Chris, Déborah, Edwin, Emile, Emilien, Faaiz, Fran, James, Jean-Francois, Jessie, Jin, Lorenzo, Michael, Natalia, Qinyi, Robert, Sheh and Tyler for their friendship over these past four years. This list wouldn't be complete without the Japanese van or the countless raspberry buns that came and went along the way. I'd also like to thank the department's staff, especially Beverley, Joanna, Mark, Stuart and Susan, for their help on any issue of mine, no matter how large nor small.

I also owe a lot of gratitude to Mete for his unwavering support whilst allowing me to follow my instincts, and for creating an ideal environment for me that resulted in my time at SRUK being fruitful far beyond my expectations. I want to take the opportunity here to thank Ching-Ling, as it was around now the music started to play again.

It was a real privilege too to intern at DeepMind in London in the final year of my PhD, and for that I am extremely grateful to my host James. I'd like to thank James too for his willingness to share his vast depths of knowledge and experience, and for his inspiring dedication and attention to detail. Thanks also to Alex, Alexis, Ben, Guodong, Lisa, Sam, Tomas, and all of the staff that made DeepMind a special place to be.

In addition, I'm truly indebted to the incredible researchers who I've been lucky to work with and learn from so far. Thanks to Andy, Anne, Andrei, Avishkar, Balaji, Eugenio, Guy, Jonathan, Judith, Karolina, Nalini, Nenad, Peter, Soufiane, and Ulrich.

Finally, and most importantly, I turn to my family, to whom I dedicate this work. For all your constant love and support, I cannot begin to describe how much I appreciate you. To my sister Jenny, for her tenacity; to my mum Yan, for her positivity; and to my dad Li, for his passion and enthusiasm. Thank you. Without you, this thesis would not have been possible.

Anyway, here's wonderwall...

Contents

Chapter 1

Introduction

Deep learning (DL) systems, and deep neural networks (DNNs) in particular, have achieved unprecedented success in recent years, driving advances in Machine Learning (ML) applications such as computer vision (Krizhevsky et al., 2012), natural language processing (Devlin et al., 2018), and achieving superhuman performance in complex games (Silver et al., 2017), to name a few. At its core, a DNN is simply a collection of small modular building blocks, with trainable parameters, that are composed in some topology to form an expressive function of input data. From this basic recipe, recent progress in DL has been facilitated largely through practical developments, like the availability of large-scale real world training datasets (Deng et al., 2009; Panayotov et al., 2015; Raffel et al., 2020), and computational improvements in both the hardware (Owens et al., 2008; Lindholm et al., 2008; Jouppi et al., 2017) and software (Rumelhart et al., 1986; Baydin et al., 2018; Paszke et al., 2019; Bradbury et al., 2018) that we use to process data. As such, the success of DNNs has occurred to a large extent through empirical validation, as opposed to theoretical verification.

Despite empirical progress thus far, advancing the theory of DL remains important in order to provide a concrete framework for understanding the mechanisms behind how DNNs learn from data, and make predictions or take actions based on that learning. In turn, an improved understanding is crucial for further developing effective DL algorithms and realising the full potential that DNNs have to offer in practice. Some potential opportunities that an improved understanding of DNNs would enable include improving DNNs in areas where they traditionally underperform other ML approaches (Shwartz-Ziv and Armon, 2022; Grinsztajn et al., 2022), or simplifying the complex design choices that a practitioner needs to make when training a large-scale DL system (Shallue et al., 2019; Zhang et al., 2019b; Yang et al., 2022a).

However, modern DNNs are afflicted by high-dimensional, non-convex and non-linear loss surfaces in their parameter space (Dauphin et al., 2014; Li et al., 2018). This means that fundamental results about the mechanisms of DNN training, i.e. the process by which a DNN makes use of available data, are difficult to establish. As such, it is common for theoreticians to resort to simplified settings such as linear (Saxe et al., 2013), single hidden-layer (Sirignano and Spiliopoulos, 2018; Mei et al., 2018; Allen-Zhu and Li, 2020) and/or wide DNNs (Neal, 1996; Matthews et al., 2018; Lee et al., 2018a; Jacot et al., 2018;

Yang, 2019a) to be able to maintain analytic tractability.

In particular, studying the *wide* limit of DNNs has been fruitful within the research community in recent years, not least because it has uncovered universal links that exist between DNNs and classical ML methods like kernels and Gaussian processes (GPs) (Neal, 1996; Matthews et al., 2018; Lee et al., 2018a; Jacot et al., 2018; Yang, 2019a), whose theoretical analyses are more mathematically convenient and hence better understood. As a result, one key advantage of this so-called *kernel correspondence* of wide DNNs is the analytic tractability that it allows (Lee et al., 2019), leading to groundbreaking results in a range of areas pertaining to DNNs. These include the first proof of global convergence and generalisation in DNNs (Allen-Zhu et al., 2019; Cao and Gu, 2019; Du et al., 2019), as well as characterising the frequency bias in learnt functions (Ronen et al., 2019; Bietti and Mairal, 2019; Cao et al., 2019; Tancik et al., 2020) and designing performant DNN architectures, including more efficient ones without skip connections nor normalisation layers (Xiao et al., 2018; Martens et al., 2021; Zhang et al., 2022; Zaidi et al., 2022), for example.

Following an initial excitement around kernel behaviour in wide DNNs, a consensus in the field has emerged that the kernel correspondence is not sufficient to describe the *entire* behaviour of DNN training (Chizat et al., 2019; Aitchison, 2020; Fort et al., 2020; Ghorbani et al., 2020; Allen-Zhu and Li, 2020; Yang and Hu, 2020; Baratin et al., 2021). In particular, a core ingredient of DNNs that is incompatible with the infinite-width kernel limit is the notion of *feature learning*, where the DNN flexibly adapts its internal "representations" of data as it comes across new data during training. Though not fully understood, it is widely believed that feature learning is crucial to explaining many of the complicated phenomena that DNNs are capable of in practice, such as: the benefits of pretraining (Devlin et al., 2018), meta-learning (Finn et al., 2017), knowledge distillation (Hinton et al., 2015) and ensembling (Hansen and Salamon, 1990; Perrone and Cooper, 1992; Krogh and Vedelsby, 1994). While recent works have shown the existence of theoretical regimes that exhibit feature learning in general wide DNNs architectures (Yang and Hu, 2020; Bordelon and Pehlevan, 2022), these regimes lose the analytic tractability that the kernel limit enjoys. As such, practical insights that one can derive from theories of feature learning have so far been limited compared to the kernel regime, and reversing this trend is an active area of research (Yang et al., 2022a).

1.1 Thesis Outline

The similarities and contrast between kernel and feature learning has provided a new lens through which to view DNNs, and is central to the contents of this thesis. This is an integrated thesis, where Chapters 3 to 5 are separated into self-contained papers that each explore one of, or both, kernel and feature learning in DNNs. In Chapter 2 we first introduce background material on DNNs and kernel methods, before reviewing recent literature that has connected these seemingly disparate ML algorithms in the wide DNN limit. We then proceed to introduce feature learning as an alternative to kernel learning, and discuss how various DL phenomena fit into this paradigm between kernel and feature learning, setting the scene for the ensuing chapters.

In Chapter 3 (He et al., 2020a), we use the kernel regime to characterise the relationship between deep

ensembles (Lakshminarayanan et al., 2017) and Bayesian inference. This is a question of significance for the Bayesian DL community because ensembling forms a very strong baseline in uncertainty quantification (UQ) in DNNs, but was not originally proposed as a principled Bayesian approach to UQ. We show that, in wide DNNs, deep ensembles are not Bayesian owing to missing variance at initialisation, and provide a modification to correct for this, so that our trained ensembles approximate a so-called *Neural Tangent Kernel Gaussian Process* (NTKGP) posterior in the infinite-width limit. On standard regression and classification UQ tasks, we demonstrate that our Bayesian deep ensembles outperform standard deep ensembles.

In Chapter 4 (He and Ozay, 2021), we combine feature learning with kernel learning by treating the *feature kernel*, defined in Section 2.3, as a key object to target for knowledge distillation (KD) (Hinton et al., 2015), which is a method that uses a teacher DNN in order to improve the performance of a student DNN. By focusing on the feature kernel, our proposed *Feature Kernel Distillation* (FKD) has the benefit relative to standard KD that it does not need the prediction spaces and training datasets of the teacher and student to match. We extend the theory of Allen-Zhu and Li (2020) to show that FKD can improve student NN test accuracy, whereas standard training without KD fails to generalise, and further use our theory to motivate implementation considerations for FKD in practice. Finally, we experimentally corroborate our theory in the image classification setting, showing that FKD is amenable to ensemble distillation, can transfer knowledge across datasets, and outperforms both vanilla KD & other feature kernel based distillation baselines across a range of standard settings in KD.

Finally, in Chapter 5 (He and Ozay, 2022), we explore the gap between *collapsed* and *whitened* features in self-supervised learning (SSL), where DNNs learn useful features without labelled data. Avoiding collapsed features, a degenerate state where the DNN produces constant output across all inputs, is a common goal shared among different methods in SSL. To that end, whitened features have recently been proposed as a alternative (Ermolov et al., 2021; Zbontar et al., 2021; Bardes et al., 2021) in order to avoid collapse. We identify collapsed and whitened features as two extremes of a spectrum in the features' eigenvalues, and identify power-law decay, parameterised by exponent $\beta \geq 0$, as one way to bridge between them. From this insight, we investigate factors that affect the degree to which features are whitened in SSL, such as projection heads and regularisation strength, and highlight that downstream generalisation performance is not necessarily monotonic in the amount of whitening, particularly in label scarce settings. Motivated by our findings, we propose a method, Post-hoc Manipulation of the Principal Axes & Trace (*PostMan-Pat*), which efficiently post-processes pretrained DNN features to enforce eigenvalue decay rate with power law exponent β, and find that PostMan-Pat delivers improved label efficiency and transferability across a range of SSL settings.

1.2 Omitted work

In this subsection we mention projects we have been a part of, but have not included in this thesis, mainly due to cohesiveness or space limitations:

Stable ResNets (Hayou et al., 2021a) In this work, we study and improve the behaviours of deep residual networks (ResNets) at initialisation, using the signal propagation analysis we describe in Section 2.2.2. Deep ResNet architectures have achieved state of the art performance on many tasks. While they solve the problem of gradient vanishing, they might suffer from gradient exploding as the depth becomes large (Yang and Schoenholz, 2017). Moreover, recent results have shown that ResNets might lose expressivity as the depth goes to infinity (Yang and Schoenholz, 2017; Hayou et al., 2019). To resolve these issues, we introduce a new class of ResNet architectures, called Stable ResNet, that have the property of stabilizing the gradient while ensuring expressivity in the infinite depth limit.

Signal Propagation of Self-Attention layers in Transformers (He et al., 2023) We continue the theme on signal propagation now in this project, which focuses on the self-attention layer and the modifications needed therein, in order to remove skip connections and normalisation layers in deep transformers. Skip connections and normalisation layers form two standard architectural components that are ubiquitous for the training of Deep Neural Networks (DNNs), but whose precise roles are poorly understood. Recent approaches such as Deep Kernel Shaping (Martens et al., 2021) have made progress towards reducing our reliance on them, using insights from wide NN kernel theory to improve signal propagation in vanilla DNNs (which we define as networks without skips or normalisation). However, these approaches are incompatible with the self-attention layers present in transformers, whose kernels are intrinsically more complicated to analyse and control. And so the question remains: *is it possible to train deep vanilla transformers?* We answer this question in the affirmative by designing several approaches that use combinations of parameter initialisations, bias matrices and location-dependent rescaling to achieve faithful signal propagation in vanilla transformers. Our methods address various intricacies specific to signal propagation in transformers, including the interaction with positional encoding and causal masking. In experiments on WikiText-103 and C4, our approaches enable deep transformers without normalisation to train at speeds matching their standard counterparts, and deep vanilla transformers to reach the same performance as standard ones after about 5 times more iterations.

Uncertainty Quantification in Implicit Neural Representations for Computed Tomography (Vasconcelos et al., 2022) Moving now towards uncertainty quantification and Bayesian deep learning (explored further in Chapter 3), we study the predictive uncertainty qualities of implicit neural representations (INRs) (Mildenhall et al., 2020; Sitzmann et al., 2020; Tancik et al., 2020) in the setting of underdetermined computed tomography. Implicit neural representations (INRs) have achieved impressive results for scene reconstruction and computer graphics, where their performance has primarily been assessed on reconstruction accuracy. As INRs make their way into other domains, where model predictions inform high-stakes decision-making, uncertainty quantification of INR inference is becoming critical. To that end, we study a Bayesian reformulation of INRs, UncertaINR, in the context of computed tomography, and evaluate several Bayesian deep learning implementations in terms of accuracy and calibration. We find that they achieve well-calibrated uncertainty, while retaining accuracy compet-

itive with other classical, INR-based, and CNN-based reconstruction techniques. Contrary to common intuition in the Bayesian deep learning literature, we find that INRs obtain the best calibration with Monte Carlo dropout, outperforming more expensive methods like Hamiltonian Monte Carlo and deep ensembles. Moreover, in contrast to the best-performing prior approaches, UncertaINR does not require a large training dataset, but only a handful of validation images.

Probabilistic fine-tuning of pruning masks and PAC-Bayes self-bounded learning (Hayou et al., 2021b) In this work, we study an approach to learning pruning masks by optimizing the expected loss of stochastic pruning masks, i.e., masks which zero out each weight independently with some weight-specific probability. We analyse the training dynamics of the induced stochastic predictor in the setting of linear regression, and observe a data-adaptive L1 regularization term, in contrast to the data adaptive L2 regularization term known to underlie dropout in linear regression. We also observe a preference to prune weights that are less well-aligned with the data labels. We evaluate probabilistic fine-tuning for optimizing stochastic pruning masks for neural networks, starting from masks produced by several baselines (namely, magnitude pruning (Han et al., 2015), SNIP (Lee et al., 2018b), and random masks). In each case, we see improvements in test error over baselines, even after we threshold fine-tuned stochastic pruning masks. Finally, since a stochastic pruning mask induces a stochastic neural network, we consider training the weights and/or pruning probabilities simultaneously to minimize a PAC-Bayes bound on generalization error. Using data-dependent priors (Dziugaite et al., 2021), we obtain a self-bounded learning algorithm with strong performance and numerically tight bounds. In the linear model, we show that a PAC-Bayes generalization error bound is controlled by the magnitude of the change in feature alignment between the "prior" and "posterior" data.

Effectiveness and Resource Requirements of Test, Trace and Isolate Strategies (He et al., 2020b) The Royal Society DELVE initiative is a multi-disciplinary group, convened by the Royal Society at the start of the Covid-19 pandemic, to support a data-driven approach to learning from the different approaches countries are taking to managing the pandemic. In the first of two projects with DELVE, we use an individual-level transmission and contact simulation model to explore the effectiveness and resource requirements of various test-trace-isolate (TTI) strategies for reducing the spread of Covid-19 in the UK, in the context of different scenarios with varying levels of stringency of non-pharmaceutical interventions. Based on modelling results, we show that self-isolation of symptomatic individuals and quarantine of their household contacts has a substantial impact on the number of new infections generated by each primary case. We further show that adding contact tracing of non-household contacts of confirmed cases to this broader package of interventions reduces the number of new infections otherwise generated by 5–15%. We also explore impact of key factors, such as tracing application adoption and testing delay, on overall effectiveness of TTI.

Spatial mapping of Covid-19 pandemic (Teh et al., 2021) In our second and final DELVE project, we present a statistical model designed to capture the fine-grained spatial correlations of a pandemic, as it spreads through a larger region e.g. a country. The spatio-temporal pattern of Covid-19 infections, as for most infectious disease epidemics, is highly heterogenous as a consequence of local variations in risk factors and exposures. Consequently, the widely quoted national-level estimates of reproduction numbers are of limited value in guiding local interventions and monitoring their effectiveness. It is crucial for national and local policy makers, and for health protection teams, that accurate, well-calibrated and timely predictions of Covid-19 incidences and transmission rates are available at fine spatial scales. Obtaining such estimates is challenging, not least due to the prevalence of asymptomatic Covid-19 transmissions, as well as difficulties of obtaining high resolution and high frequency data. In addition, low case counts at a local level further confounds the inference for Covid-19 transmission rates, adding unwelcome uncertainty.

In this work we develop a hierarchical Bayesian method for inference of transmission rates at fine spatial scales. Our model incorporates both temporal and spatial dependencies of local transmission rates in order to share statistical strength and reduce uncertainty. It also incorporates information about population flows to model potential transmissions across local areas. A simple approach to posterior simulation quickly becomes computationally infeasible, which is problematic if the system is required to provide timely predictions. We describe how to make posterior simulation for the model efficient, so that we are able to provide daily updates on epidemic developments.

The results can be found at our website `https://localcovid.info`, which is updated daily to display estimated instantaneous reproduction numbers and predicted case counts for the next weeks, across local authorities in Great Britain. We hope that our methodology and website will be of interest to researchers, policy makers and the public alike, to help identify upcoming local outbreaks and to aid in the containment of Covid-19 through both public health measures and personal decisions taken by the general public.

Chapter 2

Literature Review

In this chapter we review existing literature on which the work included in this thesis builds. We first present an overview on kernel methods/Gaussian processes (GPs) and neural networks (NNs), before delving into the rich theoretical connections that tie together these different ML methods. These connections to kernels and GPs occur in certain wide limits of NNs, and characterise NN behaviour both at initialisation and during training. We discuss the use and limitations of this so-called *kernel learning*, which leads us to describe *feature learning* as an alternative paradigm for NNs. Finally, we discuss concrete examples of NN behaviour: ensembling, knowledge distillation, and self-supervised learning, in the context of kernel and feature learning.

2.1 Background and Notation

Suppose we are given some data $\mathcal{D}$ which we seek to use to train a model f. Unless specified otherwise, in this thesis we consider *supervised learning*, where $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ is a set of input-output pairs, with $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ and $y \in \mathcal{Y} \subset \mathbb{R}^c$ representing inputs and outputs respectively. In this case, our model $f(\cdot) : \mathcal{X} \to \mathcal{Y}$ is a function that takes an unseen input-output pair $(\boldsymbol{x}^*, y^*)$ and outputs a prediction $f(\boldsymbol{x}^*) \in \mathcal{Y}$ that we hope generalises in some way to reflect y^*. For *regression* tasks, we have y^* is a c-dimensional real-valued vector, and we seek to minimise the least squares distance $\|f(\boldsymbol{x}^*) - y^*\|_2^2$. For *classification* tasks with c classes, $y^* \in \mathbb{R}^c$ can be thought of as a one-hot encoding vector that has all entries equal to 0 except for one, where it takes value 1 to denote the true class. In this case, a correct prediction $f(\boldsymbol{x}^*)$ satisfies $\arg\max f(\boldsymbol{x}^*) = \arg\max y^*$.

Typically, we choose $f(\cdot)$ to be a parametric function $f(\cdot, \boldsymbol{\theta})$, with parameters $\boldsymbol{\theta} \in \mathbb{R}^p$, and the training process tries to learn $\boldsymbol{\theta}$ from the available data $\mathcal{D}$. In most circumstances, we learn a single parameter point estimate $\hat{\boldsymbol{\theta}} \in \mathbb{R}^p$ to give predictions $f(\cdot, \hat{\boldsymbol{\theta}})$, but in Section 2.4.1 we also consider an *ensemble* of learnt model parameters $\{\hat{\boldsymbol{\theta}}_k\}_{k=1}^K$ that give an ensemble of predictions $\{f(\cdot, \hat{\boldsymbol{\theta}}_k)\}_{k=1}^K$.

There are a variety of ways that we can choose to construct the parameter space of the function $f(\cdot, \boldsymbol{\theta})$, and the choice that we make has important implications for the properties of our trained model. We spend the rest of this section describing some different possibilities for $f(\cdot, \boldsymbol{\theta})$, starting with the simplest and arguably most famous: the linear model.

2.1.1 Linear models

In the linear model, we choose $f(\boldsymbol{x}, \boldsymbol{\theta})$ be a linear function in the input data $\boldsymbol{x}$:

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{x}^\top \boldsymbol{\theta} \tag{2.1}$$

for parameters $\boldsymbol{\theta} \in \mathbb{R}^{d \times c}$. For simplicity, we assume output dimension $c = 1$, though our exposition is straightforwardly extended to settings with $c > 1$.

In statistics, the linear model is commonly used in the regression setting, reflecting an assumption that the output y is generated according to a linear dependence in $\boldsymbol{x}$ with some additive noise:

$$y = \boldsymbol{x}^\top \boldsymbol{\theta}^* + \epsilon \tag{2.2}$$

where $\epsilon \in \mathbb{R}^c$ represents random noise and $\boldsymbol{\theta}^* \in \mathbb{R}^{d \times c}$ is an unknown parameter value that we seek to approximate.

If we assume our training data $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ consists of n input-output pairs, let us define $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{n \times c}$ to have row i equal to $\boldsymbol{x}_i^\top$ and y_i respectively. Moreover, for now we assume that $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$ is invertible, which will be the case when $n > d$ and $\mathbf{X}$ is full rank. Then, one way to estimate $\boldsymbol{\theta}$ is by *least squares*, which gives the following estimator, $\hat{\boldsymbol{\theta}}_{\text{LS}}$:

$$\hat{\boldsymbol{\theta}}_{\text{LS}} := \arg\min_{\boldsymbol{\theta}} \left\{ \|Y - \mathbf{X}\boldsymbol{\theta}\|_2^2 \right\} \tag{2.3}$$

$$= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top Y \tag{2.4}$$

The tractability of Equation (2.4) owes to the fact that the least squares objective Equation (2.3) is convex in the parameters $\boldsymbol{\theta}$, and the uniqueness of $\hat{\boldsymbol{\theta}}_{\text{LS}}$ owes to the invertibility of $\mathbf{X}^\top \mathbf{X}$.

Using $\hat{\boldsymbol{\theta}}_{\text{LS}}$, we obtain predictions:

$$\hat{Y}_{\text{LS}} := \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top Y \tag{2.5}$$

which can be thought of intuitively as a linear projection of Y onto the column space of $\mathbf{X}$.

Under the additional assumption that noise terms ϵ_i are zero mean, uncorrelated, and have constant variance (*homoscedastic*) across inputs $i \leq n$, it is straightforward to see that our estimated $\hat{\boldsymbol{\theta}}_{\text{LS}}$ is unbiased $\mathbb{E}\big(\hat{\boldsymbol{\theta}}_{\text{LS}}\big) = \boldsymbol{\theta}^*$. Moreover, the Gauss-Markov theorem tells us that the least squares estimator $\hat{\boldsymbol{\theta}}_{\text{LS}}$ is the best linear unbiased estimator under our assumptions, such that for any other unbiased estimator $\tilde{\boldsymbol{\theta}}$ that is linear in Y, the variance of $\tilde{\boldsymbol{\theta}}$ is larger than the variance of $\hat{\boldsymbol{\theta}}_{\text{LS}}$, in the sense that $\text{Var}(\tilde{\boldsymbol{\theta}}) - \text{Var}(\hat{\boldsymbol{\theta}}_{\text{LS}})$ is positive semi-definite.

Ridge regression When we have non-invertible $\mathbf{X}^\top \mathbf{X}$, the optimisation problem is still convex but now has an infinite hyperplane of solutions. One common solution to avoid this under-determined nature is *ridge regression* (Hoerl and Kennard, 1970): we add L_2 regularisation on parameters with strength $\sigma^2 > 0$, giving unique solutions regardless of the invertibility of $\mathbf{X}^\top \mathbf{X}$:

$$\hat{\boldsymbol{\theta}}_\sigma := \arg\min_{\boldsymbol{\theta}} \left\{ \|Y - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \sigma^2 \|\boldsymbol{\theta}\|_2^2 \right\} \tag{2.6}$$

$$= (\mathbf{X}^\top \mathbf{X} + \sigma^2 I)^{-1} \mathbf{X}^\top Y \tag{2.7}$$

While the regularised objective, Equation (2.6), results in an biased estimator $\hat{\boldsymbol{\theta}}_\sigma$, it can be shown that $\hat{\boldsymbol{\theta}}_\sigma$ actually outperforms the least squares estimator $\hat{\boldsymbol{\theta}}_{\text{LS}}$ for certain values of σ^2 (Hoerl and Kennard, 1970) as this bias is offset by reduced variance as the parameters are shrinked towards 0.

For our purposes, studying the fitted values, $\hat{Y}_\sigma$, with ridge estimated parameters, Equation (2.7), yields a new perspective on the linear model. Namely:

$$\hat{Y}_\sigma := \mathbf{X}(\mathbf{X}^\top \mathbf{X} + \sigma^2 I)^{-1} \mathbf{X}^\top Y \tag{2.8}$$

$$= \mathbf{X}\mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \sigma^2 I)^{-1} Y \tag{2.9}$$

$$= \Sigma(\Sigma + \sigma^2 I)^{-1} Y \tag{2.10}$$

where $\Sigma = \mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{n \times n}$ is a *Gram matrix* that measures the linear pairwise similarities between the different inputs $\{\boldsymbol{x}_i\}$. Thus, one can reinterpret the fitted values under ridge regression, $\hat{Y}_\sigma$, as a weighted average of training outputs Y, where the weights are determined by how linearly "similar" different training inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$ are, in terms of their linear inner products $\boldsymbol{x}_i^\top \boldsymbol{x}_j$.

2.1.2 Kernel methods

Of course, the linear model only considers linear relationships in the input data, which may be unnecessarily restrictive, and we may wish to model non-linear dependencies. An obvious extension that bypasses linearity would be to first map our inputs through a nonlinear *feature mapping* $\phi(\cdot) : \mathcal{X} \to \mathcal{H}$, for some inner product space $\mathcal{H}$, before comparing linear similarities via inner products $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ in feature space. This brings us now to introduce kernel methods.

A kernel k, Definition 2.1, measures some degree of similarity between two inputs $\boldsymbol{x}, \boldsymbol{x}'$. When the kernel $k(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}$ is large it indicates that $\boldsymbol{x}$ & $\boldsymbol{x}'$ are similar, and conversely when $k(\boldsymbol{x}, \boldsymbol{x}')$ is small the two inputs are dissimilar.

Definition 2.1. *A (positive definite) kernel $k : \mathcal{X} \times \mathcal{X} \to \mathcal{H}$ is a symmetric real-valued function such that, for any collection of n inputs $\{\boldsymbol{x}_i\}_{i=1}^n$ for some $n > 0$, the $n \times n$ matrix K with entries $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semi-definite.*

Examples We have already seen an example of a common kernel in Section 2.1.1: the *linear* kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$. As mentioned, an appeal to using kernels is that they can capture potentially non-linear interactions in input space, with different choices of kernel leading to different notions of similarity. A popular example of such a choice is the *Gaussian* kernel, defined as:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(- \frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\gamma^2} \right) \tag{2.11}$$

which decays exponentially as the squared distance between $\boldsymbol{x}, \boldsymbol{x}'$ increases, controlled by a length-scale $\gamma > 0$.

Given a feature map $\phi(\cdot) : \mathcal{X} \to \mathcal{H}$, it is easy to construct a viable kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$. On the other hand, the converse is also true, courtesy of the Moore-Aronszajn theorem:

Theorem 2.2 (Moore-Aronszajn (Aronszajn, 1950))**.** *For any kernel k, there exists a feature map $\phi(\cdot) : \mathcal{X} \to \mathcal{H}$ with $\mathcal{H}$ an inner product space, such that $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$.*

In fact, the proof of the Moore-Aronszajn theorem allows us to construct such a feature map $\phi(\cdot)$:

$$\phi(\boldsymbol{x}) = k(\cdot, \boldsymbol{x}),$$

with $\mathcal{H}$ defined as a vector space of functions of form:

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, \boldsymbol{x}_i), \tag{2.12}$$

for some $n \in \mathbb{N}$, with $\boldsymbol{x}_i \in \mathcal{X}$ and $\alpha_i \in \mathbb{R}$ $\forall i \leq n$.

For two functions $f, g \in \mathcal{H}$, with $f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, \boldsymbol{x}_i)$ and $g(\cdot) = \sum_{j=1}^{n'} \beta_j k(\cdot, \boldsymbol{x}'_j)$, the following candidate $\langle f, g \rangle_{\mathcal{H}}$ satisfying

$$\langle f, g \rangle_{\mathcal{H}} := \alpha^\top K_{\mathbf{X}, \mathbf{X}'} \beta, \tag{2.13}$$

can be shown to define a valid inner product of f and g in $\mathcal{H}$, where $\alpha \in \mathbb{R}^n$ has entries $(\alpha_i)_{i=1}^n$, $\beta \in \mathbb{R}^{n'}$ has entries $(\beta_j)_{j=1}^{n'}$ and $\left(K_{\mathbf{X}, \mathbf{X}'}\right)_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}'_j)$. And in turn, the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ induces a norm on $\mathcal{H}$:[1]

$$\|f\|_{\mathcal{H}}^2 := \langle f, f \rangle_{\mathcal{H}} = \alpha^\top K_{\mathbf{X}, \mathbf{X}} \alpha. \tag{2.14}$$

Kernel ridge regression Recall that linear ridge regression corresponds to the solution of a norm regularised least squares objective in parameter space (Equation (2.6)). Similarly, *kernel ridge regression* (Saunders et al., 1998) can be obtained as the solution of a norm regularised least squares objective in function space $\mathcal{H}$, giving solution $\hat{f}_\sigma$:

$$\hat{f}_\sigma := \underset{f \in \mathcal{H}}{\arg\min} \left\{ \|Y - f(\mathbf{X})\|_2^2 + \sigma^2 \|f\|_{\mathcal{H}}^2 \right\} \tag{2.15}$$

Though Equation (2.15) involves an optimisation problem over a potentially infinite-dimensional space, using the representer theorem (Kimeldorf and Wahba, 1970; Schölkopf et al., 2001), it can be shown that, at input $\boldsymbol{x}$, kernel ridge regression gives prediction:

$$\hat{f}_\sigma(\boldsymbol{x}) = K_{\boldsymbol{x}, \mathbf{X}} (K_{\mathbf{X}, \mathbf{X}} + \sigma^2 I)^{-1} Y \tag{2.16}$$

Much like how the linear kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$ determines the Gram matrix that appears in linear ridge regression (Equation (2.10)), the choice of kernel k determines the Gram matrix in the kernel ridge regression predictor (Equation (2.16)). Thus, we note $\hat{f}_\sigma$ still constitutes a weighted average of training outputs Y, but now with weights (and the notions of "similarity") determined by the choice of kernel k.

[1] With some care concerning the *completion* of $\mathcal{H}$, it can be shown that $\mathcal{H}$ is not just an inner product space but a special type of Hilbert space known as a *reproducing kernel Hilbert space* (RKHS).

2.1.3 Gaussian Processes

While kernel methods can be thought of as allowing one to construct and optimise predictors over a rich space of functions, in some cases we may wish to work with a probability distribution of functions. Gaussian Processes (Rasmussen and Williams, 2005) can be thought of as an probabilistic extension of kernel methods, that allow one to model uncertainty in predictions and perform inference over functions.

A Gaussian Process (GP), is a *stochastic process* (a family of random variables indexed over an input space $\mathcal{X}$), satisfying the property that the joint distribution of any finite collection of these random variables is multivariate Gaussian. A GP, f, depends on a choice of kernel k, and mean function $\mu : \mathcal{X} \to \mathbb{R}$, with notation $f \sim \mathcal{GP}(\mu, k)$.

The choice of kernel k determines the covariance structure among the GP's evaluations, and we typically consider the mean function $\mu(\cdot)$ to be the constant zero function. More specifically, for any set of inputs $\{\boldsymbol{x}_i\}_{i=1}^n$, a GP $f \sim \mathcal{GP}(0, k)$ satisfies $\{f(\boldsymbol{x}_i)\}_{i=1}^n \sim \mathcal{N}(0, K)$, where $K \in \mathbb{R}^{n \times n}$ has entries $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Since $f(\boldsymbol{x})$ is defined over all possible values of $\boldsymbol{x} \in \mathcal{X}$, we can think of f as establishing a distribution over random functions on $\mathcal{X}$, with kernel k controlling the likelihood associated to different functions.

GP posteriors for regression GPs are commonly used in Bayesian inference to model a prior distribution, $p(f)$, over functions. Given observed data $\mathcal{D} = (\mathbf{X}, Y) = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, and an observation model $p(y|f(\boldsymbol{x}))$ of how likely an output y is for a specific input $\boldsymbol{x}$ and function f, we can apply Bayes' rule to compute the posterior:

$$p(f|\mathcal{D}) \propto p(Y|f(\mathbf{X}))p(f). \tag{2.17}$$

The Bayesian paradigm is popular across statistics and machine learning, owing to the fact that it enables a principled way to update prior beliefs in light of observed data (Ramsey, 1926; de Finetti, 1931). Unfortunately, the posterior, Equation (2.17), is often computationally prohibitive and intractable, especially over large parameter spaces or function spaces for GPs, in which case one needs to resort to approximate inference (MacKay et al., 2003; Jordan et al., 1999; Andrieu et al., 2003).

However, one appealing property of GPs is that the GP posterior is *analytic* in the case of regression. Recall in regression problems we seek to minimise the least squares objective $\|f(\mathbf{X}) - Y\|_2^2$ on training data $\mathcal{D} = (\mathbf{X}, Y)$, which is equivalent to maximising the likelihood of the Gaussian observation model with independent noise $p(Y|f(\mathbf{X})) \sim \mathcal{N}(f(\mathbf{X}), \sigma^2 I)$. It turns out with such a Gaussian observation model, data $\mathcal{D} = (\mathbf{X}, Y)$, and a GP prior $f \sim \mathcal{GP}(0, k)$, the analytic posterior predictive distribution of $f(\boldsymbol{x})$ at a new input $\boldsymbol{x}$ is:

$$f(\boldsymbol{x})|\mathcal{D} \sim \mathcal{N}\big(K_{\boldsymbol{x},\mathbf{X}}(K_{\mathbf{X},\mathbf{X}} + \sigma^2 I)^{-1} Y,\ \ K_{\boldsymbol{x},\boldsymbol{x}} - K_{\boldsymbol{x},\mathbf{X}}(K_{\mathbf{X},\mathbf{X}} + \sigma^2 I)^{-1} K_{\mathbf{X},\boldsymbol{x}}\big) \tag{2.18}$$

which can be derived by the fact that the joint distribution of $(Y, f(\boldsymbol{x}))$ is Gaussian under our model, and making use of Gaussian conditional distributions.

We note that the mean predictive of the GP posterior, $K_{\boldsymbol{x},\mathbf{X}}(K_{\mathbf{X},\mathbf{X}} + \sigma^2 I)^{-1} Y$, is exactly the kernel ridge regression predictor, Equation (2.16). Thus, the GP posterior for regression can be thought of as taking the kernel ridge regression predictor and adding *predictive uncertainty* with variance $K_{\boldsymbol{x},\boldsymbol{x}} -$

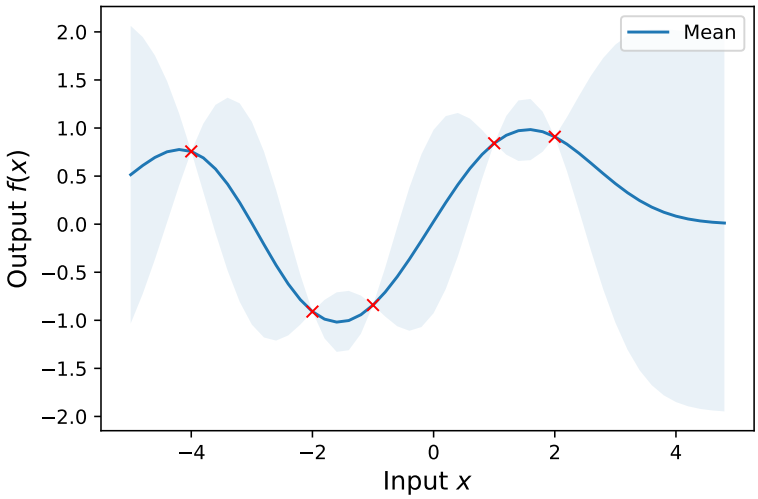


Figure 2.1: GP posterior predictive, with red crosses indicating observed data

$K_{\boldsymbol{x},\mathbf{X}}(K_{\mathbf{X},\mathbf{X}}+\sigma^2 I)^{-1}K_{\mathbf{X},\boldsymbol{x}}$, which is determined by how similar the new input $\boldsymbol{x}$ is to the training inputs $\mathbf{X}$, in the eyes of kernel k. Intuitively, inputs that the kernel deems are more similar to the training inputs will have smaller posterior variance, and vice versa. We plot an example of GP posterior predictive mean and variances in Fig. 2.1, using the Gaussian kernel.

2.1.4 Random features

A key issue with both kernels and GPs is scalability: the $n \times n$ matrix inverses in the posterior predictive, Equation (2.18), require $O(n^3)$ computational complexity and $O(n^2)$ memory, which is prohibitive for large dataset sizes n. One popular way to make kernels/GPs more scalable is to use *random features*, which approximate the kernel computation $k(\boldsymbol{x}, \boldsymbol{x}')$ with a cheaper alternative.

Recall that the Moore-Aronszajn theorem (Theorem 2.2) gave us an abstract and uncomputable feature map $\phi(\boldsymbol{x}) = k(\cdot, \boldsymbol{x}) \in \mathcal{H}$ from which we could evaluate the kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$. The idea in random features is instead to construct an approximate feature map $\phi(\boldsymbol{x}) \in \mathbb{R}^m$, satisfying $k(\boldsymbol{x}, \boldsymbol{x}') \approx \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$, that lies in a real vector space of finite dimension $m < \infty$. Given a training input set $\mathbf{X} \in \mathbb{R}^{n \times d}$, we can compute and store the entire set of feature vectors $\Phi = \phi(\mathbf{X}) \in \mathbb{R}^{n \times m}$, and approximate the Gram matrix as $K_{\mathbf{X},\mathbf{X}} \approx \Phi\Phi^\top$.

Through a series of algebraic manipulations we can use random features to offset the cubic complexity when computing the posterior predictives. For example, we can estimate the posterior mean/kernel ridge predictor as

$$K_{\boldsymbol{x},\mathbf{X}}(K_{\mathbf{X},\mathbf{X}} + \sigma^2 I)^{-1} Y \approx \phi(\boldsymbol{x})\Phi^\top(\Phi\Phi^\top + \sigma^2 I)^{-1} Y \tag{2.19}$$

$$= \phi(\boldsymbol{x})(\Phi^\top\Phi + \sigma^2 I)^{-1}\Phi^\top Y, \tag{2.20}$$

which only requires inverting an $m \times m$ matrix and, at $O(m^3 + nm^2)$ computational cost, scales linearly in n. Here m is a hyperparameter we choose, that typically controls the trade off between computational complexity and approximation quality.

Example For the Gaussian kernel, $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\gamma^2}\right)$, Rahimi and Recht (2007) constructed the *random fourier feature* map:

$$\phi(\boldsymbol{x}) = \frac{\sqrt{2}}{m}\big(\cos(W_1\boldsymbol{x} + u_1), \ldots, \cos(W_m\boldsymbol{x} + u_m)\big), \tag{2.21}$$

where $\{W_i\}_{i=1}^m \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \gamma^{-2}I)$ and $\{u_i\}_{i=1}^m \overset{\text{i.i.d.}}{\sim} \text{Unif}[-\pi, \pi]$. From Bochner's theorem (e.g. (Rudin, 1962), we know that $\mathbb{E}[\phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')] = k(\boldsymbol{x}, \boldsymbol{x}')$ with $O(\frac{1}{m})$ variance. This means that the approximation quality (and computational requirements) increase with m, and in the $m \to \infty$ limit, the random features approximation becomes exact due to the law of large numbers.

We note that the random feature predictor, Equation (2.20), is equivalent to the linear ridge regression predictor, Equation (2.10), but replacing input $\boldsymbol{x}$ with the feature map $\phi(\boldsymbol{x})$. That is to say, if we modelled f as a linear model in features ϕ,

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \phi(\boldsymbol{x})^\top \boldsymbol{\theta} \tag{2.22}$$

with $\boldsymbol{\theta} \in \mathbb{R}^{m \times c}$, and solved the regularised objective for parameters

$$\hat{\boldsymbol{\theta}}_\sigma := \underset{\boldsymbol{\theta} \in \mathbb{R}^{m \times c}}{\arg\min} \left\{ \|Y - \phi(\mathbf{X})\boldsymbol{\theta}\|_2^2 + \sigma^2 \|\boldsymbol{\theta}\|_2^2 \right\}, \tag{2.23}$$

we would end up with exactly the random feature approximation to kernel ridge regression, Equation (2.20). Combining these insights, we see that the random features approximation to kernel methods/GPs becomes exact as the number of parameters, dictated by m, increases to infinity.

2.1.5 Neural Networks

Both kernels and GPs are *non-parameteric* methods that construct predictive functions without a fixed parameter space, and instead can be thought of as corresponding to an infinite number of parameters. On the other hand, the random features model is a parametric approximation to kernels/GPs, by training the parameters of a linear model in some *untrainable*, but nonlinear, feature map $\phi(\cdot) : \mathcal{X} \to \mathbb{R}^m$: $f(\boldsymbol{x}, \boldsymbol{\theta}) = \phi(\boldsymbol{x})^\top \boldsymbol{\theta}$. Neural networks (NNs) are parametric models and similar in this regard to random features, but with the key difference that the feature map ϕ has additional trainable parameters, and hence isn't fixed during training.

As the DL community has applied NNs to increasingly many tasks and data types in recent years, the range of possibilities for how to construct the feature extractor, ϕ, has grown too at a phenomenal rate. All deep NN (DNN) *architectures* we consider in this thesis will have a *feedforward* structure, where for some nonlinear operation $\mathcal{F}$ and depth L, an input $\boldsymbol{x}$ is propagated through the network recursively as:

$$h_0(\boldsymbol{x}) = \boldsymbol{x}, \tag{2.24}$$

$$h_l(\boldsymbol{x}) = \mathcal{F}(h_{l-1}(\boldsymbol{x}), \boldsymbol{\theta}_l), \;\; \forall 1 \leq l \leq L \tag{2.25}$$

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = h_L(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}, \tag{2.26}$$

with trainable parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{L+1}\}$.

Here, $h_l(\boldsymbol{x})$ can be thought of as "hidden" layers internal to the DNN, and the final feature extractor $\phi = h_L$ is the final hidden layer. The feedforward name comes from the fact that there is a hierarchy

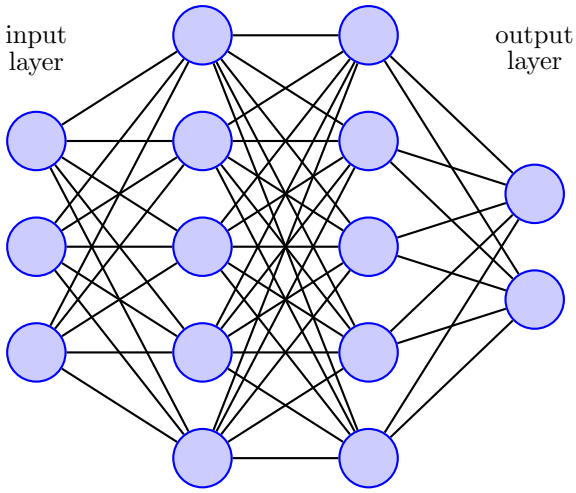


Figure 2.2: Example MLP architecture

of hidden layers that do not form a cycle, one feeding the next, from input to output. Particular architectures of note include the residual convolutional NN (CNN) (He et al., 2016b), which is particularly suited for image data, and the transformer (Vaswani et al., 2017), which has seen widespread success but particularly excels at processing sequential data, like natural language.

Example The *multilayer perceptron* (MLP) (Rosenblatt, 1958) is one of the most basic and widely used NN architectures. We will use it as a running example, though the results we discuss later have extensions to general architectures. The MLP is a fully-connected feedforward network of different hidden layers that are sequentially composed. For depth L, and a width $d_l \in \mathbb{N}$ for each layer $l \leq L$ (we take $d_0 = d$ and $d_{L+1} = c$ by default), an input, $\boldsymbol{x}$, is propagated through the MLP as:

$$g_l(\boldsymbol{x}) = h_{l-1}(\boldsymbol{x})^\top W_l + b_l, \tag{2.27}$$

$$h_l(\boldsymbol{x}) = \psi(g_l(\boldsymbol{x})) \;\; \forall 1 \leq l \leq L. \tag{2.28}$$

Here, the *activation*, $\psi : \mathbb{R} \to \mathbb{R}$, is a non-linear function that is applied element-wise, and we have trainable weights $W_l \in \mathbb{R}^{d_{l-1} \times d_l}$ and biases $b_l \in \mathbb{R}^{d_l}$ parameters, giving $\boldsymbol{\theta}_l = \{W_l, b_l\}$. Popular activation functions include ReLU, $\psi(x) := \max(0, x)$, or sigmoid, $\psi(x) := (1 + e^{-x})^{-1}$. We plot an example architecture with $L = 2$ hidden layers in Fig. 2.2.

For some weight and bias variances σ_W^2, σ_b^2, we consider Gaussian initialised weights, W_l^0 & b_l^0, scaled as

$$W_l^0 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{\sigma_W^2}{\text{fan-in}}), \;\; \text{and} \;\; b_l^0 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_b^2), \tag{2.29}$$

where fan-in denotes the dimension of the incoming features (d_{l-1} in our notation). This "fan-in" scaling choice is standard (LeCun et al., 2012; He et al., 2015), and is necessary to preserve the variance of the *pre-activations*, $g_l(\boldsymbol{x})$, to be $\Theta(1)$ at initialisation, across layers l.[2]

We see, in Equation (2.28), that the MLP feature map,

$$\phi(\boldsymbol{x}) = h_L(\boldsymbol{x}), \tag{2.30}$$

[2]We use standard mathematical notation $a(m) = \Theta(b(m))$ such that $\exists 0 < k_1, k_2 < \infty$ satisfying $k_1 \leq \frac{a(m)}{b(m)} \leq k_2$, for all sufficiently large m. Likewise, the "big-O" notation $a(m) = O(b(m))$ is the same, but allows for the case $k_1 = 0$

is constructed through a composition of trainable linear and non-linear operations. This gives the network flexibility to adapt ϕ during the training process, relative to say random features models, but at the expense of losing the convexity and analytic tractability of training compared to the previous models described thus far.

Instead of an analytic solution, NN parameters are typically initialised randomly (LeCun et al., 2012; Glorot and Bengio, 2010; He et al., 2015) e.g. like in Equation (2.29), and iteratively updated with a gradient-based local optimisation method (Bottou, 2012; Tieleman et al., 2012; Zeiler, 2012; Duchi et al., 2011; Kingma and Ba, 2014; Martens and Grosse, 2015) to minimise some loss, such as the regularised least-squares objective (Equations (2.6) and (2.23)):

$$\mathcal{L}(\boldsymbol{\theta}) = \left\{ \|Y - f(\mathbf{X}, \boldsymbol{\theta})\|_2^2 + \sigma^2 \|\boldsymbol{\theta}\|_2^2 \right\}, \tag{2.31}$$

Due to efficient gradient computation using auto-differentiaton, NNs scale linearly both in model and dataset sizes, and this scalability is crucial to their success. Given non-convexity in parameter space, the initialisation of NNs becomes an important consideration in determining the final performance of a trained model. In the next section, we investigate the implications of NN initialisation in the context of large width, and review known connections to kernel methods and GPs.

2.2 Kernel Learning in Neural Networks

In Section 2.1, we presented kernel methods and GPs as (related) non-parametric methods that define predictors in the space of functions, and effectively correspond to infinite-dimension parameter spaces. We also saw how random features models are parametric approximations to kernel-based methods that become exact as the parameter dimension increases to infinity. Surprisingly, it turns out that NNs also become kernel-based methods under certain settings when the *width* (and hence number of parameters) of the NN increases to infinity, as we describe in this section. Besides the useful analytic tractability that one can obtain, studying the large-width setting for NNs is an interesting endeavour in its own right as understanding the limiting properties for our models can inform us and shed light on the behaviours of finite models. This is particularly the case for NNs, where a general trend appears to be that bigger models perform better (Krizhevsky et al., 2012; Zagoruyko and Komodakis, 2016b; Belkin et al., 2019; Huang et al., 2019; Bubeck and Sellke, 2021; Yang et al., 2022a).

2.2.1 Kernel Correspondence for wide Neural Networks

The connection between wide NNs and kernels/GPs dates back to Neal (1996), who observed that if one initialises a single hidden layer MLP with i.i.d. Gaussian weights, then the induced NN function converges in distribution to a GP in the limit of large width d_1. Two concurrent works (Lee et al., 2018a; Matthews et al., 2018) extended the GP correspondence for randomly initialised wide MLPs of *any* depth.

Lee et al. (2018a) coined the term *Neural Network Gaussian Process* (NNGP) to describe the resulting GP and its kernel,[3] and focused on the implications of their result in connecting SGD-trained NNs,

[3]Other works in the literature have referred to the NNGP kernel as the *conjugate* kernel (Daniely, 2017)

Bayesian inference in NNs, and the NNGP posterior. Matthews et al. (2018) proved their result in a more general and realistic setting, where the widths at deep layers grow wide jointly, as opposed to the simpler setting where widths increase sequentially, one layer after another. Various other works extended the NNGP correspondence for wide NNs to different NN architectures with some notion of large width, including CNNs (Novak et al., 2019; Garriga-Alonso et al., 2019) with many channels and transformers with either many attention heads (Hron et al., 2020b) or large model dimension (Yang, 2019b). In each case, the NNGP kernel depends on the choice of model architecture, including the depth, as well as the non-linear activation functions and weight initialisations used. We describe how the NNGP kernel can be computed recursively in depth in Section 2.2.2.

Tensor Programs Yang (2019a,b, 2020a,b) presented a general framework, *Tensor Programs* (TP), to automate and prove limiting behaviour in wide NNs of almost any architecture, including compatibility with recurrent NNs (e.g. LSTMs Hochreiter and Schmidhuber (1997)), skip connections (He et al., 2016a), and batch normalisation (Ioffe and Szegedy, 2015). The appeal of the TP framework is that it is sufficiently powerful and general to encapsulate essentially all results concerning infinite-width DNNs, including the aforementioned NNGP limit, as well as later results on the Neural Tangent Kernel (Jacot et al. (2018); Section 2.2.3) and feature learning (Yang and Hu (2020); Section 2.3).

To enable this, the authors introduced a general set of permissible computations, (e.g. matrix multiplication, elementwise non-linearities, linear combinations) corresponding to an NN computation graph and show under general conditions that any NN that is expressible in terms of these permissible computations satisfies certain limiting properties in large width $m \to \infty$.[4] Such an NN computation graph is referred to as a Tensor Program.

In particular, at any width m, the TP framework defines a set of N so-called *G-vars*, $\{g^i\}_{i=1}^N$,[5] which are particular vectors of dimension m that appear in the NN computation graph after a matrix multiplication. Intuitively, a G-var has entries that can be thought of as being i.i.d. Gaussian as they are a sum of many independent terms, which follows a central limit theorem if correctly scaled. For example, in our MLP example (Equation (2.28)) the pre-activations $g_l(\boldsymbol{x}), g_l(\boldsymbol{x}')$, for any layer l and any two training inputs $\boldsymbol{x}, \boldsymbol{x}'$, appear in the initial forward pass computations after a matrix multiplication with random weights W_l, $g_l(\boldsymbol{x}) = h_{l-1}(\boldsymbol{x})^\top W_l + b_l$, and are considered G-vars. The key technical insight in TP is then to show that there is a *single* N-dimensional Gaussian distribution, $\mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^N$ and covariance $\Sigma \in \mathbb{R}^{N \times N}$, that controls the behaviour of these N different m-dimensional G-vars at large width. Formally, this is shown in the so-called TP Master Theorem (Yang, 2019a), which we restate in Theorem 2.3.

The construction for the G-vars' mean μ and covariance Σ depends on the precise NN computation graph involved in obtaining different G-vars, as well as the distribution of the weights. The first TP paper (Yang, 2019b) considers only i.i.d. "fan-in" Gaussian $\mathcal{N}(0, \frac{\sigma_W^2}{m})$ weights, which is sufficient for the NNGP correspondence at initialisation. Subsequent TP papers extend both the permissible NN computations and corresponding constructions of G-vars, e.g. Yang (2020a) allows backwards computation to obtain

[4] For simplicity of presentation we suppose that all layers have the same width m, which increases together across layers.

[5] Here, $N \in \mathbb{N}$ is width-independent and finite, and is solely a function of the NN computation graph.

gradients by considering weight matrix transposes, whilst Golikov and Yang (2022) permit non-Gaussian initialisations.

Theorem 2.3 (Tensor Program Master Theorem (Yang, 2019a), informal)**.** *Consider any NN computation graph that satisfies the requirements to be a TP, with associated G-vars,* $\{g^i\}_{i=1}^N$*. Then, we can construct an* N*-dimensional Gaussian distribution* $\mathcal{N}(\mu, \Sigma)$*, such that for any well-behaved function* $\xi : \mathbb{R}^N \to \mathbb{R}$*, as width* $m \to \infty$*:*

$$\frac{1}{m}\sum_{\alpha=1}^{m} \xi(g_\alpha^1, g_\alpha^2, \ldots, g_\alpha^N) \xrightarrow{a.s.} \mathbb{E}_{Z\sim\mathcal{N}(\mu,\Sigma)}[\xi(Z^{g^1}, Z^{g^2}, \ldots, Z^{g^N})] \tag{2.32}$$

where $\xrightarrow{a.s.}$ *means almost sure convergence.*

Intuitively, the Master Theorem (Theorem 2.3) shows that the joint distribution of the m-dimensional G-vars, $\{g_\alpha^i\}_{i=1}^N$, "sliced" at a particular dimension, α, can be thought of as being identically and independently distributed across $\alpha \leq m$. This is in the sense that empirical averages converge to a limit determined by $\mathcal{N}(\mu, \Sigma)$, akin to a law of large numbers. Thus, to compute any empirical average involving G-vars, it is sufficient to simply treat each α-slice as coming from the same distribution $Z \sim \mathcal{N}(\mu, \Sigma)$, and track this distribution instead.

The TP master theorem (Theorem 2.3) is a powerful result, and many non-trivial results about wide NNs drop out as simple corollaries. For example, it is straightforward to show that inner products of NN feature maps $\phi(\cdot) = h_L(\cdot) = \psi(g_L(\cdot))$ converge to a limit (the aforementioned NNGP kernel) by noting that final layer pre-activations $g_L(\boldsymbol{x}), g_L(\boldsymbol{x}')$ are both G-vars. We present this in the case of an MLP below:

Corollary 2.4 (NNGP kernel convergence of MLP feature map inner-products)**.** *Consider an MLP, Equation* (2.28)*, that is linear function of features* $\phi(\cdot) = \psi(g_L(\cdot))$*. Then, for any two inputs* $\boldsymbol{x}, \boldsymbol{x}'$*, the normalised inner product,*

$$\frac{1}{m}\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle = \frac{1}{m}\sum_{\alpha=1}^{m} \psi(g_l(\boldsymbol{x}))_\alpha \psi(g_l(\boldsymbol{x}'))_\alpha, \tag{2.33}$$

has a determinstic kernel limit, $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$*, as* $m \to \infty$*.* $\mathcal{K}$ *is known as the NNGP kernel.*

From this NNGP kernel convergence, it is simple to show that the output function of a wide NN at initialisation converges in distribution to a GP, $f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K})$, with kernel that is the NNGP $\mathcal{K}$. To see this, we write $f(\boldsymbol{x}) = \psi(g_L(\boldsymbol{x}))^\top W_L$ (omitting bias parameters for simplicity) for last layer linear weights $W_L \in \mathbb{R}^{m\times c}$, and use the following proposition, again from Yang (2019b):

Proposition 2.5 (Convergence of output to Gaussian given convergence 2nd moments, Proposition G.4 of Yang (2019b))**.** *For each* $m \in \mathbb{N}$*, consider a sequence of random vectors of length* m*,* $\{\phi^a\}_{a=1}^n$*, for some* $n \in \mathbb{N}$ *that is independent of* m*. Suppose that as* $m \to \infty$*,* $\frac{1}{m}\phi^{a\top}\phi^{a'} \xrightarrow{d} \Sigma_{a,a'}$ *for some deterministic p.s.d. matrix* $\Sigma \in \mathbb{R}^{n\times n}$*,* $\quad \forall a, a' \leq n$*. Suppose further that at each* m*,* $W_L \in \mathbb{R}^{m\times c}$ *is sampled with i.i.d. entries from* $\mathcal{N}(0, \frac{\sigma_f^2}{m}I)$ *and denote* $\Phi \in \mathbb{R}^{n\times m}$ *to have row* $a \leq n$ *equal to* $\phi_a^\top$*. Then, if we define*

$$f = \Phi W_L \in \mathbb{R}^{n\times c}, \tag{2.34}$$

we have that $f \xrightarrow{d} \mathcal{N}(0, \Sigma^f)$ *as* $m \to \infty$*, where the covariance* $\Sigma^f = \{\Sigma^f_{a,b,a',b'}\}_{a \leq n, b \leq c, a' \leq n, b' \leq c}$ *satisfies*

$$(\Sigma^f)_{a,b,a',b'} = \begin{cases} \sigma_f^2 \Sigma_{a,a'} & \text{if } b = b', \\ 0 & \text{else.} \end{cases} \tag{2.35}$$

Using Proposition 2.5, combined with the NNGP kernel limit for wide NN feature map inner-products (e.g. Corollary 2.4 for an MLP), the convergence of a wide NN function at initialisation to a GP with NNGP kernel $\mathcal{K}$ (matching the earlier results of (Neal, 1996; Lee et al., 2018a; Matthews et al., 2018)) is easy to show:

Corollary 2.6 (NNGP function output convergence)**.** *A wide NN function converges in distribution to a GP, with kernel that is defined by the NNGP kernel limit of last layer feature map inner products.*

From a Bayesian perspective that treats NN parameter initialisation as inducing a prior distribution, one can thus view the NNGP as the limiting function-space prior for a wide Bayesian NN. Adlam et al. (2020) used this initialisation-time NNGP correspondence to study the uncertainty quantification of infinite-width Bayesian NNs, facilitated by the fact that the Bayesian NN posterior also converges to the NNGP posterior with large width (Hron et al., 2020a). Other lines of work have studied different models that combine aspects of GPs with DNNs (Damianou and Lawrence, 2013; Wilson et al., 2016; Aitchison et al., 2021).

2.2.2 Signal Propagation in wide DNNs

We have seen in the previous section that despite distinct motivations, there are many ways to view wide DNNs through the lens of kernels, such as the GP correspondence for DNN function outputs and NNGP kernel limits for feature map inner products at initialisation. We now describe the efforts that have been made to characterise and improve the behaviour of the NNGP kernel, particularly in terms of the impact of large *depth* in a wide DNN, where certain issues arise if one is not careful. This is a popular line of work that is known in the literature as *signal propagation* (Daniely et al., 2016; Poole et al., 2016; Schoenholz et al., 2017). We begin by presenting signal propagation in the context of MLPs (Equation (2.28)), before discussing extensions to other architectures.

In signal propagation, we are interested in studying the initialisation-time behaviour of the deterministic "kernel" limit of normalised feature inner products $\frac{1}{d_l}\langle h_l(\boldsymbol{x}), h_l(\boldsymbol{x}')\rangle$ at different layers l. As described in Section 2.2.1, these limits are shown formally to exist in Lee et al. (2018a); Matthews et al. (2018); Yang (2019b), and we denote such limits in terms of a layer-dependent function $q_l(\boldsymbol{x}, \boldsymbol{x}')$, in the sense that $\frac{1}{d_l}\langle h_l(\boldsymbol{x}), h_l(\boldsymbol{x}')\rangle \xrightarrow{a.s.} q_l(\boldsymbol{x}, \boldsymbol{x}')$ as $d_l \to \infty$.

We sometimes refer to $q_l(\boldsymbol{x}, \boldsymbol{x}')$ as a *covariance* function, because each "slice" of the pre-activations $g_{l+1}(\cdot)$ can be thought of as an independent zero-mean GP with covariance related to q_l: for the MLP this is $g_{l+1}(\cdot)_\alpha \sim \mathcal{GP}(0, \sigma_W^2 q_l(\cdot,\cdot) + \sigma_b^2)$ for each $\alpha \leq d_{l+1}$, as $d_1, d_2, \ldots, d_l \to \infty$. By expressing $\frac{1}{d_{l+1}}\langle h_{l+1}(\boldsymbol{x}), h_{l+1}(\boldsymbol{x}')\rangle$ in terms of $g_{l+1}(\cdot)$ and noting the above, it follows that we can write q_{l+1} recursively in terms of q_l. For the MLP (Equation (2.28)) with weights initialised according to Equation (2.29),

we have:

$$q_{l+1}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{f \sim \mathcal{GP}\left(0, \sigma_W^2 q_l(\cdot,\cdot) + \sigma_b^2\right)}[\psi(f(\boldsymbol{x}))\psi(f(\boldsymbol{x}'))] \tag{2.36}$$

with input layer $q_0(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sigma_W^2}{d}\boldsymbol{x}^\top \boldsymbol{x}' + \sigma_b^2$. We note that analytic formulas for Equation (2.36) writing q_{l+1} in terms of q_l exist for some activations, including the popular ReLU activation (Cho and Saul, 2009). The Neural Tangents library (Novak et al., 2020) in JAX (Bradbury et al., 2018) allows analytic computation of infinite-width NN kernels for such appropriate activation functions.

Through studying the evolution of Equation (2.36) as depth l increases, we can identify and ideally mitigate several degeneracies that occur for NNs at large depth, at least in terms of initialisation. One setting we must avoid is when the diagonal covariance values, $q_l(\boldsymbol{x}, \boldsymbol{x})$, rapidly grow or shrink with depth. This corresponds to pre-activations norms either exploding or vanishing, respectively, and can lead to numerical errors. While uncontrolled pre-activations norms are relatively easy to prevent, e.g. using "fan-in" initialisation, a second and arguably less straightforward issue is when the *correlation* function:

$$c_l(\boldsymbol{x}, \boldsymbol{x}') = \frac{q_l(\boldsymbol{x}, \boldsymbol{x}')}{\sqrt{q_l(\boldsymbol{x}, \boldsymbol{x}) q_l(\boldsymbol{x}', \boldsymbol{x}')}} \tag{2.37}$$

collapses to an input-independent constant, $0 \leq c^* \leq 1$, such that

$$\lim_l c_l(\boldsymbol{x}, \boldsymbol{x}') = c^*, \ \forall \boldsymbol{x}, \boldsymbol{x}', \tag{2.38}$$

and the DNN features lose all information corresponding to the geometry of input space, i.e. the relative distances in feature space do not reflect relative distances in input space.

As argued by Schoenholz et al. (2017); Martens et al. (2021), collapsed correlations lead to DNNs that either fail to train, or overfit to the training data; in both cases the DNN will fail to generalise due to its degenerate initialisation. Thus, avoiding a collapsed correlation function is essential for DNNs to be trainable. Martens et al. (2021) also further argue and provide empirical evidence that having well-behaved correlation and covariance functions is *sufficient* for a DNN to enjoy rapid training. The hope then is that by studying signal propagation, we can provide a set of rules and principles that aid DL practitioners in designing new, high-performing models and architectures.

Mitigating signal degeneration Despite the widespread assumption that depth is crucial for optimal performance in NNs (Simonyan and Zisserman, 2014; He et al., 2015), Poole et al. (2016); Schoenholz et al. (2017); Martens et al. (2021) use the recursive kernel behaviour (Equation (2.36)) to show that naively stacking layers (without architectural tools like skip connections and normalisation layers) will lead to degenerate correlation functions $c_l(\boldsymbol{x}, \boldsymbol{x}')$ as depth increases to infinity. These findings mirror the observation that deeper NNs tend to be harder to train (Glorot and Bengio, 2010; Nielsen, 2015). Xiao et al. (2020) use signal propagation theory, combined with NTK analysis (Section 2.2.3), to demonstrate a tension between trainability and generalisation in deep wide NNs. Lou et al. (2022) examine the role of signal propagation for understanding the feature alignment phenomenon that occurs during training (Baratin et al., 2021).

The rate of convergence to collapsed correlations is determined through the choice of activation function ψ, as well as the choice of weight and bias variances σ_W^2, σ_b^2. At worst, convergence to the

collapsed state c^* is exponentially fast in depth l, though this rate can be slowed to be sub-exponential in depth by using certain ψ-dependent choices of (σ_W^2, σ_b^2), in a line of work known as the *Edge of Chaos* (Schoenholz et al., 2017; Hayou et al., 2019).

Instead of considering the infinite depth limit, Martens et al. (2021) use the recursive kernel behaviour to design modifications, including transformations to the activation function ψ, aimed at controlling and improving signal propagation in a *fixed*, but potentially very deep, DNN architecture. The ensuing framework, called *Deep Kernel Shaping* (DKS), adapts to the specific architecture at hand to ensure non-degenerate signal propagation, and achieves impressive results training DNNs without skip connections and normalisation layers on ImageNet classification (Martens et al., 2021; Zhang et al., 2022).

The key insight formalised in DKS is that collapsed correlations stem from a DNN being "too" non-linear, and in order to control the *total* non-linearity in a deep architecture, each individual DNN layer's non-linearity (which is determined by ψ in an MLP) must be transformed to be more linear and "identity function like". Recently, Li et al. (2022) considered a joint width-and-depth, rather than a width-first-then-depth, limit and showed that instead of a deterministic recursion (Equation (2.36)), one obtains a stochastic differential equation that governs the behaviour of the NNGP kernel at deep layers, when using similarly transformed activation functions. Hayou (2022) study the infinite-depth limit for finite-width residual networks at initialisation, and find that the individual neuron activations evolve following a zero-drift diffusion process over depth.

A closely related line of work has studied the effect of various architectural tools like residual connections and normalisation layers, from a signal propagation perspective. Indeed, many works have noted the benefits of explicitly downscaled residual connections,

$$h_{l+1}(\boldsymbol{x}) = h_l(\boldsymbol{x}) + \lambda_l \psi(h_l(\boldsymbol{x})), \tag{2.39}$$

where $\lambda_l < 1$ is some downscaling factor and ψ is a non-linear layer (Hanin and Rolnick, 2018; Zhang et al., 2018; Arpit et al., 2019; Zhang et al., 2019a; De and Smith, 2020; Huang et al., 2020; Xu et al., 2020; Hayou et al., 2021a; Bachlechner et al., 2021; Martens et al., 2021; Touvron et al., 2021; Anagnostidis et al., 2022; Wang et al., 2022). In Hayou et al. (2021a), we show that certain scaled residual NNs, which we called *Stable ResNets*, have NNGP kernels that possess non-degenerate infinite-depth limits, both in terms of diagonal covariance values $q_l(\boldsymbol{x}, \boldsymbol{x})$ and also correlation functions $c_l(\boldsymbol{x}, \boldsymbol{x}')$.

One can think of a downscaled residual branch as another way of making a non-linear layer more "linear", much like a transformed activation function in an MLP. In fact, De and Smith (2020) show that an implicit effect of using residual branches that have normalisation layers is precisely to downweight the residual connection, which may partly explain the popularity of skip connections and normalisation layers as two staples in most modern DNN architectures (Xiong et al., 2020).

Signal propagation for different NN non-linear layers While in theory the kernel recursion used in signal propagation can be extended to arbitrary architectures that observe kernel limits, in the MLP case (Equation (2.36)) the dynamics have the appealing property that they can be reduced to two single-dimension systems (Poole et al., 2016; Martens et al., 2021), which is crucial for tractable analyses. This means that extending signal propagation analyses to other non-linear layers like self-attention or

convolutional layers may potentially be significantly harder than for MLPs. Having said that, we now discuss extensions of signal propagation theory to different architectures beyond MLPs.

We note that DKS (Martens et al., 2021) was actually introduced not for MLPs but more generally for CNNs, and the authors use the fact that an MLP can be viewed as a particular type of CNN with 1x1 filter size and feature height/width. For CNNs, a so-called "Delta-Orthogonal" initialisation (Balduzzi et al., 2017; Xiao et al., 2018; Martens et al., 2021), which zeros out certain parameters to effectively reduce the CNN to an MLP at initialisation, is necessary to prevent signal degradation at large depths, and Xiao et al. (2018) in particular use this to train remarkably deep CNNs of up to 10000 layers! A similarly motivated initialisation scheme is adopted by Zaidi et al. (2022) to enable compatibility of DKS with graph NNs (Scarselli et al., 2008).

For attention-based transformers that process sequence data, Dong et al. (2021) showed that naive stacking of attention layers without residual connections leads to a signal propagation phenomenon called *rank collapse*, where all elements of the sequence have perfectly correlated features at large depths. Anagnostidis et al. (2022) showed that rank collapse leads to vanishing gradients for certain parameters in the attention layers, thus hindering trainability, though downweighted residual connections, Equation (2.39), can prevent this degradation. In He et al. (2023), we consider the signal propagation properties of the self-attention layer directly, and derive modified versions of self-attention that enable deep transformers to be trained without architectural props like skip connections and normalisation layers, for the first time.

2.2.3 The Neural Tangent Kernel

So far, we have focused on the kernel properties of the *forward propagation* in wide DNNs at initialisation. This theory has highlighted particular issues that occur in DNNs are large depths, and shed light on how we can correct them to improve the performance of DNNs in practice. However, DNNs are typically trained through gradient-based methods, and so to fully characterise DNN training, we would also, ideally, like to understand the *backward propagation* of signal. Seminal work by Jacot et al. (2018) attempts to do this, by demonstrating kernel behaviour to the backwards pass too and allowing one to treat the entire DNN training process as *kernel learning*. In doing so, the so-called *Neural Tangent Kernel* (NTK) appears as a distinct, but closely related, kernel to the NNGP kernel we have discussed so far.

The NTK naturally emerges by considering the gradient descent dynamics of a parametric model $f(\cdot, \boldsymbol{\theta})$ on some loss function $\mathcal{L}(\boldsymbol{\theta})$ that depends on the model's predictions, $f(\mathbf{X}, \boldsymbol{\theta})$, on some training set $\mathbf{X} \in \mathbb{R}^{n \times d}$. If we consider continuous time gradient flow, with learning rate η, then our parameters evolve as:[6]

$$\dot{\boldsymbol{\theta}}^t = -\eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^t) \tag{2.40}$$

$$= -\eta \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}^t)^\top \nabla_{f(\mathbf{X}, \boldsymbol{\theta}^t)} \mathcal{L} \tag{2.41}$$

by the chain rule. In turn, we can plug in the parameter dynamics to study how the model prediction

[6] In this chapter, for NNs we will use superscripts to index time, and subscripts to index layers

$f(\boldsymbol{x}, \boldsymbol{\theta}^t)$ at some test input $\boldsymbol{x}$ evolves in time:

$$\dot{f}(\boldsymbol{x}, \boldsymbol{\theta}^t) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^t)\dot{\boldsymbol{\theta}}^t \tag{2.42}$$

$$= -\eta \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^t) \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}^t)^\top \nabla_{f(\mathbf{X}, \boldsymbol{\theta}^t)} \mathcal{L} \tag{2.43}$$

$$= -\eta \hat{\Theta}_t(\boldsymbol{x}, \mathbf{X}) \nabla_{f(\mathbf{X}, \boldsymbol{\theta}^t)} \mathcal{L} \tag{2.44}$$

where

$$\hat{\Theta}_t(\boldsymbol{x}, \mathbf{X}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^t) \nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}^t)^\top \in \mathbb{R}^{1 \times n}$$

can be seen to be computing a strength of "similarity" between input $\boldsymbol{x}$ and the training data $\mathbf{X}$, and updating $f(\boldsymbol{x})$ based on training data function gradients $\nabla_{f(\mathbf{X}, \boldsymbol{\theta}^t)} \mathcal{L} \in \mathbb{R}^{n \times c}$, weighted by similarity. We shall refer to $\hat{\Theta}_t$ as the *empirical NTK*, as it can be seen to be defining a kernel with time-dependent feature map $\phi_t(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^t)$. Note that the empirical NTK depends on the value of $\boldsymbol{\theta}^t$, which in turn depends on the random initialisation $\boldsymbol{\theta}^0$. Hence, a priori we have that $\hat{\Theta}_t$ is a random kernel, that will vary during training.

NTK convergence For a feedforward architecture (Equation (2.24)), of depth L with trainable parameters in each layer $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{L+1}\}$, we can decompose the empirical NTK into contributions from each layer:

$$\hat{\Theta}_t(\boldsymbol{x}, \boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^t), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}', \boldsymbol{\theta}^t) \rangle \tag{2.45}$$

$$= \sum_{l=1}^{L+1} \langle \nabla_{\boldsymbol{\theta}_l} f(\boldsymbol{x}, \boldsymbol{\theta}^t), \nabla_{\boldsymbol{\theta}_l} f(\boldsymbol{x}', \boldsymbol{\theta}^t) \rangle \tag{2.46}$$

If we inspect the contribution to the empirical NTK from the last layer parameters $\boldsymbol{\theta}_{L+1}$,

$$\langle \nabla_{\boldsymbol{\theta}_{L+1}} f(\boldsymbol{x}, \boldsymbol{\theta}^t), \nabla_{\boldsymbol{\theta}_{L+1}} f(\boldsymbol{x}', \boldsymbol{\theta}^t) \rangle,$$

we find that it is precisely the inner product of last layer features, $\langle h_L(\boldsymbol{x}), h_L(\boldsymbol{x}') \rangle$, which we know converges to the NNGP kernel (under rescaling) at initialisation, from Corollary 2.4. This can be easily seen by noting that $f(\boldsymbol{x}, \boldsymbol{\theta}) = h_L(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}$, so that $\nabla_{\boldsymbol{\theta}_{L+1}} f(\boldsymbol{x}, \boldsymbol{\theta}^t) = h_L(\boldsymbol{x})$. Thus, we know (at a minimum) that the contribution to the empirical NTK from the last layer parameters converges with width at initialisation (if properly scaled), and that this limit is precisely the NNGP kernel.[7]

Remarkably, Jacot et al. (2018) proved that in the large width limit, using a so-called *NTK parameterisation*, an MLP's *entire* empirical NTK not only converges to a deterministic kernel, Θ, at initialisation, but also stays constant *throughout* training, under regularity assumptions on the loss function. We state this result in Theorem 2.7:

Theorem 2.7 (NTK regime training dynamics (Jacot et al., 2018), informal)**.** *Under the NTK parameterisation and regularity conditiions on the loss/activation functions, the empirical NTK, $\hat{\Theta}_t$, converges to a deterministic kernel, Θ, at initialisation $t = 0$, and remains constant at all times $t < \infty$.*

The limiting kernel, Θ, was given the name the name *Neural Tangent Kernel* (NTK) by Jacot et al. (2018), and we will refer to the setting where the NTK stays constant as the *NTK or kernel regime*. We

[7] This is what we refer to when we say that the NTK and the NNGP are closely related kernels.

note that the NTK regime and its implications also appeared in concurrent work (Li and Liang, 2018; Du et al., 2018; Allen-Zhu et al., 2019) that demonstrated the convergence and generalisation properties of overparameterised NNs. Jacot et al. (2018) showed Theorem 2.7 for MLPs in the limit $d_1, \ldots, d_L \to \infty$ sequentially, and Arora et al. (2019a) derived the NTK extension for CNNs in the large channel limit. These results have been generalised to a joint width limit and all standard architectures in the Tensor Program framework by Yang (2020a); Yang and Littwin (2021).

Different NN parameterisations To understand better the differences between the NTK setting and the standard parameterisation (SP) (Equation (2.28)), as well as to feature learning regimes later on in Section 2.3, we now introduce different ways to parameterise NNs, and their different scalings with width. The NTK parameterisation (NTP), introduced by Jacot et al. (2018), can be thought of as one such scaling that preserves the same forward computations as SP, whilst rescaling the gradient computations in NNs to ensure that the empirical NTK,

$$\hat{\Theta}_t(\boldsymbol{x}, \boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^t), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}', \boldsymbol{\theta}^t) \rangle, \tag{2.47}$$

doesn't blow up as the width increases, unlike in the SP case (Equation (2.28)).

For an MLP with constant width m across layers, we introduce a family of *abc-parameterisations* (Yang and Hu, 2020), of which both the standard and NTK parameterisations are examples:[8]

$$g_l(\boldsymbol{x}) = \frac{1}{m^{a_l}} h_{l-1}(\boldsymbol{x})^\top W_l, \tag{2.48}$$

$$h_l(\boldsymbol{x}) = \psi(g_l(\boldsymbol{x})) \;\; \forall 1 \leq l \leq L. \tag{2.49}$$

with initialisations:

$$W_l^0 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{\sigma_W^2}{m^{2b_l}}), \tag{2.50}$$

and per-layer learning rates ηm^{-c_l}. For now, we suppose that a_l, b_l, c_l do not vary between layers and take values a, b, c respectively. Then, in an abc-parameterisation, "a" rescales the (pre-)activations, "b" sets the initialisation scale of the weight matrices, and "c" controls how the learning rate scales with width.

To see the effect of the abc rescaling on the NTK, consider the contribution to the (empirical) NTK from the weight parameters W_l in some layer l: $\langle \nabla_{W_l} f(\boldsymbol{x}, \boldsymbol{\theta}), \nabla_{W_l} f(\boldsymbol{x}', \boldsymbol{\theta}) \rangle$. From Equation (2.48), it is relatively straightforward to compute that:

$$\nabla_{W_l} f(\boldsymbol{x}, \boldsymbol{\theta}) = m^{-a} dg_l(\boldsymbol{x}) h_{l-1}(\boldsymbol{x})^\top$$

where $dg_l(\boldsymbol{x}) = \nabla_{g_l} f(\boldsymbol{x}, \boldsymbol{\theta})$ are *pre-activation derivatives*. Thus, the contribution to the NTK from W_l:

$$\langle \nabla_{W_l} f(\boldsymbol{x}, \boldsymbol{\theta}), \nabla_{W_l} f(\boldsymbol{x}', \boldsymbol{\theta}) \rangle = m^{-2a} \langle h_{l-1}(\boldsymbol{x}), h_{l-1}(\boldsymbol{x}') \rangle \langle dg_l(\boldsymbol{x}), dg_l(\boldsymbol{x}') \rangle$$

is scaled down by m^{-2a}, and is independent of b and c. We can now state how standard and NTK parameterisations fit into the abc framework:

[8] We omit bias parameters in our abc-parameterisation for clarity.

- The **standard parameterisation** corresponds to setting $a = 0, b = 0.5$, which means there is no downscaling of the NTK. As a result, it's been shown that the learning rate needs to be downscaled, with $c = 1$, to counteract an exploding NTK and gradients at large widths (Park et al., 2019a; Karakida et al., 2019; Sohl-Dickstein et al., 2020; Yang and Hu, 2020).
- The **NTK parameterisation**, on the other hand, is an abc-parameterisation that sets $a = 0.5, b = 0$. This has the effect of downscaling the NTK by a factor of m so that a constant learning rate $c = 0$ can be used, even at large widths.

Note that the NTP and SP lead to the same forward computations, so that the NNGP limit of $\frac{1}{m}\langle h_l(\boldsymbol{x}), h_l(\boldsymbol{x}')\rangle$ is the same in the NTK parameterisation as introduced previously in Section 2.2.1.

Implications of the NTK regime One way to interpret the abc-scaling in the NTK parameterisation is that it scales down the gradients at each parameter by m^{2a}, so that in gradient-based optimisation the parameters $\boldsymbol{\theta}$ stay close to their initialisations $\boldsymbol{\theta}^0$. Lee et al. (2019) showed that an implication of the NTK regime is that the difference between a DNN $f(\boldsymbol{x}, \boldsymbol{\theta})$ and its linearisation about $\boldsymbol{\theta}^0$, $f^{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta})$, vanishes as width increases to ∞, where:

$$f^{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}) = f(\boldsymbol{x}, \boldsymbol{\theta}^0) + \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^0)\Delta\boldsymbol{\theta}, \tag{2.51}$$

with $\Delta\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\theta}^0$ and $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^0)$ is the *fixed* Jacobian feature map of the empirical NTK at initialisation, $\hat{\Theta}_0$.

From this, we see that the NTK limit is an extremely powerful result, as it says that the previously non-convex and intractable NN parameter landscapes become linear and well-behaved in the infinite-width limit. Thus, we can apply all of our knowledge concerning random features/kernel methods/GPs (e.g. Sections 2.1.1, 2.1.3 and 2.1.4) to study the convergences and training dynamics of DNNs. For example, with gradient flow (Equation (2.40)) on the least squares loss given training data $\{\mathbf{X}, Y\}$, it can be seen that we will obtain the trained solution:

$$f^{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_\infty) = \hat{\Theta}_0(\boldsymbol{x}, \mathbf{X})\hat{\Theta}_0(\mathbf{X}, \mathbf{X})^{-1}Y + f(\boldsymbol{x}, \boldsymbol{\theta}^0) - \hat{\Theta}_0(\boldsymbol{x}, \mathbf{X})\hat{\Theta}_0(\mathbf{X}, \mathbf{X})^{-1}f(\mathbf{X}, \boldsymbol{\theta}^0), \tag{2.52}$$

assuming the invertibility of $\Theta(\mathbf{X}, \mathbf{X})$, which closely approximates a trained wide DNN under NTK parameterisation (Lee et al., 2019).

Because: i) $f(\boldsymbol{x}, \boldsymbol{\theta}^0)$ converges to a GP draw, $\mathcal{GP}(0, \mathcal{K})$ with NNGP $\mathcal{K}$ (Proposition 2.5), ii) the empirical NTK converges to the limiting NTK in the infinite width, as well as iii) the fact that Equation (2.52) is linear in $f(\cdot, \boldsymbol{\theta}^0)$, we obtain an analytic GP distribution for f^{lin} (hence f) that involves both the NNGP, $\mathcal{K}$, and the NTK, Θ, in the large width limit. Lee et al. (2019) showed that this GP distribution in Equation (2.52) has the form:

$$f \sim \mathcal{GP}(\mu, \Sigma)$$

with mean:

$$\mu(\boldsymbol{x}) = \Theta(\boldsymbol{x}, \mathbf{X})\Theta(\mathbf{X}, \mathbf{X})^{-1}Y,$$

which is simply kernel regression with the NTK, and covariance:

$$\Sigma(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}_{\boldsymbol{x},\boldsymbol{x}'} + \Theta_{\boldsymbol{x},\mathbf{X}}\Theta^{-1}_{\mathbf{X},\mathbf{X}}\mathcal{K}_{\mathbf{X},\mathbf{X}}\Theta^{-1}_{\mathbf{X},\mathbf{X}}\Theta_{\mathbf{X},\boldsymbol{x}'} - \Theta_{\boldsymbol{x},\mathbf{X}}\Theta^{-1}_{\mathbf{X},\mathbf{X}}\mathcal{K}_{\mathbf{X},\boldsymbol{x}'} - \mathcal{K}_{\boldsymbol{x},\mathbf{X}}\Theta^{-1}_{\mathbf{X},\mathbf{X}}\mathcal{K}_{\mathbf{X},\boldsymbol{x}}$$

using the subscript notation $\mathcal{K}_{\boldsymbol{x},\boldsymbol{x}'} = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$ for clarity. In Chapter 3, we study this limiting distribution and its relation to a GP posterior.

Empirical performance of NTK methods Given the dramatic theoretical implications that the NTK promises, it is natural to ask how the empirical performance of infinite-width NTK-based methods (e.g. kernel ridge regression with NTK kernel) compares to their equivalent finite-width trained NNs. On small tabular datasets, it has been observed that NTKs outperform finite-width NNs (Arora et al., 2019b). However, the general consensus in the community is that NTKs underperform finite-width NNs on tasks that NNs particularly excel at, e.g. CIFAR-10 (Krizhevsky et al., 2009) classification with CNNs (Arora et al., 2019a; Shankar et al., 2020; Lee et al., 2020),[9] though this is somewhat exacerbated by the fact that kernel methods are much less scalable than NNs, which enjoy efficient boosts in performance from tricks like data-augmentation. Fort et al. (2020) observed a related phenomenon where the empirical NTK, $\hat{\Theta}_t$, of NNs trained in practice with SGD performs better on CIFAR-10/100 image classification at *later* times t.

While these findings may not be too surprising given that finite-width CNNs are designed to excel on image data, they do cast some doubt on the capabilities of the NTK theory to explain the behaviour of practical NNs, which we explore further in the next section.

2.3 Feature Learning in Neural Networks

In the previous section we saw that the NTK offers a valid mathematical regime to precisely characterise the *entire* training trajectory of a wide DNN of any architecture or depth. In this limit, the NN's behaviour reduces to kernel learning dictated by two closely related kernels, the NNGP and the NTK, which govern the NN's output at initialisation and training dynamics respectively. Crucial to the NTK regime's analytic tractability is that the NTK, and hence NNGP, remain constant kernels throughout training. Recalling that the NNGP kernel was derived as the limit of normalised feature inner products $\frac{1}{m}\langle h_L(\boldsymbol{x}), h_L(\boldsymbol{x})\rangle$ in large width m, we also have that the so-called last-layer *feature kernel* $k_L(\boldsymbol{x}, \boldsymbol{x}')$:

$$k_L(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{m}\langle h_L(\boldsymbol{x}), h_L(\boldsymbol{x}')\rangle \tag{2.53}$$

not only converges to a deterministic limit (the NNGP) but also is constant during training in the infinite-width NTK regime.

This lack of feature kernel evolution, combined with the relatively poor empirical performance of infinite-width DNN kernels, has been a major source of criticism for the NTK regime as a theory to describe the success of DNNs in practice, leading some to call the NTK "lazy learning" (Chizat et al., 2019) due to the fact that the model can be described by its linearisation at initialisation, from which the

[9] We note that Shankar et al. (2020); Lee et al. (2020) in fact observed that in many cases the NNGP kernel (which corresponds to only training last layer parameters in the infinite width limit) outperforms the NTK kernel in terms of predictive accuracy.

parameters do not move far away. Empirical findings support this criticism (Fort et al., 2020; Baratin et al., 2021), demonstrating that the NTK changes significantly during training on practical networks, particularly during the initial phases of training. We note that the classical methods we present in Section 2.1 (kernels, GPs and random features), can be viewed as lazy learning methods, as the kernels involved are by construction fixed throughout the learning process.

The alternative to the lazy or kernel learning regime is often colloquially referred to as *feature learning* or *representation learning*,[10] and though it promises to be a more relevant theory for DNNs than the kernel regime, one could argue there isn't a precise definition for feature learning in the literature. In this thesis, we will refer to feature learning as being equivalent to feature kernel evolution i.e. when the feature kernel (Equation (2.53)) changes during training, similar to considerations by (Aitchison, 2020; Yang and Hu, 2020). In the infinite-width limit, Yang and Hu (2020) show that this definition is equivalent to other definitions that consider the updates to individual elements of the features $h_L(\boldsymbol{x}) \in \mathbb{R}^m$.

A number of works have derived extensions to the kernel regime to enable compatability of wide DNNs with feature learning. These include using larger learning rates (Lewkowycz et al., 2020), higher-order approximations to the training dynamics (Huang and Yau, 2020), joint depth-and-width limits (Hanin and Nica, 2019; Li et al., 2021, 2022) and finite-width corrections (Dyer and Gur-Ari, 2019; Roberts et al., 2021). These extensions mostly focus on demonstrating the possibility of feature learning and leaving the kernel regime in DNNs, as opposed to providing tractable training dynamics.

One the other hand, a separate line of work, known in the literature as the *mean-field regime*, has focused on a distinct non-kernel limit for wide NNs that is compatible with feature learning. The idea in the mean-field limit is to study the infinite-width limit of the *empirical distribution over neurons* in a hidden-layer and how this limiting distribution's evolution during training can be tracked as the solution of a non-trivial partial differential equation. Owing to these more complicated training dynamics, the mean-field regime has predominantly been studied in the shallow setting, with most works focusing on the single-hidden layer case (Chizat and Bach, 2018; Sirignano and Spiliopoulos, 2018; Mei et al., 2018; Rotskoff and Vanden-Eijnden, 2018; De Bortoli et al., 2020). Having said that, several extensions to the mean-field limit have been proposed for deeper NNs in certain settings (Araújo et al., 2019; Fang et al., 2021; Yang and Hu, 2020; Pham and Nguyen, 2021). In Section 2.3.2 we outline the extension of Yang and Hu (2020), which is compatible with feature learning in any architecture/depth and introduces the abc-parameterisations we described in Section 2.2.3 to provide insight on the connections and differences between kernel and feature learning regimes.

Example (pre-training with linear evaluation) To intuitively see why a lack of feature learning (as predicted by the NTK regime) is an issue in describing the success of DNNs, consider a setting where we pre-train our DNN on a large pre-training dataset before throwing away the last layer, $\boldsymbol{\theta}_{L+1}$, and adding a new linear layer, $\boldsymbol{\theta}^{\text{new}}$, on top of fixed pre-trained features $h_L(\cdot)$ for a downstream task. That is, we train $\boldsymbol{\theta}^{\text{new}}$ in our new model $f^{\text{new}}(\boldsymbol{x}) = h_L(\boldsymbol{x})^\top \boldsymbol{\theta}^{\text{new}}$ on the downstream task, while keeping $h_L(\boldsymbol{x})$ *fixed* from before. This is a popular setting for pre-training e.g. self-supervised pre-training on

[10] The opposite of kernel/lazy learning, which we call feature/representation learning, has also been referred to as "active learning" (Chizat et al., 2019) or the "rich regime" (Woodworth et al., 2020) in the literature.

ImageNet and downstream classification on CIFAR-10, and has been observed to significantly improve performance on downstream tasks compared to using randomly initialised features without pre-training. Oftentimes, we will have that the downstream loss function is simple and convex in $\boldsymbol{\theta}^{\text{new}}$ e.g. when using least squares loss or cross entropy. In these cases we will have unique solutions for f^{new} that are *exactly defined* by the feature kernel (c.f. Appendix A in Chapter 4). As a result, without feature kernel evolution in pre-training, it is not possible to explain why the initial pre-training phase leads to a dramatic improvement in performance on downstream settings. (Devlin et al., 2018; Brown et al., 2020; Caron et al., 2020; Dosovitskiy et al., 2020)

2.3.1 Microscropic feature learning in the NTK regime

At first it may appear surprising that the NTK regime leads to any non-trivial behaviour at all (besides simply learning a linear model on top of final-layer features $h_L(\boldsymbol{x})$), given that the feature kernel is not changing during training. To reconcile these two seemingly contradictory observations, we highlight that there *is* actually feature learning occuring in the NTK regime, but that this feature learning occurs at a "microscopic" scale, that vanishes as width increases to infinity.

For a randomly initialised NN in the NTK parameterisation with output dimension 1 (for simplicity), we have: $f(\boldsymbol{x}, \boldsymbol{\theta}^0) = \frac{1}{\sqrt{m}} h_L^0(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}^0$, where entries of both $h_L^0(\boldsymbol{x}) \in \mathbb{R}^m$ and $\boldsymbol{\theta}_{L+1}^0 \in \mathbb{R}^m$ are $\Theta(1)$ at initialisation. Crucially, because the weights are randomly initialised, $\boldsymbol{\theta}_{L+1}^0$ and $h_L^0(\boldsymbol{x})$ are independent at initialisation, and so

$$f(\boldsymbol{x}, \boldsymbol{\theta}^0) = \frac{1}{\sqrt{m}} h_L^0(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}^0 = \frac{1}{\sqrt{m}} \sum_{\alpha=1}^{m} h_L^0(\boldsymbol{x})_\alpha (\boldsymbol{\theta}_{L+1}^0)_\alpha \tag{2.54}$$

observes central limit theorem behaviour and is correctly scaled (by $\frac{1}{\sqrt{m}}$) to give an $\Theta(1)$ GP output f.

If we now suppose that we take a gradient step for all parameters from $\boldsymbol{\theta}^0$ to $\boldsymbol{\theta}^1$, but keep last layer parameters $\boldsymbol{\theta}_{L+1}$ fixed at $\boldsymbol{\theta}_{L+1}^0$ then we will have updated features $h_L^1(\boldsymbol{x}) = h_L^0(\boldsymbol{x}) + \Delta h_L(\boldsymbol{x})$ with updates $\Delta h_L(\boldsymbol{x})$, and an updated forward pass:

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{\theta}^1) &= \frac{1}{\sqrt{m}} h_L^1(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}^0 && (2.55) \\ &= \frac{1}{\sqrt{m}} \big(h_L^0(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}^0 + \Delta h_L(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}^0 \big) && (2.56) \\ &= f(\boldsymbol{x}, \boldsymbol{\theta}^0) + \frac{1}{\sqrt{m}} \Delta h_L(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}^0 && (2.57) \end{aligned}$$

This time around, $\Delta h_L(\boldsymbol{x})$ will be strongly correlated with $\boldsymbol{\theta}_{L+1}^0$, which reflects the fact that the parameters in $h_L(\boldsymbol{x})$ were updated with gradient descent, and $\boldsymbol{\theta}_{L+1}^0$ appears in any such gradient. As a result, $\frac{1}{\sqrt{m}} \Delta h_L(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}^0$ observes a law of large numbers behaviour, and it is thus necessary for $\Delta h_L(\boldsymbol{x})$ to have entries that are $\Theta(\frac{1}{\sqrt{m}})$ in order to have non-trivial and non-exploding function updates $f(\boldsymbol{x}, \boldsymbol{\theta}^1) - f(\boldsymbol{x}, \boldsymbol{\theta}^0)$ with large m.

In this case, the new feature kernel after one gradient step:

$$\begin{aligned} \frac{1}{m} \langle h_L^1(\boldsymbol{x}), h_L^1(\boldsymbol{x}) \rangle &= \frac{1}{m} \langle h_L^0(\boldsymbol{x}) + \Delta h_L(\boldsymbol{x}), h_L^0(\boldsymbol{x}) + \Delta h_L(\boldsymbol{x}) \rangle && (2.58) \\ &= \frac{1}{m} \langle h_L^0(\boldsymbol{x}), h_L^0(\boldsymbol{x}) \rangle + O(\frac{1}{\sqrt{m}}) && (2.59) \end{aligned}$$

which converges to the same initialisation-time NNGP limit as $\frac{1}{m}\langle h_L^0(\boldsymbol{x}), h_L^0(\boldsymbol{x})\rangle$ for $m \to \infty$, i.e. vanishing feature learning.

2.3.2 Feature learning with infinite-width DNNs

Given that the NTK regime was the first theoretical result to characterise the training behaviour of deep NNs, and in particular is specific to the infinite-width limit of NNs, a common conclusion in the community after 2018 was that feature learning is a property specific to trained finite-width NNs. Such a conclusion calls into question the merit of studying infinite-width NNs, which have important tractability advantages compared to their finite-width counterparts, as a vehicle to understand NNs. This mirrors the famous quote from MacKay (1998) regarding implications of the GP correspondence of shallow wide NNs for Bayesian NNs (Neal, 1996): "Have we thrown the baby out with the bathwater?". As we will now see, recent works have demonstrated that this is not necessarily the case.

Going back to the previous example, we considered an NN in NTK parameterisation, $f(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{\sqrt{m}} h_L(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}$, that outputs an inner product of last layer features $h_L(\boldsymbol{x})$ and linear parameters $\boldsymbol{\theta}_{L+1}$. With this setup, we saw how feature learning in the large-width NTK regime is not possible due to the highly correlated interactions between initialised last-layer parameters $\boldsymbol{\theta}_{L+1}^0$ and feature updates $\Delta h_L(\boldsymbol{x})$. $\boldsymbol{\theta}_{L+1}^0$ has $\Theta(1)$ elements, which forces the learnt features updates $\Delta h_L(\boldsymbol{x})$ to have vanishing $O(\frac{1}{\sqrt{m}})$ elements in order to avoid an exploding function update. Similar conclusions can be made for a wide NN in the standard parameterisation, $f(\boldsymbol{x}, \boldsymbol{\theta}) = h_L(\boldsymbol{x})^\top \boldsymbol{\theta}_{L+1}$, with initial last-layer parameters $\boldsymbol{\theta}_{L+1}^0$ having $\Theta(\frac{1}{\sqrt{m}})$ elements.

These insights shed light on how we can achieve well-behaved feature learning in infinite-width NNs: by reducing the *scale of initialisation* of the last-layer weights $\boldsymbol{\theta}_{L+1}^0$, with an appropriately scaled learning rate. For single hidden layer NNs with $L = 1$, this is akin to the aforementioned *mean-field regime* (Chizat and Bach, 2018; Sirignano and Spiliopoulos, 2018; Mei et al., 2018; Rotskoff and Vanden-Eijnden, 2018). For an arbitrary depth L, Yang and Hu (2020) generalise the mean-field regime to allow feature learning in infinite-width NNs. The key idea is to use a new abc-parameterisation (Equation (2.48)), the so-called **maximal update** or μ-parameterisation, described below, which the authors show allows feature learning even in the infinite width using the Tensor Programs framework. Intuitively, the idea is that in order to encourage feature learning, we have to rebalance the size of parameters and updates in different layers, so that the hidden layers receive larger updates relative to the last layer.

Recall that for an abc-parameterisation notation with width m, layer l has preactivations g_l that are scaled as $g_l(\boldsymbol{x}) = \frac{1}{m^{a_l}} h_{l-1}(\boldsymbol{x})^\top W_l$, with initial weights $W_l^0 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{\sigma_W^2}{m^{2b_l}})$ and learning rate $O(\frac{1}{m^{c_l}})$. Then, the μ-parameterisation (Yang and Hu, 2020) is as follows:

1. The **output layer** has downscaled parameters $W_{L+1}^0 \sim \mathcal{N}(0, \frac{\sigma_W^2}{m^2})$, and a scaled learning rate that is $O(\frac{1}{m})$ with width m, so that $b_{L+1} = 1$ and $c_{L+1} = 1$. One implication of this is that feature learning DNNs in the infinite-width provably have output function that is identically zero at initialisation, rather than a GP like in the standard parameterisation.

2. The **hidden layers** are unchanged from the standard parameterisation, i.e. $W_l^0 \sim \mathcal{N}(0, \frac{\sigma_W^2}{m})$ for $1 < l < L+1$, so $b_l = 0.5$. An $O(1)$ learning rate is used, which is larger than standard in order to

counteract the downscaled last-layer, W^0_{L+1}, and promote feature learning, so $c_l = 0$.

3. (Optional) The **input layer** has upscaled learning rate $O(m^1)$, so that $c_1 = -1$. Like standard parameterisation, we keep $W^0_1 \sim \mathcal{N}(0, \frac{\sigma_W^2}{d})$ for fixed input-dimension d, so that $b_1 = 0$. This is optional in the sense that feature learning still occurs (i.e. the last layer feature kernel still evolves) even if the input layer doesn't have upscaled learning rate.

4. All layers have no explicit preactivation scaling, so $a_l = 0 \; \forall l$. This makes use of a degree of freedom in the abc-parameterisation, and so the version of μ parameterisation presented here is the version in Yang et al. (2022a), as opposed to the original version (Yang and Hu, 2020).

Implications of feature learning in infinite-width NNs The discovery of the possibility for feature learning in wide deep NNs with μ-parameterisation is significant in many ways. To start with, it resolves the apparent contradiction that deep NNs lose the key property of feature learning as they become overparameterised. This is one of the implications of the NTK theory, and is at odds with the general trends in DL of increasing model scale as one way to improve performance (Krizhevsky et al., 2012; Zagoruyko and Komodakis, 2016b; Huang et al., 2019; Kaplan et al., 2020; Yang et al., 2022a). To get by this issue, Yang and Hu (2020) introduce abc-parameterisations to show that while the NTK regime is one valid mathematical limit for infinite-width NNs, an entire hyperplane of valid limits exist, some of which (like the μ-parameterisation) will lead to feature learning. This opens the door to a new and deeper understanding of the unique properties of DNNs, in both finite and infinite-width, and how they relate to and differ from more classical methods like kernels and GPs.

Moreover, in any infinite-width limit for NNs, be it feature or kernel learning, the training dynamics of the model can always be simplified to be viewed in terms of (potentially data-dependent) kernels that are induced by inner products of some features ϕ. In the feature learning limit, we consider the last-layer feature kernel $k_L(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{m}\langle h^t_L(\boldsymbol{x}), h^t_L(\boldsymbol{x}')\rangle$ that evolves during training to "acquire features", and in the kernel learning regime, we consider the empirical NTK $\hat{\Theta}_t(\boldsymbol{x}, \boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}^t), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}', \boldsymbol{\theta}^t)\rangle$ which has a static limit. Likewise, in signal propagation we consider the evolution in depth of hidden-layer feature kernels at initialisation $k_l(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{m}\langle h^0_l(\boldsymbol{x}), h^0_l(\boldsymbol{x}')\rangle$. In all possible limits, the training evolution of the NN can be tracked by tracking such kernels, which evolve according to a set of rules, as detailed in e.g. Yang and Hu (2020), *without ever needing to instantiate or consider the NN's parameter space.* Similar insights can be taken from other large-width feature learning limits (Bordelon and Pehlevan, 2022). From this shared property of kernel and feature learning regimes, it is thus somewhat natural to think of NNs in terms of (possibly learnt) kernels, even at finite widths i.e. one can think of the training process in terms of the data-dependent evolution (or lack thereof) of the feature kernel. All the subsequent chapters in this thesis take inspiration from this viewpoint. For instance, in Chapter 4, we study the role of the feature kernel for knowledge distillation in NNs.

Finally, there is hope that feature learning in wide NNs may lead to new theories of DL that can offer insights into, and ultimately improve, the training dynamics of large-scale NNs in practice. Yang et al. (2022a) provide an example of such a practical insight, by leveraging the μ-parameterisation to allow more efficient hyperparameter tuning in large models via transfer from smaller models. We note,

however, that there is much that is still unknown regarding feature learning: these feature learning limits for wide NNs have merely been shown to exist, and are non-convex thus significantly less tractable than the NTK regime. Moreover, they have training dynamics that are more expensive to compute than the kernel regime (which is already cubic in dataset size), which further limits their utility and our current understanding.[11] This presents opportunity and challenge alike for the community.

2.4 Deep Learning Techniques through the lens of Kernel vs Feature Learning

As discussed in Chapter 1, NNs have lead to huge advances in the way the we can process and make predictions from large-scale complex data. A key reason for this is the fact that NNs are capable of benefiting from many more training techniques and setups than other ML methods. Having introduced kernel and feature learning as two alternatives whose contrast provides a new lens for the community to view NNs through, in this section we now summarise the techniques: ensembling, knowledge distillation, and self-supervised learning, that we study with this lens in the remaining chapters of this thesis.

2.4.1 Ensemble learning

Ensemble learning is an ML technique that combines multiple models to make improved predictions compared to any individual model. The goal of ensembling is to improve performance by aggregating the capabilities of multiple models. Unlike the other techniques we will consider in this section, ensembling is popularly used both for NNs (Hansen and Salamon, 1990; Krogh and Vedelsby, 1994; Perrone and Cooper, 1992; Zhou et al., 2002; Lee et al., 2015; Lakshminarayanan et al., 2017; Jumper et al., 2021) and also more generally in ML (Breiman, 1996; Dietterich, 2000; Breiman, 2001; Opitz and Maclin, 1999; Polikar, 2006; Rokach, 2010).

There are many ways in which one can form an ensemble of models. In our context, we will consider a *baselearner* model $f(\cdot, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, and an ensemble will consist of a set of multiple learnt baselearners with parameters $\{\hat{\boldsymbol{\theta}}_k\}_{k=1}^K$. Given some observation model $p(y|f(\boldsymbol{x}))$ e.g. a Gaussian model $y \sim \mathcal{N}(f(\boldsymbol{x}), \sigma^2)$, we can use our ensembled parameters to generate an empirical distribution, $\hat{p}(\cdot|\boldsymbol{x})$, of predictions at any test input $\boldsymbol{x}$:

$$\hat{p}(\cdot|\boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^{K} p(\cdot|f(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_k)) \tag{2.60}$$

This distribution of predictions, Equation (2.60), can then be used both to make point predictions, e.g. using the predictive mean of $\hat{p}(\cdot|\boldsymbol{x})$, or obtain predictive uncertainties, e.g. using the predictive variance of $\hat{p}(\cdot|\boldsymbol{x})$ (for regression tasks).

Clearly, it is possible for the empirical distribution $\hat{p}(\cdot|\boldsymbol{x})$ to give K identical observation models, when all baselearners take the same parameter value. However, this ensemble wouldn't be particularly

[11] Solving for the feature learning training dynamics exactly in (Yang and Hu, 2020) is (super-)exponential in training time, and (Bordelon and Pehlevan, 2022) provide a monte-carlo based approximation that is cubic in *both* training time and dataset size. This implies that using wide NN feature learning regimes is unlikely to be directly practical, but may be useful in guiding finite-width NNs. (Yang et al., 2022b) show a form of projected gradient descent that can reduce the exponential complexity in training time to a quadratic dependency.

useful as one would obtain the same predictions with just a single trained model. Thus, it is necessary for there to be *diversity* in the predictions among different baselearners, in order to fully utilise an ensemble. In convex models like linear models or random features, this means that some form of baselearner diversification is needed, such as baselearner specific regularisation (Hoffman and Ribak, 1991; Matthews et al., 2017; Osband et al., 2018) or "bagging" i.e. dataset resampling (Breiman, 1996).

In NNs, it turns out simply independent initialisation and mini-batching for gradient computation in the baselearners is sufficient in order to promote useful diversity: an ensemble of NNs provides significant boosts to both the predictive accuracy (Lee et al., 2015; Szegedy et al., 2015; Allen-Zhu and Li, 2020) and uncertainty (Lakshminarayanan et al., 2017) compared to a single baselearner. This, combined with the simple and parallelisable nature of ensemble training, means that ensembling is a very successful technique in DL, and in particular constitutes a state-of-the-art approach for obtaining calibrated predictive uncertainties with NNs (Ovadia et al., 2019; Gustafsson et al., 2020).

There are various theories for explaining the success of ensembling for predictive accuracy and uncertainty in DNNs. For improved accuracy, a popular non-DNN specific hypothesis is that ensembling helps to *reduce the variance* of a single model (Dietterich, 2000; Mehta et al., 2019; Adlam and Pennington, 2020). In the case of NNs, this variance appears from random initialisations and batch ordering in the training process, and is exacerbated by the non-convex nature of parameter loss landscapes. The variance reduction explanation is consistent with the kernel perspective and GP correspondence of wide NNs: a random initialisation in parameter space induces a random GP function at initialisation (Corollary 2.6), which the NN learns to accommodate during training (c.f. Equation (2.52) and Chapter 3). Ensembling can be seen to simply average away this random noise. On the other hand, Allen-Zhu and Li (2020) argue that feature learning is a necessary consideration in order to explain the predictive accuracy benefits of ensemble in NNs, and provide experimental evidence that both the variance reduction hypothesis, and also considerations restricted to the kernel regime, are not sufficient in explaining ensembling in DNNs. We discuss these arguments further in the next subsection (Section 2.4.2).

In terms of explaining the predictive uncertainty benefits of NN ensembling, a key debate concerns the connection between deep ensembles and Bayesian inference, which arises because one can view the posterior predictive distribution at a test input $\boldsymbol{x}$ given training data $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$:

$$p(\cdot|\boldsymbol{x}, \mathcal{D}) = \int p(\cdot|f(\boldsymbol{x}, \boldsymbol{\theta}))p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$$

as an ensembled prediction, Equation (2.60), weighted by the posterior $p(\boldsymbol{\theta}|\mathcal{D})$.

While not originally presented as having a Bayesian interpretation (Lakshminarayanan et al., 2017), Wilson and Izmailov (2020) argue that deep ensembles are a closer approximation to the Bayes' posterior than many Bayesian DL approaches due to their ability to capture different modes in the non-convex parameter landscape of NNs (Fort et al., 2019). Many works have also derived updates to deep ensemble training schemes that enable a Bayesian interpretation to NN ensembling (Pearce et al., 2018; Ciosek et al., 2020; D' Angelo and Fortuin, 2021). Our work, in Chapter 3, provides a precise characterisation of the relationship between deep ensembles and Bayesian inference in the infinite-width limit, making use of the NTK and GP correspondences we described in Section 2.2.

Other works have studied the benefits of, and differences between, ensembling compared to simply training a single larger model (Lobacheva et al., 2020; Abe et al., 2022), while Wenzel et al. (2020); Zaidi et al. (2021) explore the predictive uncertainty benefits of explicitly promoting diversity in baselearners, by ensembling over different hyperparameters and architectures. Ortega et al. (2022) examine the interplay between diversity and generalisation in deep ensembles, and Schut et al. (2021) highlight the fact that ensemble diversity vanishes for wide DNN ensembles that exhibit feature learning, owing to a lack of function diversity at initialisation (c.f. Section 2.3.2).

2.4.2 Knowledge Distillation

Unlike ensembling, knowledge distillation (KD) is a technique that has enjoyed the most success in DNNs. In KD, the goal is to transfer the "knowledge" learnt by a *teacher* model (that is usually large and complex) to a *student* model (usually smaller and more efficient). The hope is that by doing so we can create a student model that achieves similar or better performance compared to the teacher model, while being faster and more resource-efficient to run.

The idea of compressing a large model into a smaller model first appeared in Buciluǎ et al. (2006), and the seminal work of Hinton et al. (2015) introduced KD in the context of DNNs. The setting we consider for KD involves two models: a teacher NN $f_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{T}})$ and student NN $f_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{S}})$, and we focus on KD for classification tasks. As we would like KD to be applicable even for different teacher/student model architectures and parameter spaces, it becomes unclear what we even mean by the "knowledge" contained in a model, and how it can be transfered across models.

In Hinton et al. (2015), the "knowledge" is taken to be contained in the *predictions* $f_{\mathcal{T}}$ and $f_{\mathcal{S}}$, and the student learns from a teacher model by adding a regulariser $\mathcal{R}(f_{\mathcal{S}}, f_{\mathcal{T}})$ to the student's training objective, that encourages the student to make more similar predictions to the teacher given the same input $\boldsymbol{x}$. The intuition for doing so is that there is "dark knowledge" contained in the teacher's predictions that can aid a student beyond simply the class label (in a classification setting), e.g. an image of a car is much more likely to be predicted as a truck than a carrot. We refer to such prediction-based KD as *vanilla KD* in Chapter 4.

Despite receiving considerable success and attention, vanilla KD is limited by the fact that it requires the teacher and student to be trained on datasets which possess the same classes. If we denote our teacher model $f_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{\theta}) = h_{\mathcal{T}}(\boldsymbol{x})^\top W_{\mathcal{T}}$ to have features $h_{\mathcal{T}}(\boldsymbol{x})$ and student model $f_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{\theta}) = h_{\mathcal{S}}(\boldsymbol{x})^\top W_{\mathcal{S}}$ to have features $h_{\mathcal{S}}(\boldsymbol{x})$, then one way to bypass this issue is to treat the features $h_{\mathcal{T}}(\boldsymbol{x})$ and $h_{\mathcal{S}}(\boldsymbol{x})$ as "knowledge" and to regularise the student towards the teacher instead in feature space given the same input $\boldsymbol{x}$, $\mathcal{R}(h_{\mathcal{S}}, h_{\mathcal{T}})$. Feature-based KD alternatives have also proven popular in the community (Romero et al., 2014; Zagoruyko and Komodakis, 2016a), but have issues of their own including needing additional parameters in order to map between potentially unmatched teacher and student architectures/feature-sizes.

An alternative framework for KD, which enjoys the benefit of being agnostic to both architectures and datasets, is to treat the "knowledge" in a model in terms of how "related" or "similar" different pairs of inputs $\boldsymbol{x}, \boldsymbol{x}'$ are viewed. For example, Relational KD (RKD) (Park et al., 2019b) encourages

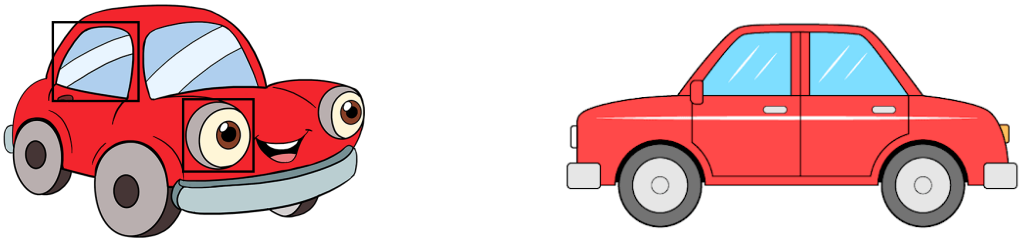


Figure 2.3: Left: multi-view nature for car class, where headlights and windows are both identifying features. Right: side image of car, which a model will only correctly identify if it has learnt the window feature (not just headlight).

the student to preserve distances across inputs in feature space i.e. $\|h_{\mathcal{S}}(\boldsymbol{x}) - h_{\mathcal{S}}(\boldsymbol{x}')\|_2^2$ to be regularised towards $\|h_{\mathcal{T}}(\boldsymbol{x}) - h_{\mathcal{T}}(\boldsymbol{x}')\|_2^2$ for any given pair $\boldsymbol{x}, \boldsymbol{x}'$. Related approaches like Similarity Preserving (SP) (Tung and Mori, 2019) or PKT (Passalis and Tefas, 2018) encourage feature space inner products $\langle h_{\mathcal{S}}(\boldsymbol{x}), h_{\mathcal{S}}(\boldsymbol{x}')\rangle$ and $\langle h_{\mathcal{T}}(\boldsymbol{x}), h_{\mathcal{T}}(\boldsymbol{x}')\rangle$ to be close, across different pairs of inputs $\boldsymbol{x}, \boldsymbol{x}'$. These approaches can be seen to be treating the teacher's learnt feature kernel, defined in Section 2.3, as the key source of "knowledge" in KD.

Though KD was originally proposed as a method to compress a large and computationally-expensive teacher model into a smaller student, Furlanello et al. (2018); Zhang et al. (2019c) demonstrated that in fact it is possible to obtain improvements in the student's generalisation performance even when the teacher shares *exactly the same* NN architecture and training regimes as the student. This is a setting known as *self-distillation*, and is a particularly interesting observation as it suggests that there are factors beyond the student NN's capacity that are needed in order to explain KD.

For vanilla KD, Allen-Zhu and Li (2020) provide the first step towards a formal study of the mechanisms for how "knowledge" is transferred in NNs from teacher to student, encompassing both KD and self-distillation. They posit that KD is inherently a feature learning process, that is unexplained via so-called *feature selection* methods like kernels, and is strongly linked to ensembling in NNs.

To do so, the authors hypothesise that real-world data like image data has a *multi-view* structure, in that many useful features exist that can be learnt, but gradient-based training leads a single NN model to only learn a *subset* of these features, which is enough for an NN to obtain zero training loss. In this proposed setup, different initialisations will bias different NN models to learn different subsets of features, and an ensembled model will learn the union of all features learnt in the constituent baselearner models, which is useful when presented with unusual inputs that only exhibit certain features. Moreover, Distillation helps in this case as we can think of KD as implicitly ensembling the student and teacher models into the student's architecture.

For example, the car class may be discerned by either the headlight or window features, but a student model may only learn the headlight feature. When presented with a side-view image of a car (Fig. 2.3 right) that obscures the headlight feature, our student model may fail to correctly identify it as a car, whereas an ensemble model that has learnt the window feature would have no issue. KD successfully allows the single student to learn the window feature from the teacher.

In Chapter 4, we consider the theoretical properties of "similarity" or "relation" based KD by extending the ideas of Allen-Zhu and Li (2020) to the setting where the feature kernel is the primary

source of "knowledge" to be transferred in KD. In our proposed *Feature Kernel Distillation* (FKD) we demonstrate that independent parameter initialisations bias different models to learn different feature kernels after training, and use our theoretical insights to motivate implementation considerations that improve FKD in practice.

Other works that have focused on understanding the behaviour of KD include Phuong and Lampert (2019) who theoretically studied the benefits of distillation in (deep) linear models, and Mobahi et al. (2020) who propose that self-distillation regularises the solutions in a Hilbert space of functions to help prevent overfitting. This connects to DL via the NTK regime for wide NNs, which has also been studied for KD by Ji and Zhu (2020). Empirically, Stanton et al. (2021) highlighted the interplay between student fidelity (i.e. how accurately the student mimics the teacher) and generalisation, whereas Beyer et al. (2022) focus on identifying the specific KD training setups that result in best student generalisation performance. Gou et al. (2021) present an excellent review of the literature surrounding KD.

2.4.3 Self-supervised learning

A popular data setting nowadays is where one has access to large amounts of unlabelled inputs $\mathbf{X}$, due to the higher cost associated with obtaining labels Y. *Self-supervised learning* (SSL) aims to learn useful DNN feature representations from *only* unlabelled input data, that can later be used to improve performance in some downstream task. SSL has proven to be an extremely effective paradigm in DL systems, especially for natural language, where a popular approach is when a model is *pre-trained* on unlabeled language data by predicting masked words in a large number of different sentences (Vincent et al., 2008; Collobert et al., 2011; Devlin et al., 2018).

For vision data like images, SSL has also shown remarkable potential in closing the gap to standard supervised settings, though masking approaches (Chen et al., 2020a; Dosovitskiy et al., 2020) are less successful in vision (Chen et al., 2021), possibly due to the high-dimensional and real-valued nature of e.g. image pixel space, compared to language, which is discrete and tokenisable. Instead, a popular technique for SSL in vision uses so-called *joint embeddings* (Becker and Hinton, 1992; Bromley et al., 1993; Goldberger et al., 2004; Chopra et al., 2005; Hadsell et al., 2006).

In the joint embedding approach, we consider two transformations, $\mathcal{T}_1(\boldsymbol{x})$ and $\mathcal{T}_2(\boldsymbol{x})$, of an image $\boldsymbol{x}$ e.g. two different data augmentations like crops or rotations of the same image of a cat. Given some NN feature extractor $h(\boldsymbol{x})$, the aim is then to pre-train h such that the feature representations of $\mathcal{T}_1(\boldsymbol{x})$ and $\mathcal{T}_2(\boldsymbol{x})$ are close in feature-space i.e. such that $\|h(\mathcal{T}_1(\boldsymbol{x})) - h(\mathcal{T}_2(\boldsymbol{x}))\|_2^2$ is small, across different unlabelled images $\boldsymbol{x} \in \mathbf{X}$. After this initial pre-training phase, the features $h(\cdot)$ are used as the backbone for a new model that is trained on labelled data from a downstream task. For example, we could pretrain on the larger ImageNet-1K dataset, and treat the smaller dataset like CIFAR-10 as the downstream task (e.g. in Chen et al. (2020b)).

One clear issue with the joint embedding approach is that it is possible for the feature extractor to make features of different input transformations to be close by "cheating" and making all inputs be mapped to the same vector in feature space, $h(\boldsymbol{x}) = h^*$, $\forall \boldsymbol{x}$. This is known as *feature collapse* and is intimately linked to the signal propagation issues with deep NNs at initialisation we discussed in

Section 2.2.2. With collapsed features, the feature extractor cannot discern between different inputs, and is thus of no use for downstream tasks.

To get by feature collapse, a range of methods within the joint-embedding framework have been proposed in the literature. These include: 1) contrastive approaches that encourage different inputs to have distinct features (Chen et al., 2020b; He et al., 2020c); 2) non-contrastive methods that use asymmetry between the joint embeddings (Grill et al., 2020; Chen and He, 2021); 3) clustering (Caron et al., 2020) and 4) feature whitening (Ermolov et al., 2021; Zbontar et al., 2021; Bardes et al., 2021). These different methods all share the same goal of prevent SSL training from reducing to the degenerate state of collapsed features.

A range of works have studied the theoretical properties of joint-embedding SSL, with much attention focused on demonstrating the generalisation benefits on downstream tasks in contrastive SSL (Arora et al., 2019c; Tosh et al., 2021; Nozawa and Sato, 2021; Awasthi et al., 2022). Wen and Li (2021) study constrastive learning as a feature learning process to highlight the role of data augmentation, and Balestriero and LeCun (2022) unify different joint-embedding approaches by connecting their solutions to different spectral methods, like kernel CCA (Lai and Fyfe, 2000). Dubois et al. (2022) provide conditions for ideal SSL-learnt representations, in terms of optimality on downstream tasks that are invariant to the data augmentations used.

Tian et al. (2021) instead study factors behind the learning dynamics that prevent feature collapse in asymmetric non-constrastive methods like BYOL (Grill et al., 2020) or SimSiam (Chen and He, 2021). Key to their findings is a simplified deep linear setting where one can consider the dynamics of each eigenmode (of the representations $h(\mathbf{X})$) *independently*, which sheds light on the roles of different hyper-parameters like weight decay, learning rate, and momentum in avoiding collapse in practice. Halvagal et al. (2022) extend the framework of Tian et al. (2021) to demonstrate that BYOL and SimSiam also have an implicit variance regularisation effect, akin to the explicit regularisation in VICReg (Bardes et al., 2021).

A related phenomenon to feature collapse in SSL is known as *dimensional collapse* (Hua et al., 2021), which corresponds to partial collapse of the feature space during SSL training. More formally, if we denote $\mathbf{X} \in \mathbb{R}^{n \times d}$ to be a large unlabelled dataset of n inputs, and an m-dimensional feature space $h(\mathbf{X}) \in \mathbb{R}^{n \times m}$ with $m < n$, then dimensional collapse corresponds to the setting where $\frac{1}{n} h(\mathbf{X})^\top h(\mathbf{X})$ has fewer than m non-zero eigenvalues. Jing et al. (2021) study the occurence of dimensional collapse in the context of contrastive SSL, identifying factors like strong data augmentation and implicit regularisation as possible causes. In Chapter 5, we study the properties of learnt features in SSL through the *rate of decay* in their eigenvalues, identifying factors that affect this eigenspectrum and highlighting its importance for downstream task performance.

Chapter 3

Bayesian Deep Ensembles via the Neural Tangent Kernel

This paper was published as the following

Bobby He, Balaji Lakshminarayanan, Yee Whye Teh. Bayesian Deep Ensembles via the Neural Tangent Kernel. In Advances in Neural Information Processing Systems 33, 2020.

Bayesian Deep Ensembles via the Neural Tangent Kernel

Bobby He
Department of Statistics
University of Oxford
bobby.he@stats.ox.ac.uk

Balaji Lakshminarayanan
Google Research
Brain team
balajiln@google.com

Yee Whye Teh
Department of Statistics
University of Oxford
y.w.teh@stats.ox.ac.uk

Abstract

We explore the link between deep ensembles and Gaussian processes (GPs) through the lens of the Neural Tangent Kernel (NTK): a recent development in understanding the training dynamics of wide neural networks (NNs). Previous work has shown that even in the infinite width limit, when NNs become GPs, there is no GP posterior interpretation to a deep ensemble trained with squared error loss. We introduce a simple modification to standard deep ensembles training, through addition of a computationally-tractable, randomised and untrainable function to each ensemble member, that enables a posterior interpretation in the infinite width limit. When ensembled together, our trained NNs give an approximation to a posterior predictive distribution, and we prove that our Bayesian deep ensembles make more conservative predictions than standard deep ensembles in the infinite width limit. Finally, using finite width NNs we demonstrate that our Bayesian deep ensembles faithfully emulate the analytic posterior predictive when available, and can outperform standard deep ensembles in various out-of-distribution settings, for both regression and classification tasks.

1 Introduction

Consider a training dataset $\mathcal{D}$ consisting of N i.i.d. data points $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, with $\boldsymbol{x} \in \mathbb{R}^d$ representing d-dimensional features and y representing C-dimensional targets. Given input features $\boldsymbol{x}$ and parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ we use the output, $f(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^C$, of a neural network (NN) to model the predictive distribution $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ over the targets. For univariate regression tasks, $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ will be Gaussian: $-\log p(y|\boldsymbol{x}, \boldsymbol{\theta})$ is the squared error $\frac{1}{2\sigma^2}(y - f(\boldsymbol{x}, \boldsymbol{\theta}))^2$ up to additive constant, for fixed observation noise $\sigma^2 \in \mathbb{R}_+$. For classification tasks, $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ will be a Categorical distribution.

Given a prior distribution $p(\boldsymbol{\theta})$ over the parameters, we can define the posterior over $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}|\mathcal{D})$, using Bayes' rule and subsequently the *posterior predictive* distribution at a test point $(\boldsymbol{x}^*, y^*)$:

$$p(y^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(y^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, d\boldsymbol{\theta} \tag{1}$$

The posterior predictive is appealing as it represents a marginalisation over $\boldsymbol{\theta}$ weighted by posterior probabilities, and has been shown to be optimal for minimising predictive risk under a well-specified model [1]. However, one issue with the posterior predictive for NNs is that it is computationally intensive to calculate the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ exactly. Several approximations to $p(\boldsymbol{\theta}|\mathcal{D})$ have been introduced for *Bayesian neural networks* (BNNs) including: Laplace approximation [2]; Markov chain Monte Carlo [3, 4]; variational inference [5–9]; and Monte-Carlo dropout [10].

Despite the recent interest in BNNs, it has been shown empirically that deep ensembles [11], which lack a principled Bayesian justification, outperform existing BNNs in terms of uncertainty quantification and out-of-distribution robustness, cf. [12]. Deep ensembles independently initialise and train individual NNs (referred to herein as *baselearners*) on the negative log-likelihood loss

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta}))$ with $\ell(y, f(\boldsymbol{x}, \boldsymbol{\theta})) = -\log p(y|\boldsymbol{x}, \boldsymbol{\theta})$, before aggregating predictions. Understanding the success of deep ensembles, particularly in relation to Bayesian inference, is a key question in the uncertainty quantification and Bayesian deep learning communities at present: Fort et al. [13] suggested that the empirical performance of deep ensembles is explained by their ability to explore different functional modes, while Wilson and Izmailov [14] argued that deep ensembles are actually approximating the posterior predictive.

In this work, we will relate deep ensembles to Bayesian inference, using recent developments connecting GPs and wide NNs, both before [15–21] and after [22, 23] training. Using these insights, we devise a modification to standard NN training that yields an exact posterior sample for $f(\cdot, \boldsymbol{\theta})$ in the infinite width limit. As a result, when ensembled together our modified baselearners give a posterior predictive approximation, and can thus be viewed as a *Bayesian deep ensemble*.

One concept that is related to our methods concerns ensembles trained with *Randomised Priors* to give an approximate posterior interpretation, which we will use when modelling observation noise in regression tasks. The idea behind randomised priors is that, under certain conditions, regularising baselearner NNs towards independently drawn "priors" during training produces exact posterior samples for $f(\cdot, \boldsymbol{\theta})$. Randomised priors recently appeared in machine learning applied to reinforcement learning [24] and uncertainty quantification [25, 26], like this work. To the best of our knowledge, related ideas first appeared in astrophysics where they were applied to Gaussian random fields [27]. However, one such condition for posterior exactness with randomised priors is that the model $f(\boldsymbol{x}, \boldsymbol{\theta})$ is linear in $\boldsymbol{\theta}$. This is not true in general for NNs, but has been shown to hold for wide NNs local to their parameter initialisation, in a recent line of work. In order to introduce our methods, we will first review this line of work, known as the *Neural Tangent Kernel* (NTK) [22].

2 NTK Background

Wide NNs, and their relation to GPs, have been a fruitful area recently for the theoretical study of NNs: we review only the most salient developments to this work, due to limited space.

First introduced by Jacot et al. [22], the *empirical NTK* of $f(\cdot, \boldsymbol{\theta}_t)$ is, for inputs $\boldsymbol{x}, \boldsymbol{x}'$, the kernel:

$$\hat{\Theta}_t(\boldsymbol{x}, \boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}', \boldsymbol{\theta}_t) \rangle \tag{2}$$

and describes the functional gradient of a NN in terms of the current loss incurred on the training set. Note that $\boldsymbol{\theta}_t$ depends on a random initialisation $\boldsymbol{\theta}_0$, thus the empirical NTK is random for all $t > 0$.

Jacot et al. [22] showed that for an MLP under a so-called NTK parameterisation, detailed in Appendix A, the empirical NTK converges in probability to a deterministic limit Θ, that stays constant during gradient training, as the hidden layer widths of the NN go to infinity sequentially. Later, Yang [28, 29] extended the NTK convergence result to convergence almost surely, which is proven rigorously for a variety of architectures and for widths (or channels in Convolutional NNs) of hidden layers going to infinity in unison. This limiting positive-definite (p.d.) kernel Θ, known as the NTK, depends only on certain NN architecture choices, including: activation, depth and variances for weight and bias parameters. Note that the NTK parameterisation can be thought of as akin to training under standard parameterisation with a learning rate that is inversely proportional to the width of the NN, which has been shown to be the largest scale for stable learning rates in wide NNs [30–32].

Lee et al. [23] built on the results of Jacot et al. [22], and studied the *linearised regime* of an NN. Specifically, if we denote as $f_t(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{\theta}_t)$ the network function at time t, we can define the first order Taylor expansion of the network function around randomly initialised parameters $\boldsymbol{\theta}_0$ to be:

$$f_t^{\text{lin}}(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0) \Delta\boldsymbol{\theta}_t \tag{3}$$

where $\Delta\boldsymbol{\theta}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$ and $f_0 = f(\cdot, \boldsymbol{\theta}_0)$ is the randomly initialised NN function.

For notational clarity, whenever we evaluate a function at an arbitrary input set $\mathcal{X}'$ instead of a single point $\boldsymbol{x}'$, we suppose the function is vectorised. For example, $f_t(\mathcal{X}) \in \mathbb{R}^{NC}$ denotes the concatenated NN outputs on training set $\mathcal{X}$, whereas $\nabla_{\boldsymbol{\theta}} f_t(\mathcal{X}) = \nabla_{\boldsymbol{\theta}} f(\mathcal{X}, \boldsymbol{\theta}_t) \in \mathbb{R}^{NC \times p}$. In the interest of space, we will also sometimes use subscripts to signify kernel inputs, so for instance $\Theta_{\boldsymbol{x}'\mathcal{X}} = \Theta(\boldsymbol{x}', \mathcal{X}) \in \mathbb{R}^{C \times NC}$ and $\Theta_{\mathcal{X}\mathcal{X}} = \Theta(\mathcal{X}, \mathcal{X}) \in \mathbb{R}^{NC \times NC}$ throughout this work.

The results of Lee et al. [23] showed that in the infinite width limit, with NTK parameterisation and gradient flow under squared error loss, $f_t^{\text{lin}}(\boldsymbol{x})$ and $f_t(\boldsymbol{x})$ are equal for any $t \geq 0$, for a shared random

initialisation $\boldsymbol{\theta}_0$. In particular, for the linearised network it can be shown, that as $t \to \infty$:

$$f_\infty^{\text{lin}}(\boldsymbol{x}) = f_0(\boldsymbol{x}) - \hat{\Theta}_0(\boldsymbol{x}, \mathcal{X})\hat{\Theta}_0(\mathcal{X}, \mathcal{X})^{-1}(f_0(\mathcal{X}) - \mathcal{Y}) \tag{4}$$

and thus as the hidden layer widths converge to infinity we have that:

$$f_\infty^{\text{lin}}(\boldsymbol{x}) = f_\infty(\boldsymbol{x}) = f_0(\boldsymbol{x}) - \Theta(\boldsymbol{x}, \mathcal{X})\Theta(\mathcal{X}, \mathcal{X})^{-1}(f_0(\mathcal{X}) - \mathcal{Y}) \tag{5}$$

We can replace $\Theta(\mathcal{X}, \mathcal{X})^{-1}$ with the generalised inverse when invertibility is an issue. However, this will not be a main concern of this work, as our methods will add regularisation that corresponds to modelling observation/output noise, which both ensures invertibility and alleviates any potential convergence issues due to fast decay of the NTK eigenspectrum [33].

From Eq. (5) we see that, conditional on the training data $\{\mathcal{X}, \mathcal{Y}\}$, we can decompose f_∞ into $f_\infty(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \gamma(\boldsymbol{x})$ where $\mu(\boldsymbol{x}) = \Theta(\boldsymbol{x}, \mathcal{X})\Theta(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y}$ is a deterministic mean and $\gamma(\boldsymbol{x}) = f_0(\boldsymbol{x}) - \Theta(\boldsymbol{x}, \mathcal{X})\Theta(\mathcal{X}, \mathcal{X})^{-1}f_0(\mathcal{X})$ captures predictive uncertainty, due to the randomness of f_0. Now, if we suppose that, at initialisation, $f_0 \overset{d}{\sim} \mathcal{GP}(0, k)$ for an arbitrary kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{C \times C}$, then we have $f_\infty(\cdot) \overset{d}{\sim} \mathcal{GP}(\mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}, \boldsymbol{x}'))$ for two inputs $\boldsymbol{x}, \boldsymbol{x}'$, where:[1]

$$\Sigma(\boldsymbol{x}, \boldsymbol{x}') = k_{\boldsymbol{x}\boldsymbol{x}'} + \Theta_{\boldsymbol{x}\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}k_{\mathcal{X}\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}\Theta_{\mathcal{X}\boldsymbol{x}} - \left(\Theta_{\boldsymbol{x}\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}k_{\mathcal{X}\boldsymbol{x}'} + h.c.\right) \tag{6}$$

For a generic kernel k, Lee et al. [23] observed that this limiting distribution for f_∞ does not have a posterior GP interpretation unless k and Θ are multiples of each other.

As mentioned in Section 1, previous work [15–21] has shown that there is a distinct but closely related kernel $\mathcal{K}$, known as the *Neural Network Gaussian Process* (NNGP) kernel, such that $f_0 \overset{d}{\to} \mathcal{GP}(0, \mathcal{K})$ at initialisation in the infinite width limit and $\mathcal{K} \neq \Theta$. Thus Eq. (6) with $k{=}\mathcal{K}$ tells us that, for wide NNs under squared error loss, there is no Bayesian posterior interpretation to a trained NN, nor is there an interpretation to a trained deep ensemble as a Bayesian posterior predictive approximation.

3 Proposed modification to obtain posterior samples in infinite width limit

Lee et al. [23] noted that one way to obtain a posterior interpretation to f_∞ is by randomly initialising f_0 but only training the parameters in the final linear readout layer, as the contribution to the NTK Θ from the parameters in final hidden layer is exactly the NNGP kernel $\mathcal{K}$.[2] f_∞ is then a sample from the GP posterior with prior kernel NNGP, $\mathcal{K}$, and noiseless observations in the infinite width limit i.e. $f_\infty(\mathcal{X}') \overset{d}{\sim} \mathcal{N}(\mathcal{K}_{\mathcal{X}'\mathcal{X}}\mathcal{K}_{\mathcal{X}\mathcal{X}}^{-1}\mathcal{Y},\ \mathcal{K}_{\mathcal{X}'\mathcal{X}'} - \mathcal{K}_{\mathcal{X}'\mathcal{X}}\mathcal{K}_{\mathcal{X}\mathcal{X}}^{-1}\mathcal{K}_{\mathcal{X}\mathcal{X}'})$. This is an example of the "sample-then-optimise" procedure of Matthews et al. [34], but, by only training the final layer this procedure limits the earlier layers of an NN solely to be random feature extractors.

We now introduce our modification to standard training that trains all layers of a finite width NN and obtains an exact posterior interpretation in the infinite width limit with NTK parameterisation and squared error loss. For notational purposes, let us suppose $\boldsymbol{\theta} = \texttt{concat}(\{\boldsymbol{\theta}^{\leq L}, \boldsymbol{\theta}^{L+1}\})$ with $\boldsymbol{\theta}^{\leq L} \in \mathbb{R}^{p - p_{L+1}}$ denoting L hidden layers, and $\boldsymbol{\theta}^{L+1} \in \mathbb{R}^{p_{L+1}}$ denoting final readout layer $L{+}1$. Moreover, define $\Theta^{\leq L} = \Theta - \mathcal{K}$ to be the p.d. kernel corresponding to contributions to the NTK from all parameters before the final layer, and $\hat{\Theta}_t^{\leq L}$ to be the empirical counterpart depending on $\boldsymbol{\theta}_t$. To motivate our modification, we reinterpret f_t^{lin} in Eq. (3) by splitting terms related to $\mathcal{K}$ and $\Theta^{\leq L}$:

$$f_t^{\text{lin}}(\boldsymbol{x}) = \underbrace{f_0(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}^{L+1}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\Delta\boldsymbol{\theta}_t^{L+1}}_{\mathcal{K}} + \underbrace{\mathbf{0}_C + \nabla_{\boldsymbol{\theta}^{\leq L}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\Delta\boldsymbol{\theta}_t^{\leq L}}_{\Theta - \mathcal{K}} \tag{7}$$

where $\mathbf{0}_C \in \mathbb{R}^C$ is the zero vector. As seen in Eq. (7), the distribution of $f_0^{\text{lin}}(\boldsymbol{x}) = f_0(\boldsymbol{x})$ lacks extra variance, $\Theta^{\leq L}(\boldsymbol{x}, \boldsymbol{x})$, that accounts for contributions to the NTK Θ from all parameters $\boldsymbol{\theta}^{\leq L}$ before the final layer. This is precisely why no Bayesian intepretation exists for a standard trained wide NN, as in Eq. (6) with $k{=}\mathcal{K}$. The motivation behind our modification is now very simple: we propose to manually add in this missing variance. Our modified NNs, $\tilde{f}(\cdot, \boldsymbol{\theta})$, will then have trained distribution:

$$\tilde{f}_\infty(\mathcal{X}') \overset{d}{\sim} \mathcal{N}(\Theta_{\mathcal{X}'\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}\mathcal{Y},\ \Theta_{\mathcal{X}'\mathcal{X}'} - \Theta_{\mathcal{X}'\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}\Theta_{\mathcal{X}\mathcal{X}'}) \tag{8}$$

[1] Throughout this work, the notation "$+h.c.$" means "plus the Hermitian conjugate", like Lee et al. [23]. For example: $\Theta_{\boldsymbol{x}\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}k_{\mathcal{X}\boldsymbol{x}'} + h.c. = \Theta_{\boldsymbol{x}\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}k_{\mathcal{X}\boldsymbol{x}'} + \Theta_{\boldsymbol{x}'\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}k_{\mathcal{X}\boldsymbol{x}}$

[2] Up to a multiple of last layer width in standard parameterisation.

on a test set $\mathcal{X}'$, in the infinite width limit. Note that Eq. (8) is the GP posterior using prior kernel Θ and noiseless observations $\tilde{f}_\infty(\mathcal{X})=\mathcal{Y}$, which we will refer to as the *NTKGP* posterior predictive. We construct $\tilde{f}$ by sampling a random and untrainable function $\delta(\cdot)$ that is added to the standard forward pass $f(\cdot, \boldsymbol{\theta}_t)$, defining an augmented forward pass:

$$\tilde{f}(\cdot, \boldsymbol{\theta}_t) = f(\cdot, \boldsymbol{\theta}_t) + \delta(\cdot) \tag{9}$$

Given a parameter initialisation scheme $\texttt{init}(\cdot)$ and initial parameters $\boldsymbol{\theta}_0 \overset{d}{\sim} \texttt{init}(\cdot)$, our chosen formulation for $\delta(\cdot)$ is as follows: 1) sample $\tilde{\boldsymbol{\theta}} \overset{d}{\sim} \texttt{init}(\cdot)$ independently of $\boldsymbol{\theta}_0$; 2) denote $\tilde{\boldsymbol{\theta}} = \texttt{concat}(\{\tilde{\boldsymbol{\theta}}^{\leq L}, \tilde{\boldsymbol{\theta}}^{L+1}\})$; and 3) define $\boldsymbol{\theta}^* = \texttt{concat}(\{\tilde{\boldsymbol{\theta}}^{\leq L}, \mathbf{0}_{p_{L+1}}\})$. In words, we set the parameters in the final layer of an independently sampled $\tilde{\boldsymbol{\theta}}$ to zero to obtain $\boldsymbol{\theta}^*$. Now, we define:

$$\delta(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\boldsymbol{\theta}^* \tag{10}$$

There are a few important details to note about $\delta(\cdot)$ as defined in Eq. (10). First, $\delta(\cdot)$ has the same distribution in both NTK and standard parameterisations,[3] and also $\delta(\cdot) \mid \boldsymbol{\theta}_0 \overset{d}{\sim} \mathcal{GP}\big(0, \hat{\Theta}_0^{\leq L}\big)$ in the NTK parameterisation.[4] Moreover, Eq. (10) can be viewed as a single Jacobian-vector product (JVP), which packages that offer forward-mode autodifferentiation (AD), such as JAX [35], are efficient at computing for finite NNs. It is worth noting that our modification adds only negligible computational and memory requirements on top of standard deep ensembles [11]: a more nuanced comparison can be found in Appendix G. Alternative constructions of $\tilde{f}$ are presented in Appendix C.

To ascertain whether a trained $\tilde{f}_\infty$ constructed via Eqs. (9, 10) returns a sample from the GP posterior Eq. (8) for wide NNs, the following proposition, which we prove in Appendix B.1, will be useful:

Proposition 1. *$\delta(\cdot) \overset{d}{\to} \mathcal{GP}(0, \Theta^{\leq L})$ and is independent of $f_0(\cdot)$ in the infinite width limit. Thus, $\tilde{f}_0(\cdot) = f_0(\cdot) + \delta(\cdot) \overset{d}{\to} \mathcal{GP}(0, \Theta)$.*

Using Proposition 1, we now consider the linearisation of $\tilde{f}_t(\cdot)$, noting that $\nabla_{\boldsymbol{\theta}} \tilde{f}_0(\cdot) = \nabla_{\boldsymbol{\theta}} f_0(\cdot)$:

$$\tilde{f}_t^{\text{lin}}(\boldsymbol{x}) = \underbrace{f_0(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}^{L+1}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\Delta\boldsymbol{\theta}_t^{L+1}}_{\mathcal{K}} + \underbrace{\delta(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}^{\leq L}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\Delta\boldsymbol{\theta}_t^{\leq L}}_{\Theta - \mathcal{K}} \tag{11}$$

The fact that $\nabla_{\boldsymbol{\theta}} \tilde{f}_t^{\text{lin}}(\cdot) = \nabla_{\boldsymbol{\theta}} f_0(\cdot)$ is crucial in Eq. (11), as this initial Jacobian is the feature map of the linearised NN regime from Lee et al. [23]. As per Proposition 1 and Eq. (11), we see that $\delta(\boldsymbol{x})$ adds the extra randomness missing from $f_0^{\text{lin}}(\boldsymbol{x})$ in Eq. (7), and reinitialises $\tilde{f}_0$ as a sample from $\mathcal{GP}(0, \mathcal{K})$ to $\mathcal{GP}(0, \Theta)$ for wide NNs. This means we can set $k = \Theta$ in Eq. (6) and deduce:

Corollary 1. *$\tilde{f}_\infty(\mathcal{X}') \overset{d}{\sim} \mathcal{N}(\Theta_{\mathcal{X}'\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}\mathcal{Y},\ \Theta_{\mathcal{X}'\mathcal{X}'} - \Theta_{\mathcal{X}'\mathcal{X}}\Theta_{\mathcal{X}\mathcal{X}}^{-1}\Theta_{\mathcal{X}\mathcal{X}'})$, and hence a trained $\tilde{f}_\infty$ returns a sample from the posterior NTKGP in the infinite width limit.*

To summarise: we define our new NN forward pass to give $\tilde{f}_t(\boldsymbol{x}) = f_t(\boldsymbol{x}) + \delta(\boldsymbol{x})$ for standard forward pass $f_t(\boldsymbol{x})$, and an untrainable $\delta(\boldsymbol{x})$ defined as in Eq. (10). As given by Corollary 1, independently trained baselearners $\tilde{f}_\infty$ can then be ensembled to approximate the NTKGP posterior predictive. We will call $\tilde{f}_\infty$ trained in this section an NTKGP baselearner, regardless of parameterisation or width. We are aware that the name NTK-GP has been used previously to refer to Eq. (6) with NNGP kernel $\mathcal{K}$, which is what standard training under squared error with a wide NN yields. However, we believe GPs in machine learning are synonymous with probabilistic inference [36], which Eq. (6) has no connection to in general, so we feel the name NTKGP is more appropriate for our methods.

[3] In this work Θ always denotes the NTK under NTK parameterisation. It is also possible to model Θ to be the scaled NTK under standard parameterisation (which depends on layer widths) as in Sohl-Dickstein et al. [32] with minor reweightings to both $\delta(\cdot)$ and, when modelling observation noise, the L^2-regularisation described in Appendix D.

[4] With NTK parameterisation, it is easy to see that $\delta(\cdot) \mid \boldsymbol{\theta}_0 \overset{d}{\sim} \mathcal{GP}\big(0, \hat{\Theta}_0^{\leq L}\big)$, because $\tilde{\boldsymbol{\theta}}^{\leq L} \overset{d}{\sim} \mathcal{N}(0, I_{p-p_{L+1}})$. To extend this to standard parameterisation, note that Eq. (10) is just the first order term in the Taylor expansion of $f(\boldsymbol{x}, \boldsymbol{\theta}_0 + \boldsymbol{\theta}^*)$, which has a parameterisation agnostic distribution, about $\boldsymbol{\theta}_0$.

3.1 Modelling observation noise

So far, we have used squared loss $\ell(y, \tilde{f}(\boldsymbol{x}, \boldsymbol{\theta})) = \frac{1}{2\sigma^2}(y - \tilde{f}(\boldsymbol{x}, \boldsymbol{\theta}))^2$ for σ^2=1, and seen how our NTKGP training scheme for $\tilde{f}$ gives a Bayesian interpretation to trained networks when we assume noiseless observations. Lemma 3 of Osband et al. [24] shows us how to draw a posterior sample for linear $\tilde{f}$ if we wish to model Gaussian observation noise $y \overset{d}{\sim} \mathcal{N}(\tilde{f}(\boldsymbol{x}, \boldsymbol{\theta}), \sigma^2)$ for $\sigma^2>0$: by adding i.i.d. noise to targets $y'_n \overset{d}{\sim} \mathcal{N}(y_n, \sigma^2)$ and regularising $\mathcal{L}(\boldsymbol{\theta})$ with a weighted L^2 term, either $\|\boldsymbol{\theta}\|^2_\Lambda$ or $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2_\Lambda$, depending on if you regularise in function space or parameter space. The weighting Λ is detailed in Appendix D. These methods were introduced by Osband et al. [24] for the application of Q-learning in deep reinforcement learning, and are known as *Randomised Prior parameter* (RP-param) and *Randomised Prior function* (RP-fn) respectively. The randomised prior (RP) methods were motivated by a Bayesian linear regression approximation of the NN, but they do not take into account the difference between the NNGP and the NTK. Our NTKGP methods can be viewed as a way to fix this for both the parameter space or function space methods, which we will name NTKGP-param and NTKGP-fn respectively. Similar regularisation ideas were explored in connection to the NTK by Hu et al. [37], when the NN function is initialised from the origin, akin to kernel ridge regression.

3.2 Comparison of predictive distributions in infinite width

Having introduced the different ensemble training methods considered in this paper: NNGP; deep ensembles; randomised prior; and NTKGP, we will now compare their predictive distributions in the infinite width limit with squared error loss. Table 1 displays these limiting distributions, $f_\infty(\cdot) \overset{d}{\sim} \mathcal{GP}(\mu, \Sigma)$, and should be viewed as an extension to Equation (16) of Lee et al. [23]. In

Table 1: Predictive distributions of wide ensembles for various training methods. *std* denotes standard training with $f(\boldsymbol{x}, \boldsymbol{\theta})$, and *ours* denotes training using our additive $\delta(\boldsymbol{x})$ to make $\tilde{f}(\boldsymbol{x}, \boldsymbol{\theta})$.

Method	Layers trained	Output Noise	$\mu(\boldsymbol{x})$	$\Sigma(\boldsymbol{x}, \boldsymbol{x}')$
NNGP	Final	$\sigma^2 \geq 0$	$\mathcal{K}_{\boldsymbol{x}\mathcal{X}}(\mathcal{K}_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - \mathcal{K}_{\boldsymbol{x}\mathcal{X}}(\mathcal{K}_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'}$
Deep Ensembles	All (*std*)	$\sigma^2 = 0$	$\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - (\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'} + h.c.)$ $+\Theta_{\boldsymbol{x}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\mathcal{K}_{\mathcal{X}\mathcal{X}}\Theta^{-1}_{\mathcal{X}\mathcal{X}}\Theta_{\mathcal{X}\boldsymbol{x}'}$
Randomised Prior	All (*std*)	$\sigma^2 > 0$	$\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{Y}$	$\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} - (\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{K}_{\mathcal{X}\boldsymbol{x}'} + h.c.)$ $+\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}(\mathcal{K}_{\mathcal{X}\mathcal{X}}+\sigma^2 I)(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\Theta_{\mathcal{X}\boldsymbol{x}'}$
NTKGP	All (*ours*)	$\sigma^2 \geq 0$	$\Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\mathcal{Y}$	$\Theta_{\boldsymbol{x}\boldsymbol{x}'} - \Theta_{\boldsymbol{x}\mathcal{X}}(\Theta_{\mathcal{X}\mathcal{X}}+\sigma^2 I)^{-1}\Theta_{\mathcal{X}\boldsymbol{x}'}$

order to parse Table 1, let us denote $\mu_{\text{NNGP}}, \mu_{\text{DE}}, \mu_{\text{RP}}, \mu_{\text{NTKGP}}$ and $\Sigma_{\text{NNGP}}, \Sigma_{\text{DE}}, \Sigma_{\text{RP}}, \Sigma_{\text{NTKGP}}$ to be the entries in the $\mu(\boldsymbol{x})$ and $\Sigma(\boldsymbol{x}, \boldsymbol{x}')$ columns of Table 1 respectively, read from top to bottom. We see that $\mu_{\text{DE}}(\boldsymbol{x}) = \mu_{\text{NTKGP}}(\boldsymbol{x})$ if σ^2=0, and $\mu_{\text{RP}}(\boldsymbol{x}) = \mu_{\text{NTKGP}}(\boldsymbol{x})$ if $\sigma^2>0$. In words: the predictive mean of a trained ensemble is the same when training all layers, both with standard training and our NTKGP training. This holds because both f_0 and $\tilde{f}_0$ are zero mean. It is also possible to compare the predictive covariances as the following proposition, proven in Appendix B.2, shows:

Proposition 2. *For σ^2=0, $\Sigma_{NTKGP} \succeq \Sigma_{DE} \succeq \Sigma_{NNGP}$. Similarly, for $\sigma^2>0$, $\Sigma_{NTKGP} \succeq \Sigma_{RP} \succeq \Sigma_{NNGP}$.*

Here, when we write $k_1 \succeq k_2$ for p.d. kernels k_1, k_2, we mean that $k_1 - k_2$ is also a p.d. kernel. One consequence of Proposition 2 is that the predictive distribution of an ensemble of NNs trained via our NTKGP methods is always more conservative than a standard deep ensemble, in the linearised NN regime, when the ensemble size $K \to \infty$. It is not possible to say in general when this will be beneficial, because in practice our models will always be misspecified. However, Proposition 2 suggests that in situations where we suspect standard deep ensembles might be overconfident, such as in situations where we expect some dataset shift at test time, our methods should hold an advantage. Note, wide randomised prior ensembles ($\sigma > 0$) were also theoretically shown to make more conservative predictions than corresponding NNGP posteriors in Ciosek et al. [26], albeit without the connection to the NTK.

3.3 Modelling heteroscedasticity

Following Lakshminarayanan et al. [11], if we wish to model heteroscedasticity in a univariate regression setting such that each training point, $(\boldsymbol{x}_n, y_n)$, has an individual observation noise $\sigma^2(\boldsymbol{x}_n)$ then we use the heteroscedastic Gaussian NLL loss (up to additive constant):

$$\ell(y'_n, \tilde{f}(\boldsymbol{x}_n, \boldsymbol{\theta})) = \frac{(y'_n - \tilde{f}(\boldsymbol{x}_n, \boldsymbol{\theta}))^2}{2\sigma^2(\boldsymbol{x}_n)} + \frac{\log \sigma^2(\boldsymbol{x}_n)}{2} \tag{12}$$

where $y'_n = y_n + \sigma(\boldsymbol{x}_n)\epsilon_n$ and $\epsilon_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. It is easy to see that for fixed $\sigma^2(\boldsymbol{x}_n)$, our NTKGP trained baselearners will still have a Bayesian interpretation: $\mathcal{Y}' \leftarrow \Sigma^{-\frac{1}{2}}\mathcal{Y}'$ and $\tilde{f}(\mathcal{X}, \boldsymbol{\theta}) \leftarrow \Sigma^{-\frac{1}{2}}\tilde{f}(\mathcal{X}, \boldsymbol{\theta})$ returns us to the homoscedastic case, where $\Sigma = \text{diag}(\sigma^2(\mathcal{X})) \in \mathbb{R}^{N \times N}$. We will follow Lakshminarayanan et al. [11] and parameterise $\sigma^2(\boldsymbol{x}) = \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})$ by an extra output head of the NN, that is trainable alongside the mean function $\mu_{\boldsymbol{\theta}}(\boldsymbol{x})$ when modelling heteroscedasticity.[5]

3.4 NTKGP Ensemble Algorithms

We now proceed to train an ensemble of K NTKGP baselearners. Like previous work [11, 24], we independently initialise baselearners, and also use a fixed, independently sampled training set noise $\boldsymbol{\epsilon}_k \in \mathbb{R}^{NC}$ if modelling output noise. These implementation details are all designed to encourage diversity among baselearners, with the goal of approximating the NTKGP posterior predictive for our Bayesian deep ensembles. Appendix F details how to aggregate predictions from trained baselearners. In Algorithm 1, we outline our NTKGP-param method: `data_noise` adds observation noise to targets; `concat` denotes a concatenation operation; and `init`$(\cdot)$ will be standard parameterisation initialisation in the JAX library Neural Tangents [38] unless stated otherwise. As discussed by Pearce et al. [25], there is a choice between "anchoring"/regularising parameters towards their initialisation or an independently sampled parameter set when modelling observation noise. We anchor at initialisation as the linearised NN regime only holds local to parameter initialisation [23], and also this reduces the memory cost of sampling parameters sets. Appendix E details our NTKGP-fn method.

Algorithm 1 NTKGP-param ensemble

Require: Data $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, loss function $\mathcal{L}$, NN model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, Ensemble size $K \in \mathbb{N}$, noise procedure: `data_noise`, NN parameter initialisation scheme: `init`$(\cdot)$

for $k = 1, \ldots, K$ **do**

Form $\{\mathcal{X}_k, \mathcal{Y}_k\} = \texttt{data\_noise}(\mathcal{D})$

Initialise $\boldsymbol{\theta}_k \overset{d}{\sim} \texttt{init}(\cdot)$

Initialise $\tilde{\boldsymbol{\theta}}_k \overset{d}{\sim} \texttt{init}(\cdot)$ and denote $\tilde{\boldsymbol{\theta}}_k = \texttt{concat}(\{\tilde{\boldsymbol{\theta}}_k^{\leq L}, \tilde{\boldsymbol{\theta}}_k^{L+1}\})$

Set $\boldsymbol{\theta}^*_k = \texttt{concat}(\{\tilde{\boldsymbol{\theta}}_k^{\leq L}, \mathbf{0}_{p_{L+1}}\})$

Define $\delta(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_k)\boldsymbol{\theta}^*_k$

Define $\tilde{f}_k(\boldsymbol{x}, \boldsymbol{\theta}_t) = f(\boldsymbol{x}, \boldsymbol{\theta}_t) + \delta(\boldsymbol{x})$ and set $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_k$

Optimise $\mathcal{L}(\tilde{f}_k(\mathcal{X}_k, \boldsymbol{\theta}_t), \mathcal{Y}_k) + \frac{1}{2}\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_k\|^2_\Lambda$ for $\boldsymbol{\theta}_t$ to obtain $\hat{\boldsymbol{\theta}}_k$

end for

return ensemble $\{\tilde{f}_k(\cdot, \hat{\boldsymbol{\theta}}_k)\}_{k=1}^K$

3.5 Classification methodology

For classification, we follow recent works [23, 39, 40] which treat classification as a regression task with one-hot regression targets. In order to obtain probabilistic predictions, we temperature scale our trained ensemble predictions with cross-entropy loss on a held-out validation set, noting that Fong and Holmes [41] established a connection between marginal likelihood maximisation and cross-validation.

Because $\delta(\cdot)$ is untrainable in our NTKGP methods, it is important to match the scale of the NTK Θ to the scale of the one-hot targets in multi-class classification settings. One can do this either by introducing a scaling factor $\kappa > 0$ such that we scale either: 1) $\tilde{f} \leftarrow \frac{1}{\kappa}\tilde{f}$ so that $\Theta \leftarrow \frac{1}{\kappa^2}\Theta$, or 2) $e_c \leftarrow \kappa e_c$ where $e_c \in \mathbb{R}^C$ is the one-hot vector denoting class $c \leq C$. We choose option 2) for

[5] We use the *sigmoid* function, instead of *softplus* [11], to enforce positivity on $\sigma^2_{\boldsymbol{\theta}}(\cdot)$, because our data will be standardised.

our implementation, tuning κ on a small set of values chosen to match the second moments of the randomly initialised baselearners, in logit space, of each ensemble method on the training set. We found κ to be an important hyperparameter that can determine a trade-off between in-distribution and out-of-distribution performance: see Appendix H for further details.

4 Experiments

Due to limited space, Appendix I will contain all experimental details not discussed in this section.

Toy 1D regression task We begin with a toy 1D example $y = x\sin(x) + \epsilon$, using homoscedastic $\epsilon \overset{d}{\sim} \mathcal{N}(0, 0.1^2)$. We use a training set of 20 points partitioned into two clusters, in order to detail uncertainty on out-of-distribution test data. For each ensemble method, we use MLP baselearners with two hidden layers of width 512, and erf activation. The choice of erf activation means that both the NTK Θ and NNGP kernel $\mathcal{K}$ are analytically available [23, 42]. We compare ensemble methods to the analytic GP posterior using either Θ or $\mathcal{K}$ as prior covariance function using the Neural Tangents library [38].

Figure 1 compares the analytic NTKGP posterior predictive with the analytic NNGP posterior predictive, as well as three different ensemble methods: deep ensembles, RP-param and NTKGP-param. We plot 95% predictive confidence intervals, treating ensembles as one Gaussian predictive distribution with matched moments like Lakshminarayanan et al. [11]. As expected, both NTKGP-param and RP-param ensembles have similar predictive means to the analytic NTKGP posterior. Likewise, we see that only our NTKGP-param ensemble predictive variances match the analytic NTKGP posterior. As foreseen in Proposition 2, the analytic NNGP posterior and other ensemble methods make more confident predictions than the NTKGP posterior, which in this example results in overconfidence on out-of-distribution data.[6]

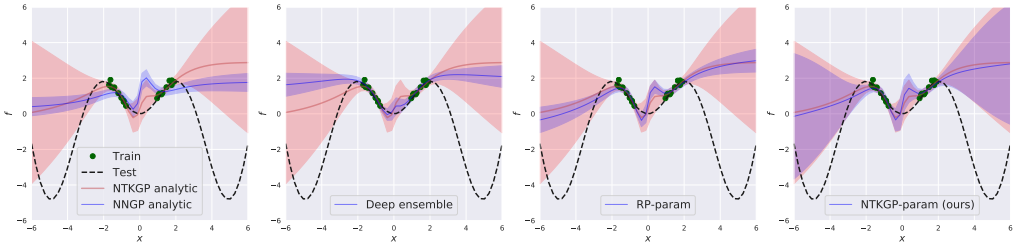


Figure 1: All subplots plot the analytic NTKGP posterior (in red). From left to right, (in blue): analytic NNGP posterior; deep ensembles; RP-param; and NTKGP-param (ours). For each method we plot the mean prediction and 95% predictive confidence interval. Green points denote the training data, and the black dotted line is the true test function $y = x\sin(x)$.

Flight Delays We now compare different ensemble methods on a large scale regression problem using the Flight Delays dataset [43], which is known to contain dataset shift. We train heteroscedastic baselearners on the first 700k data points and test on the next 100k test points at 5 different starting points: 700k, 2m (million), 3m, 4m and 5m. The dataset is ordered chronologically in date through the year 2008, so we expect the NTKGP methods to outperform standard deep ensembles for the later starting points. Figure 2 (Left) confirms our hypothesis. Interestingly, there seems to be a seasonal effect between the 3m and 4m test set that results in stronger performance in the 4m test set than the 3m test set, for ensembles trained on the first 700k data points. We see that our Bayesian deep ensembles perform slightly worse than standard deep ensembles when there is little or no test data shift, but fail more gracefully as the level of dataset shift increases.

Figure 2 (Right) plots confidence versus error for different ensemble methods on the combined test set of 5×100k points. For each precision threshold τ, we plot root-mean-squared error (RMSE) on examples where predictive precision is larger than τ, indicating confidence. As we can see, our NTKGP methods incur lower error over all precision thresholds, and this contrast in performance is magnified for more confident predictions.

MNIST vs NotMNIST We next move onto classification experiments, comparing ensembles trained on MNIST and tested on both MNIST and NotMNIST.[7] Our baselearners are MLPs with

[6] Code for this experiment is available at: https://github.com/bobby-he/bayesian-ntk.
[7] Available at http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html

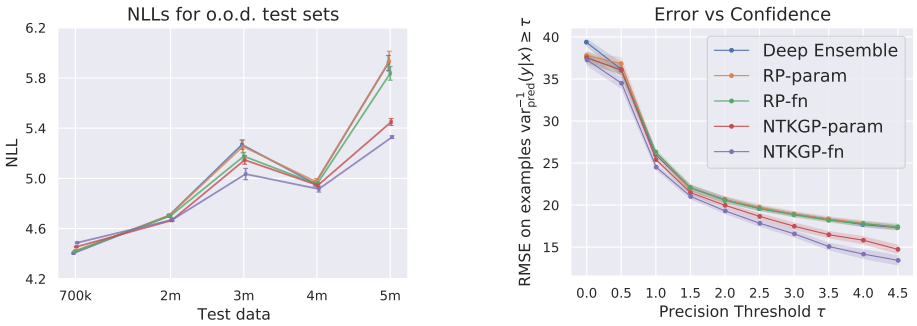


Figure 2: (Left) Flight Delays NLLs for ensemble methods trained on first 700k points of the dataset and tested on various out-of-distribution test sets, with time shift between training set and test set increasing along the x-axis. (Right) Error vs Confidence curves for ensembles tested on all 5×100k test points combined. Both plots include 95% CIs corresponding to 10 independent ensembles.

2-hidden layers, 200 hidden units per layer and ReLU activation. The weight parameter initialisation variance σ_W^2 is tuned using the validation accuracy on a small set of values around the He initialisation, σ_W^2=2, [44] for all classification experiments. Figure 3 shows both in-distribution and out-of-distribution performance across different ensemble methods. In Figure 3 (left), we see that our NTKGP methods suffer from slightly worse in-distribution test performance, with around 0.2% increased error for ensemble size 10. However, in Figure 3 (right), we plot error versus confidence on the combined MNIST and NotMNIST test sets: for each test point $(\boldsymbol{x}, y)$, we calculate the ensemble prediction $p(y = k|\boldsymbol{x})$ and define the predicted label as $\hat{y} = \text{argmax}_k p(y = k|\boldsymbol{x})$, with confidence $p(y = \hat{y}|\boldsymbol{x})$. Like Lakshminarayanan et al. [11], for each confidence threshold $0 \leq \tau \leq 1$, we plot the average error for all test points that are more confident than τ. We count all predictions on the NotMNIST test set to be incorrect. We see in Figure 3 (right) that the NTKGP methods vastly outperform both deep ensembles and RP methods, obtaining over 15% lower error on test points that have confidence τ=0.6, compared to all baselines. This is because our methods correctly make much more conservative predictions on the out-of-distribution NotMNIST test set, as can be seen by Figure 4, which plots histograms of predictive entropies. Due to the simple MLP architecture and ReLU activation, we can compare ensemble methods to analytic NTKGP results in Figures 3 & 4, where we see a close match between the NTKGP ensemble methods (at larger ensemble sizes) and the analytic predictions, both on in-distribution and out-of-distribution performance.

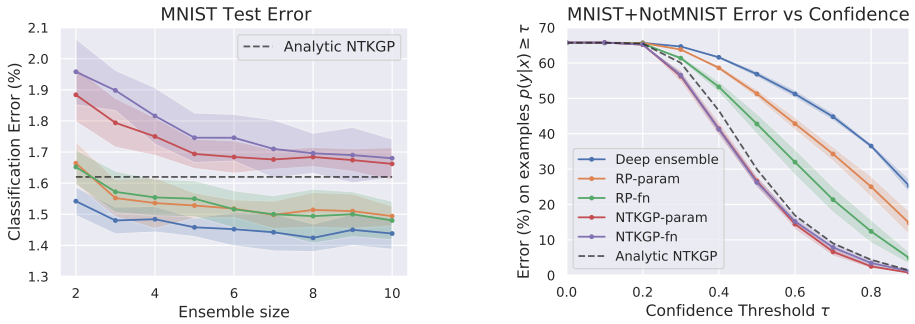


Figure 3: (Left) Classification error on MNIST test set for different ensemble sizes. (Right) Error versus Confidence plots for ensembles, of size 10, trained on MNIST and tested on both MNIST and NotMNIST. CIs correspond to 5 independent runs.

CIFAR-10 vs SVHN Finally, we present results on a larger-scale image classification task: ensembles are trained on CIFAR-10 and tested on both CIFAR-10 and SVHN. We conduct the same setup as for the MNIST vs NotMNIST experiment, with baselearners taking the Myrtle-10 CNN architecture [40] of channel-width 100. Figure 5 compares in distribution and out-of-distribution performance: we see that our NTKGP methods and RP-fn perform best on in-distribution test error. Unlike on the simpler MNIST task, there is no clear difference on the corresponding error versus confidence plot, and this is also reflected in the entropy histograms, which can be found in Figure 8 of Appendix I.

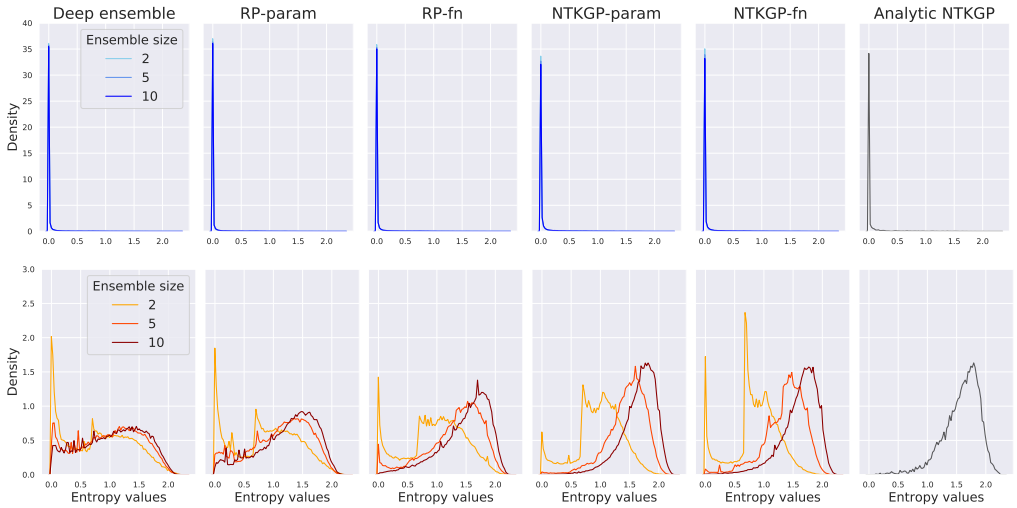


Figure 4: Histograms of predictive entropy on MNIST (top) and NotMNIST (bottom) test sets for different ensemble methods of different ensemble sizes, and also for Analytic NTKGP.

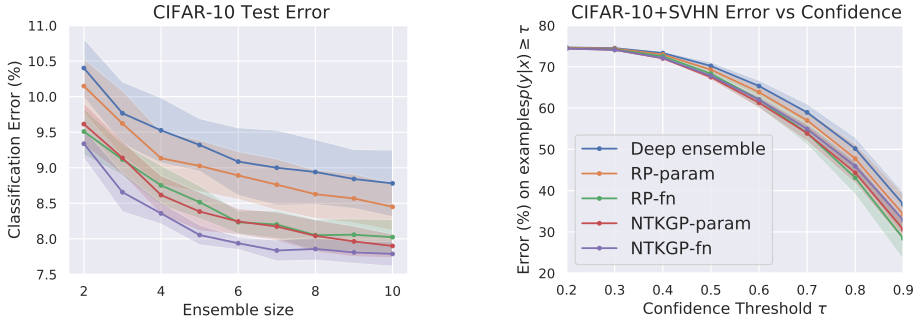


Figure 5: (Left) Classification error on CIFAR-10 test set for different ensemble sizes. (Right) Error versus Confidence plots of ensembles trained on CIFAR-10 and tested on both CIFAR-10 and SVHN. CIs correspond to 5 independent runs.

5 Discussion

We built on existing work regarding the Neural Tangent Kernel (NTK), which showed that there is no posterior predictive interpretation to a standard deep ensemble in the infinite width limit. We introduced a simple modification to training that enables a GP posterior predictive interpretation for a wide ensemble, and showed empirically that our Bayesian deep ensembles emulate the analytic posterior predictive when it is available. In addition, we demonstrated that our Bayesian deep ensembles often outperform standard deep ensembles in out-of-distribution settings for both regression and classification tasks.

In terms of limitations, our methods may perform worse than standard deep ensembles [11] when confident predictions are not detrimental, though this can be alleviated via NTK hyperparameter tuning. Moreover, our analyses are planted in the "lazy learning" regime [45, 46], and we have not considered finite-width corrections to the NTK during training [47–49]. In spite of these limitations, the search for a Bayesian interpretation to deep ensembles [11] is of particular relevance to the Bayesian deep learning community, and we believe our contributions provide useful new insights to resolving this problem by examining the limit of infinite-width.

A natural question that emerges from our work is how to tune hyperparameters of the NTK to best capture inductive biases or prior beliefs about the data. Possible lines of enquiry include: the large-depth limit [50], the choice of architecture [51], and the choice of activation [52]. Finally, we would like to assess our Bayesian deep ensembles in non-supervised learning settings, such as active learning or reinforcement learning.

Broader Impact

We believe that our Bayesian deep ensembles may be useful in situations where predictions that are robust to model misspecification and dataset shift are crucial, such as weather forecasting or medical diagnosis.

Acknowledgments and Disclosure of Funding

We thank Arnaud Doucet, Edwin Fong, Michael Hutchinson, Lewis Smith, Jasper Snoek, Jascha Sohl-Dickstein and Sheheryar Zaidi, as well as the anonymous reviewers, for helpful discussions and feedback. We also thank the JAX and Neural Tangents teams for their open-source software. BH is supported by the EPSRC and MRC through the OxWaSP CDT programme (EP/L016710/1).

References

[1] James Aitchison. Goodness of Prediction Fit. *Biometrika*, 62(3):547–554, 1975.

[2] David JC MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1992.

[3] Radford M Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media, 2012.

[4] Max Welling and Yee W Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

[5] Alex Graves. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[6] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

[7] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org, 2017.

[8] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches. *arXiv preprint arXiv:1803.04386*, 2018.

[9] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional Variational Bayesian Neural Networks. *arXiv preprint arXiv:1903.05779*, 2019.

[10] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[11] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[12] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *NeurIPS*, 2019.

[13] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep Ensembles: A Loss Landscape Perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[14] Andrew Gordon Wilson and Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. *arXiv preprint arXiv:2002.08791*, 2020.

[15] Radford M Neal. Priors for Infinite Networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

[16] Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep Neural Networks as Gaussian Processes. In *International Conference on Learning Representations*, 2018.

[17] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian Process Behaviour in Wide Deep Neural Networks. In *International Conference on Learning Representations*, volume 4, 2018.

[18] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep Convolutional Networks as shallow Gaussian Processes. In *International Conference on Learning Representations*, 2019.

[19] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes. In *International Conference on Learning Representations*, 2019.

[20] Greg Yang. Tensor Programs I: Wide Feedforward or Recurrent Neural Networks of Any Architecture are Gaussian Processes. *arXiv preprint arXiv:1910.12478*, 2019.

[21] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: NNGP and NTK for deep attention networks. *arXiv preprint arXiv:2006.10540*, 2020.

[22] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, pages 8571–8580, 2018.

[23] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models under gradient descent. In *Advances in Neural Information Processing Systems*, pages 8570–8581, 2019.

[24] Ian Osband, John Aslanides, and Albin Cassirer. Randomized Prior Functions for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

[25] Tim Pearce, Mohamed Zaki, Alexandra Brintrup, Nicolas Anastassacos, and Andy Neely. Uncertainty in Neural Networks: Bayesian Ensembling. *arXiv preprint arXiv:1810.05546*, 2018.

[26] Kamil Ciosek, Vincent Fortuin, Ryota Tomioka, Katja Hofmann, and Richard Turner. Conservative Uncertainty Estimation By Fitting Prior Networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlahxHYDS.

[27] Yehuda Hoffman and Erez Ribak. Constrained Realizations of Gaussian Fields: A Simple Algorithm. *The Astrophysical Journal*, 380:L5–L8, 1991.

[28] Greg Yang. Scaling Limits of Wide Neural Networks with Weight Sharing: Gaussian Process Behavior, Gradient Independence, and Neural Tangent Kernel Derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[29] Greg Yang. Tensor Programs II: Neural Tangent kernel for Any Architecture. *arXiv preprint arXiv:2006.14548*, 2020.

[30] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1032–1041, 2019.

[31] Daniel Park, Jascha Sohl-Dickstein, Quoc Le, and Samuel Smith. The Effect of Network Width on Stochastic Gradient Descent and Generalization: an Empirical Study. In *International Conference on Machine Learning*, pages 5042–5051, 2019.

[32] Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020.

[33] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies. In *Advances in Neural Information Processing Systems*, pages 4761–4771, 2019.

[34] Alexander G de G Matthews, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Sample-then-optimize posterior sampling for Bayesian linear models. In *NeurIPS Workshop on Advances in Approximate Bayesian Inference*, 2017.

[35] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[36] Carl Edward Rasmussen. Gaussian Processes in Machine Learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[37] Wei Hu, Zhiyuan Li, and Dingli Yu. Simple and Effective Regularization Methods for Training on Noisily Labeled Data with Generalization Guarantee. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hke3gyHYwH.

[38] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural Tangents: Fast and Easy Infinite Neural Networks in Python. In *International Conference on Learning Representations*, 2020. URL https://github.com/google/neural-tangents.

[39] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On Exact Computation with an Infinitely Wide Neural Net. In *Advances in Neural Information Processing Systems*, pages 8139–8148, 2019.

[40] Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Ludwig Schmidt, Jonathan Ragan-Kelley, and Benjamin Recht. Neural Kernels Without Tangents. *arXiv preprint arXiv:2003.02237*, 2020.

[41] E Fong and C C Holmes. On the marginal likelihood and cross-validation. *Biometrika*, 107(2): 489–496, 01 2020. ISSN 0006-3444. doi: 10.1093/biomet/asz077.

[42] Christopher KI Williams. Computing with Infinite Networks. In *Advances in neural information processing systems*, pages 295–301, 1997.

[43] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian Processes for Big Data. In *Uncertainty in Artificial Intelligence*, page 282. Citeseer, 2013.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[45] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947, 2019.

[46] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and Rich Regimes in Overparametrized Models. *arXiv preprint arXiv:2002.09277*, 2020.

[47] Ethan Dyer and Guy Gur-Ari. Asymptotics of Wide Networks from Feynman Diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

[48] Jiaoyang Huang and Horng-Tzer Yau. Dynamics of Deep Neural Networks and Neural Tangent Hierarchy. *arXiv preprint arXiv:1909.08156*, 2019.

[49] Boris Hanin and Mihai Nica. Finite Depth and Width Corrections to the Neural Tangent Kernel. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgndT4KwB.

[50] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. Mean-field Behaviour of Neural Tangent Kernel for Deep Neural Networks. *arXiv preprint arXiv:1905.13654*, 2019.

[51] Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris Holmes, Frank Hutter, and Yee Whye Teh. Neural Ensemble Search for Performant and Calibrated Predictions. *arXiv preprint arXiv:2006.08573*, 2020.

[52] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *arXiv preprint arXiv:2006.10739*, 2020.

[53] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient BackProp. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[54] David Williams. *Probability with Martingales*. Cambridge University Press, 1991. doi: 10.1017/CBO9780511813658.

[55] Patrick Billingsley. *Probability and Measure*. John Wiley and Sons, second edition, 1986.

[56] Linh Tran, Bastiaan S Veeling, Kevin Roth, Jakub Swiatkowski, Joshua V Dillon, Jasper Snoek, Stephan Mandt, Tim Salimans, Sebastian Nowozin, and Rodolphe Jenatton. Hydra: Preserving Ensemble Diversity for Model Distillation. *arXiv preprint arXiv:2001.04694*, 2020.

[57] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The Reversible Residual Network: Backpropagation Without Storing Activations. In *Advances in Neural Information Processing Systems 30*, pages 2214–2224, 2017.

[58] Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty Estimation Using a Single Deep Deterministic Neural Network. In *International Conference on Machine Learning*, 2020.

[59] Michael W Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-an Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors. *arXiv preprint arXiv:2005.07186*, 2020.

[60] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Recap of standard and NTK parameterisations

For completeness, we recap the difference between standard and NTK parameterisations & initialisations [22, 23] for an MLP in this section.

Consider an MLP with L hidden layers of widths from $n_0{=}d$ to n_L respectively, and final readout layer with $n_{L+1} = C$. For a given $\boldsymbol{x} \in \mathbb{R}^d$, under the *NTK* parameterisation the recurrence relation that constitutes the forward pass of the NN is then:

$$\alpha^{(0)}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{x} \tag{13}$$

$$\tilde{\alpha}^{(l+1)}(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\sigma_W}{\sqrt{n_l}} W^{(l)} \alpha^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}) + \sigma_b b^{(l)} \tag{14}$$

$$\alpha^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}) = \phi(\tilde{\alpha}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta})) \tag{15}$$

for $l \leq L$ where $\tilde{\alpha}^{(l)}$ and $\alpha^{(l)}$ are the preactivations and activations respectively at layer l, with entrywise nonlinearity $\phi(\cdot)$. In the NTK parameteriation, all parameters $W^{(l)} \in \mathbb{R}^{n_{l+1} \times n_l}$ and $b^{(l)} \in \mathbb{R}^{n_{l+1}}$ for all layers l are initialised as i.i.d. standard normal $\mathcal{N}(0, 1)$. The hyperparameters σ_W and σ_b are known as the weight and bias variances respectively, and are hyperparameters of the infinite width limit NTK Θ.

On the other hand, under *standard* parameterisation, the recurrence relation of the NN is:

$$\alpha^{(0)}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{x} \tag{16}$$

$$\tilde{\alpha}^{(l+1)}(\boldsymbol{x}, \boldsymbol{\theta}) = W^{(l)} \alpha^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}) + b^{(l)} \tag{17}$$

$$\alpha^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}) = \phi(\tilde{\alpha}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta})) \tag{18}$$

with $W^{(l)}_{i,j} \sim \mathcal{N}(0, \frac{1}{n_l}\sigma_W^2)$ and $b^{(l)}_j \sim \mathcal{N}(0, \sigma_b^2)$ at initialisation. Commonly used initialisation schemes like LeCun [53] or He [44] fall into this category.

Regardless of parameterisation, our notation from Sections 2 & 3 corresponds to $f(\boldsymbol{x}, \boldsymbol{\theta}) = \tilde{\alpha}^{(L+1)}(\boldsymbol{x}, \boldsymbol{\theta})$, with $\boldsymbol{\theta} = \{W^{(l)}, b^{(l)}\}_{l=0}^{L}$, $\boldsymbol{\theta}^{\leq L} = \{W^{(l)}, b^{(l)}\}_{l=0}^{L-1}$ and $\boldsymbol{\theta}^{L+1} = \{W^{(L)}, b^{(L)}\}$.

We see that the different parameterisations yield the same distribution for the functional output $f(\cdot, \boldsymbol{\theta})$ at initialisation, but give different scalings to the parameter gradients in the backward pass. Sohl-Dickstein et al. [32] have recently explored further variants of these parameterisations.

B Proofs

B.1 Proof of Proposition 1

Proposition 1. *$\delta(\cdot) \xrightarrow{d} \mathcal{GP}(0, \Theta^{\leq L})$ and is independent of $f_0(\cdot)$ in the infinite width limit. Thus, $\tilde{f}_0(\cdot) = f_0(\cdot) + \delta(\cdot) \xrightarrow{d} \mathcal{GP}(0, \Theta)$.*

Proof. For notational ease, let us define two jointly independent GPs $g(\cdot) \overset{d}{\sim} \mathcal{GP}(0, \Theta^{\leq L})$ & $h(\cdot) \overset{d}{\sim} \mathcal{GP}(0, \mathcal{K})$. By independence, we have $g(\cdot) + h(\cdot) \overset{d}{\sim} \mathcal{GP}(0, \Theta)$. Moreover, let $\delta_m(\cdot)$, $f_m^0(\cdot)$ and $\boldsymbol{\theta}_m^0$ denote $\delta(\cdot)$, $f_0(\cdot)$ and $\boldsymbol{\theta}_0$ respectively at width parameter $m \in \mathbb{N}$. The infinite width limit thus corresponds to $m \to \infty$.

For our purposes, it will be sufficient to prove convergence of finite-dimensional marginals, $(\delta_m(\mathcal{X}), f_m^0(\mathcal{X}')) \xrightarrow{d} (g(\mathcal{X}), h(\mathcal{X}'))$ jointly, for arbitrary sets of inputs $\mathcal{X}, \mathcal{X}'$. Note that previous work [16, 17] has already shown that $f_m^0(\mathcal{X}') \xrightarrow{d} h(\mathcal{X}')$.

The proof that $(\delta_m(\mathcal{X}), f_m^0(\mathcal{X}')) \xrightarrow{d} (g(\mathcal{X}), h(\mathcal{X}'))$ relies on Lévy's Convergence theorem [54] and the Cramér-Wold device (Theorem 29.4 of [55]). Using these results it is sufficient to show, denoting φ_X as the characteristic function of a random variable X, that:

$$\varphi_{Y_m}(t) \to \varphi_Y(t) \tag{19}$$

where $Y_m = u^\top \delta_m(\mathcal{X}) + v^\top f_m^0(\mathcal{X}')$ and $Y = u^\top g(\mathcal{X}) + v^\top h(\mathcal{X}')$, for all $t \in \mathbb{R}, u \in \mathbb{R}^{|\mathcal{X}|C}$ and $v \in \mathbb{R}^{|\mathcal{X}'|C}$. But:

$$\begin{aligned}
\varphi_{Y_m}(t) =& \mathbb{E}\big[\exp(itY_m)\big] && (20)\\
=& \mathbb{E}\Big[\mathbb{E}\big[\exp(itY_m) \,\big|\, \boldsymbol{\theta}_{0,m}\big]\Big] && (21)\\
=& \mathbb{E}_{\boldsymbol{\theta}_{0,m}}\big[\exp\big(-t^2 u^\top \hat{\Theta}_{0,m}^{\leq L}(\mathcal{X},\mathcal{X})u + itv^\top f_m^0(\mathcal{X}')\big)\big] && (22)\\
=& \exp\big(-t^2 u^\top \Theta^{\leq L}(\mathcal{X},\mathcal{X})u\big)\mathbb{E}_{\boldsymbol{\theta}_{0,m}}\big[\exp\big(itv^\top f_m^0(\mathcal{X}')\big)\big] + r_m && (23)\\
\to& \mathbb{E}\big[\exp(itY)\big] && (24)
\end{aligned}$$

where r_m, defined as the difference between Eqs. (23) & (22), can be shown to be $o_m(1)$ using the Bounded Convergence theorem and the empirical NTK convergence results, and by noting that proofs of NTK convergence [22, 23, 28, 29] are all done on a layer-by-layer basis.

The claim that $\tilde{f}_0(\cdot) = f_0(\cdot) + \delta(\cdot) \xrightarrow{d} \mathcal{GP}(0, \Theta)$ then follows by setting $\mathcal{X} = \mathcal{X}'$ and $v = u$.

□

B.2 Proof of Proposition 2

Proposition 2. *For σ^2=0, $\Sigma_{NTKGP} \succeq \Sigma_{DE} \succeq \Sigma_{NNGP}$. Similarly, for σ^2>0, $\Sigma_{NTKGP} \succeq \Sigma_{RP} \succeq \Sigma_{NNGP}$.*

Proof. We will prove the case for $\sigma^2 > 0$ as the case for $\sigma^2 = 0$ is similar, and one can replace inversions of $\Theta(\mathcal{X},\mathcal{X})$ and $\mathcal{K}(\mathcal{X},\mathcal{X})$ with generalised inverses if need be.

Let $\mathcal{X}'$ be an arbitrary test set. We will first show $\Sigma_{\text{RP}} \succeq \Sigma_{\text{NNGP}}$. It will suffice to show that $\Sigma_{\text{RP}}(\mathcal{X}',\mathcal{X}') - \Sigma_{\text{NNGP}}(\mathcal{X}',\mathcal{X}') \succeq 0$ is a p.s.d. matrix. But it is not hard to check that:

$$\Sigma_{\text{RP}}(\mathcal{X}',\mathcal{X}') - \Sigma_{\text{NNGP}}(\mathcal{X}',\mathcal{X}') = U(\mathcal{K}(\mathcal{X},\mathcal{X}) + \sigma^2 I)U^\top \tag{25}$$

which is clearly p.s.d, where $U = \Theta(\mathcal{X}',\mathcal{X})(\Theta(\mathcal{X},\mathcal{X}) + \sigma^2 I)^{-1} - \mathcal{K}(\mathcal{X}',\mathcal{X})(\mathcal{K}(\mathcal{X},\mathcal{X}) + \sigma^2 I)^{-1} \in \mathbb{R}^{|\mathcal{X}'|\times|\mathcal{X}|}$

Likewise, to show $\Sigma_{\text{NTK}} \succeq \Sigma_{\text{RP}}$ we can check that:

$$\Sigma_{\text{NTK}}(\mathcal{X}',\mathcal{X}') - \Sigma_{\text{RP}}(\mathcal{X}',\mathcal{X}') = U_1 + U_2\Delta(\mathcal{X},\mathcal{X})U_2^\top \succeq 0 \tag{26}$$

where

$$U_1 = \Delta(\mathcal{X}',\mathcal{X}') - \Delta(\mathcal{X}',\mathcal{X})\Delta^g(\mathcal{X},\mathcal{X})\Delta(\mathcal{X},\mathcal{X}') \tag{27}$$

and $\Delta = \Theta^{\leq L} \succeq 0$ is the contributions to the NTK from parameters before the final layer as before. Finally, we need to define U_2 as:

$$U_2 = \Theta(\mathcal{X}',\mathcal{X})(\Theta(\mathcal{X},\mathcal{X}) + \sigma^2 I)^{-1} - \Delta(\mathcal{X}',\mathcal{X})\Delta^g(\mathcal{X},\mathcal{X}) \tag{28}$$

The notation $\Delta^g(\mathcal{X},\mathcal{X})$ denotes the generalised inverse. $U_1 \succeq 0$ follows from standard properties of generalised Schur complements, as does the fact that $\Delta(\mathcal{X}',\mathcal{X})\Delta^g(\mathcal{X},\mathcal{X})\Delta(\mathcal{X},\mathcal{X}) = \Delta(\mathcal{X}',\mathcal{X})$, which is required for Eq. (26) to hold.

□

C Alternative constructions of NTKGP baselearners

To summarise the analysis in Section 3, the criteria for an NTKGP baselearner $\tilde{f}(\cdot,\boldsymbol{\theta})$ is that:
1) $\tilde{f}(\cdot,\boldsymbol{\theta}_0) \xrightarrow{d} \mathcal{GP}(0,\Theta)$ as width increases, while 2) preserving the initial Jacobian $\nabla_{\boldsymbol{\theta}} f_0(\cdot) = \nabla_{\boldsymbol{\theta}} \tilde{f}_0(\cdot)$.

A possible alternative construction would be if one could (approximately) sample a fixed $f^* \overset{d}{\sim} \mathcal{GP}(0,\Theta)$, and set:

$$\tilde{f}_t(\cdot) = f_t(\cdot) + f^*(\cdot) - f_0(\cdot) \tag{29}$$

It is easy to approximately sample f^* for finite width NNs using a single JVP, under either standard or NTK parameterisation, by sampling $\tilde{\boldsymbol{\theta}}$ independent of $\boldsymbol{\theta}_0$ and setting:

$$f^*(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\tilde{\boldsymbol{\theta}} \tag{30}$$

Note that Eq. (29) requires computation of two forward passes f_t and f_0 in addition to a JVP $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\tilde{\boldsymbol{\theta}}$. For some implementations of JVPs, such as in JAX [35], the computation of f_0 will come essentially for free alongside the computation of $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)\tilde{\boldsymbol{\theta}}$, because the JVP is centered about the same "primal" parameters $\boldsymbol{\theta}_0$ that are used for f_0. Hence, this alternative $\tilde{f}$ presented in Eq. (29) may have similar costs to our main construction in Section 3, for certain AD packages.

A second valid alternative to $\tilde{f}_t$ would be to replace f_t with f_t^{lin}, which would give $\tilde{f}^{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \tilde{\boldsymbol{\theta}})\boldsymbol{\theta}_t$ (where we swap $\tilde{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}_0$ for notational consistency with other NTKGP methods, and initialise at $\boldsymbol{\theta}_0$). Because $\tilde{\boldsymbol{\theta}}$ is fixed, we see that $\tilde{f}^{\text{lin}}(\cdot, \boldsymbol{\theta}_t)$ is linear in $\boldsymbol{\theta}_t$. This gives a realisation of the "sample-then-optimize" approach [34] to give posterior samples from randomly initialised linear models, and ensures that $\tilde{f}_\infty^{\text{lin}}(\cdot)$ is an exact posterior sample (using the empirical NTK $\hat{\Theta}_0$ as prior kernel) irrespective of parameterisation or width. Note though, of course, the linearised regime holds for $\tilde{f}_t^{\text{lin}}$ throughout parameter space, hence for strongly convex optimisation problems like regression tasks with observation noise, the initialisation is irrelevant. We will call $\tilde{f}_\infty^{\text{lin}}$ trained in such a way an *NTKGP-Lin* baselearner.

D Regularisation in the NTKGP and RP training procedures

As stated in Lemma 3 of Osband et al. [24], suppose we are in a Bayesian linear regression setting with linear map $g_{\boldsymbol{\theta}}(\boldsymbol{z}) = \boldsymbol{z}^\top \boldsymbol{\theta}$, model $y = g_{\boldsymbol{\theta}}(\boldsymbol{z}) + \epsilon$ for $\epsilon \sim \mathcal{N}(0, \sigma^2)$ i.i.d., and parameter prior $\boldsymbol{\theta} \sim N(0, \lambda I_p)$. Then, having observed training data $\{(\boldsymbol{z}_i, y_i)\}_{i=1}^n$, solving the following optimisation problem returns a posterior sample $\boldsymbol{\theta}$:

$$\tilde{\boldsymbol{\theta}} + \underset{\boldsymbol{\theta}}{\text{argmin}} \sum_{i=1}^{n} \frac{1}{2\sigma^2} \left\| \tilde{y}_i - (g_{\boldsymbol{\theta}} + g_{\tilde{\boldsymbol{\theta}}})(\boldsymbol{z}_i) \right\|_2^2 + \frac{1}{2\lambda} \|\boldsymbol{\theta}\|_2^2 \tag{31}$$

where $\tilde{y}_i \sim \mathcal{N}(y_i, \sigma^2)$ and $\tilde{\boldsymbol{\theta}} \sim \mathcal{N}(0, \lambda I_p)$.

We see that when there is a homoscedastic prior $\mathcal{N}(0, \lambda I_p)$ for $\boldsymbol{\theta}$ that the correct weighting of L^2 regularisation is $\|\boldsymbol{\theta}\|_\Lambda^2 = \frac{1}{\lambda}\boldsymbol{\theta}^\top\boldsymbol{\theta}$. In fact, even with a heteroscedastic prior $\boldsymbol{\theta} \sim \mathcal{N}(0, \Lambda)$ with a diagonal matrix $\Lambda \in \mathbb{R}_+^{p \times p}$ and diagonal entries $\{\lambda_j\}_{j=1}^p$, it is straightforward to show that the correct setting of regularisation is $\|\boldsymbol{\theta}\|_\Lambda^2 = \boldsymbol{\theta}^\top \Lambda^{-1} \boldsymbol{\theta}$ in order to obtain a posterior sample of $\boldsymbol{\theta}$. For RP-param or NTKGP-param methods, with initial parameters $\boldsymbol{\theta}_0$, we have regularisation $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_\Lambda^2 = (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \Lambda^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)$, which can be seen as a Mahalanobis distance.

For an NN in the linearised regime [23], this is related to the fact that the NTK and standard parameterisations initialise parameters differently, yet yield the same functional distribution for a randomly initialised NN. In the standard parameterisation, λ_j will be a factor of the NN width smaller than in the NTK parameterisation, but the corresponding feature map $\boldsymbol{z}$ will be a square root factor of the NN width larger. Thus, solving Eq. (31) will lead to the same functional outputs in both parameterisations, if the NN remains in the linearised regime. However, only with our NTKGP trained baselearners $\tilde{f}$ do you get a posterior interpretation to the trained NN because of the difference between the NNGP and the NTK that standard training does not account for, and because the linearised regime only holds locally to the parameter initialisation.

E Additional ensemble algorithms

Here, we present our ensemble algorithms for NTKGP-Lin (Algorithm 2) and NTKGP-fn (Algorithm 3), to complement the NTKGP-param algorithm that was presented in Section 3.4.

Algorithm 2 NTKGP-Lin ensemble

Require: Data $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, loss function $\mathcal{L}$, NN model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, Ensemble size $K \in \mathbb{N}$, noise procedure: `data_noise`, NN parameter initialisation scheme: `init`$(\cdot)$
for $k = 1, \ldots, K$ **do**
  Form $\{\mathcal{X}_k, \mathcal{Y}_k\} = \texttt{data\_noise}(\mathcal{D})$
  Initialise $\boldsymbol{\theta}_k \overset{d}{\sim} \texttt{init}(\cdot)$
  Initialise $\tilde{\boldsymbol{\theta}}_k \overset{d}{\sim} \texttt{init}(\cdot)$
  Define $\tilde{f}_k^{\text{lin}}(\boldsymbol{x}, \boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \tilde{\boldsymbol{\theta}}_k)\boldsymbol{\theta}_t$ and set $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_k$
  Optimise $\mathcal{L}(\tilde{f}_k^{\text{lin}}(\mathcal{X}_k, \boldsymbol{\theta}_t), \mathcal{Y}_k) + \frac{1}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_k\|_\Lambda^2$ for $\boldsymbol{\theta}_t$ to obtain $\hat{\boldsymbol{\theta}}_k$
end for
return ensemble $\{\tilde{f}_k^{\text{lin}}(\cdot, \hat{\boldsymbol{\theta}}_k)\}_{k=1}^K$

Algorithm 3 NTKGP-fn ensemble

Require: Data $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, loss function $\mathcal{L}$, NN model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, Ensemble size $K \in \mathbb{N}$, noise procedure: `data_noise`, NN parameter initialisation scheme: `init`$(\cdot)$
for $k = 1, \ldots, K$ **do**
  Form $\{\mathcal{X}_k, \mathcal{Y}_k\} = \texttt{data\_noise}(\mathcal{D})$
  Initialise $\boldsymbol{\theta}_k \overset{d}{\sim} \texttt{init}(\cdot)$
  Initialise $\tilde{\boldsymbol{\theta}}_k \overset{d}{\sim} \texttt{init}(\cdot)$ and denote $\tilde{\boldsymbol{\theta}}_k = \texttt{concat}(\{\tilde{\boldsymbol{\theta}}_k^{\leq L}, \tilde{\boldsymbol{\theta}}_k^{L+1}\})$
  Set $\boldsymbol{\theta}_k^* = \texttt{concat}(\{\sqrt{2}\tilde{\boldsymbol{\theta}}_k^{\leq L}, \tilde{\boldsymbol{\theta}}_k^{L+1}\})$
  Define $\delta(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \tilde{\boldsymbol{\theta}}_k)\boldsymbol{\theta}_k^*$
  Define $\tilde{f}_k(\boldsymbol{x}, \boldsymbol{\theta}_t) = f(\boldsymbol{x}, \boldsymbol{\theta}_t) + \delta(\boldsymbol{x})$ and set $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_k$
  Optimise $\mathcal{L}(\tilde{f}_k(\mathcal{X}_k, \boldsymbol{\theta}_t), \mathcal{Y}_k) + \frac{1}{2} \|\boldsymbol{\theta}_t\|_\Lambda^2$ for $\boldsymbol{\theta}_t$ to obtain $\hat{\boldsymbol{\theta}}_k$
end for
return ensemble $\{\tilde{f}_k(\cdot, \hat{\boldsymbol{\theta}}_k)\}_{k=1}^K$

In Algorithm 3 we seek to reinitialise $\tilde{f}_k(\boldsymbol{x}, \boldsymbol{\theta}_0)$ from $\mathcal{GP}(0, \mathcal{K})$ to $\mathcal{GP}(0, 2\Theta)$ in the infinite width limit, following the randomised prior function method of Osband et al. [24]. While there are many ways to do this we choose to use only one JVP, with a reweighted tangent vector, for $\delta(\cdot)$ in order to reduce extra computational costs. It would be similarly possible to model a scaling factor β for the prior function, like [24], using a single JVP with a differently reweighted tangent vector.

Note also that for the NTKGP-fn it is unreasonable to assume that the linearised NN dynamics will hold true for the duration of training because, unlike in NTKGP-param (Algorithm 1) we regularise towards the origin not the initialised parameters.

F Aggregating predictions from ensemble members

For completeness, we now describe how to aggregate predictions from ensemble members. Given a test point $(\boldsymbol{x}, y)$, for each baselearner NN $k \leq K$, we suppose we have a probabilistic prediction $p_k(y|\boldsymbol{x})$ obtained from the NN output. We then treat the ensemble as a uniformly-weighted mixture model over baselearners and combine predictions as $p(y|\boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^K p_k(y|\boldsymbol{x})$. For our Bayesian deep ensembles, we can view this aggregation as a Monte Carlo approximation of the GP posterior predictive with NTK prior.

For classification tasks, this aggregation is exactly an average of predicted probabilities. For regression tasks, the prediction is a mixture of normal distributions, and we follow Lakshminarayanan et al. [11] by approximating the ensembled prediction as a single Gaussian with matched moments. That is to say, if $p_k(y|\boldsymbol{x}) \sim \mathcal{N}(\mu_k(\boldsymbol{x}), \sigma_k^2(\boldsymbol{x}))$, then we approximate $p(y|\boldsymbol{x})$ by $\mathcal{N}(\mu_*(\boldsymbol{x}), \sigma_*^2(\boldsymbol{x}))$ for $\mu_*(\boldsymbol{x}) = \frac{1}{K}\sum_k \mu_k(\boldsymbol{x})$ and $\sigma_*^2(\boldsymbol{x}) = \frac{1}{K}\sum_k(\mu_k^2(\boldsymbol{x}) - \mu_*^2(\boldsymbol{x})) + \sigma_k^2(\boldsymbol{x})$.

G Comparison of memory and computation costs for ensemble methods

There is only a negligible training-time computational overhead for our NTKGP methods compared to other ensemble methods [11, 24], for a training set of fixed size (e.g. MNIST, CIFAR-10). This

is because one can obtain and store our fixed additive JVPs δ in a single pass over the training data. For test-time constrained applications, one can employ ensemble distillation [56] for our NTKGP ensembles as one would for standard deep ensembles.

For completeness, we include in Table 2 (left) the computational cost of different ensemble methods when the modified forward pass $\tilde{f}$ needs to be computed on the fly for new data, though we again stress that this is not necessary for train nor test time, as described in the paragraph above. A rule of thumb for a library offering forward-mode AD, like JAX [35], is that a JVP costs on the order of three standard forward passes in terms of FLOPs. We use forward-mode AD to compute JVPs as this is known to be more memory-efficient than reverse-mode AD for JVP computation. It is worth pointing out that our methods share the same trainable parameters as standard deep ensembles, and so do not incur any additional computational cost in the backward pass.

Table 2: Comparison of computational and memory costs of different ensemble methods per ensemble member. Computational costs are specified per (modified) forward pass and represent a naive worst-case scenario (presented for completeness); a more astute approach renders only a negligible difference between ensemble methods, as discussed in this section.

Method	Computational cost		Parameter sets to store	
	Forward passes	JVPs	Train time	Test time
Deep ensembles	1	0	1	1
RP-param	1	0	2	1
RP-fn	2	0	2	2
NTKGP-param	1	1	3	3
NTKGP-fn	1	1	3	3

In terms of memory, both NTKGP and RP methods require storage of extra sets of parameters in order to compute the untrainable additive functions $\delta(\cdot)$ and regularise in parameter space, displayed in Table 2 (right). However, the activations of the extra forward pass in the Randomised prior function method need not be stored. And moreover, forward mode JVPs are composed alongside the primitive operations that comprise the forward pass, so the memory requirements incurred by the extra JVP are independent of the NN depth for our NTKGP methods. Note that the memory bottleneck for large NNs is most often from the need to store activations for the backward pass [57] and not from storing parameter sets, hence our NTKGP ensembles are not affected by the main memory bottleneck for large NNs, relative to standard deep ensembles.

It is worth noting that our Bayesian deep ensembles still retain the distributability of standard deep ensembles. Moreover, our computational and memory costs still scale linearly in dataset size and parameter space dimension, enabling us to work with large scale datasets like Flight Delays [43].

Finally, in this section we only compare the costs associated to different ensemble methods. Ensembles methods are known to be computationally expensive and there has been recent interest in the community to derive new methods [58, 59] that reduce such costs. However, at the time of writing, deep ensembles [11] are state-of-the-art for uncertainty quantification tasks [12], and hence we believe a comparison of costs between ensemble methods is most appropriate for this work.

H Scaling for one-hot targets in classification

As discussed in Section 3.5 and repeated here for completeness: because $\delta(\cdot)$ is untrainable in our NTKGP methods, it is important to match the scale of the NTK Θ to the scale of the one-hot targets in multi-class classification settings. One can do this either by introducing a scaling factor $\kappa > 0$ such that we scale either: 1) $\tilde{f} \leftarrow \frac{1}{\kappa}\tilde{f}$ and so $\Theta \leftarrow \frac{1}{\kappa^2}\Theta$, or 2) $e_c \leftarrow \kappa e_c$ where $e_c \in \mathbb{R}^C$ is the one-hot vector denoting class $c \leq C$. We choose option 2) for our implementation.

To set κ, for each ensemble method we calculated the mean squared values of baselearner outputs at initialisation, which we define for convenience as ζ_0, on the training set for that particular ensemble method, and tuned κ^2 (based on validation accuracy) on a small linear scale centered around $C\zeta_0$, where C is the number of classes. This is in order to match the second moments of the random NNs at initialisation with the scaled one-hot targets across the C classes. For example, for NTKGP-param, we set $\zeta_0 = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}}\Theta(\boldsymbol{x},\boldsymbol{x}) \in \mathbb{R}^+$.

To illustrate the importance of κ, in Figure 6 we present the corresponding results to Figure 5 where instead of setting κ dependent on the scale of each ensemble methods' initialised baselearners, as above, we set $\kappa = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}}\Theta(\boldsymbol{x},\boldsymbol{x})\in\mathbb{R}$ for all ensemble methods. This is the base κ value for NTKGP-param at initialisation, but note that we did not tune neither κ (around this base value) nor weight variance (set at $\sigma_W^2 = 2$ like He initialisation [44], which has been optimised for standard NNs and hence standard deep ensembles) for Figure 6.

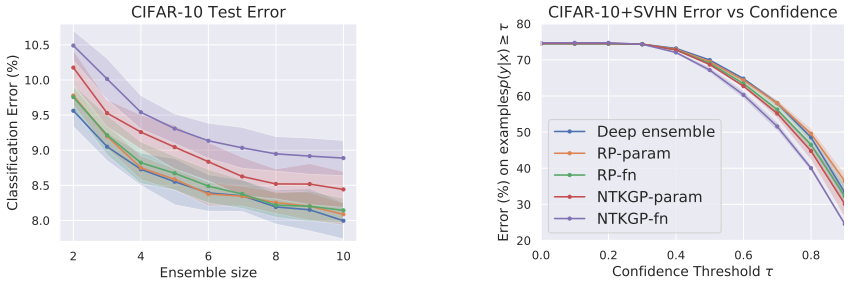


Figure 6: Figure 5 but where regression target scale κ is constant across ensemble methods and set to match the second moment of the NTK on the training set at initialisation. Error bars correspond to 5 independent runs.

In Figure 6 we see a different results to Figure 5, as here our NTKGP methods suffer slightly on in-distribution performance but also outperform the baselines methods on out-of-distribution detection. This highlights the importance of the regression target scale when considering classification tasks, and moreover reflects a general theme in our experiments of the trade-off between more aggressive predictions (that tend to perform better on in-distribution) and more conservative predictions (that tend to perform better on out-of-distribution). In our classification methodology, larger κ values tend to lead to more confident predictions. We point out that this is an issue that affects all ensemble methods and is not limited to our Bayesian ensembles.

I Experimental Details & additional plots

I.1 Toy 1d example

We set ensemble size $K = 20$, and train on full batch GD with learning rate 0.001 for 50,000 iterations under standard parameterisation in Neural Tangents [38], with $\sigma_W = 1.5$ & $\sigma_b = 0.05$, for σ_W, σ_b defined as in Appendix A. In Figure 7 we evaluate the impact of the ensemble size on this toy problem for different ensemble methods. We find that, of the two methods that approximate the analytic NTKGP mean predictor (c.f. Table 1), the approximation of the analytic mean predictor for NTKGP-param degrades compared to RP-param at small ensemble sizes, although the predictive uncertainties are well matched even at small ensemble sizes. The degradation in mean predictor is unsurprising as there is more (untrainable) noise in the initialised NTKGP baselearners. One simple possible solution to this problem, which we leave for future work, is to use separate baselearners for the mean and uncertainty predictions, like in Ciosek et al. [26].

I.2 Flight Delays

Our baselearners are MLPs with 4 hidden layers, 100 hidden units per layer and ReLU activations, and we use standard parameterisation with $\sigma_W = 1$ & $\sigma_b = 0.05$, and choose ensemble size $K = 5$. We train for 10 epochs with learning rate 0.001, batch size 100 and Adam [60]. For all experiments, all ensemble methods apart from standard deep ensembles [11] are L^2 regularised according to Appendix D, with weight decay strength set to 10^{-4} for standard deep ensembles.

We use a validation set of size 50k that is sampled uniformly from the training set of size 700k, and early stop baselearner NNs based on validation set loss. Inputs and targets are standardised so that the training data is zero mean and unit variance.

I.3 MNIST vs. NotMNIST

For all image classification experiments, we use a $90-10\%$ split for the train-validation sets needed for temperature scaling.

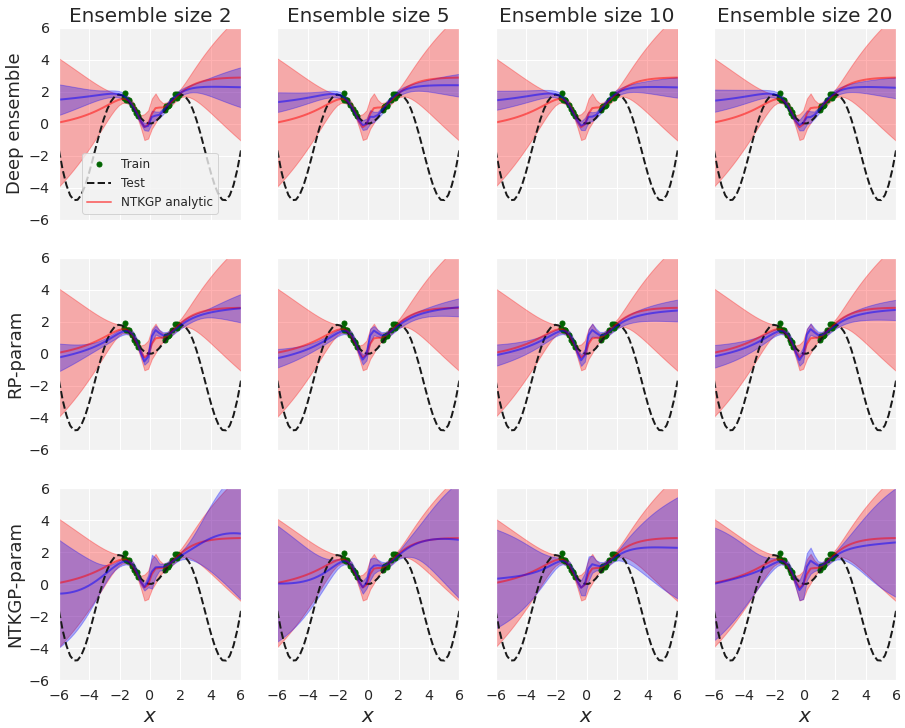


Figure 7: Comparison between ensemble methods (blue) and the analytic NTKGP posterior as the ensemble size is varied on toy example.

Baselearners are MLPs with 2-hidden layers, 200 hidden units per layer and ReLU activations. We standardise data to have mean 0 and standard deviation 1 across flattened pixels.

For all ensemble methods, we use standard parameterisation with fixed bias standard deviation $\sigma_b = 0.05$, observation noise $\sigma = 0.1$ and tune weight variance σ_W^2 on a small linear scale around $\sigma_W^2 = 2$. We set observation noise $\sigma = 0.1$ for . We train for 20 epochs with batch size 100, learning rate 0.001 and Adam [60]. We do not early stop for any classification experiment, and use the final trained baselearners throughout.

For the analytic NTKGP results, we use the NTK in NTK parameterisation, and use the same observation noise and bias variance as for ensemble methods. However, we fix $\sigma_W^2 = 2$ and also do not tune target scale κ (set to the base value described in Appendix A) due to computational resources. We also use only half the test sets both for MNIST and NotMNIST due to resource requirements, keeping the ratios of test sizes consistent in order for the error versus confidence plot Figure 3 (right) to be comparable. To compute test and out-of-distribution predictions, having obtained the optimal temperature scale T^* and analytic NTKGP predictions in logit space, $p(\cdot|\mathcal{X}, \mathcal{Y})$, we approximate the softmax class probability predictions: $\int \text{softmax}(z/T^*)p(dz|\mathcal{X}, \mathcal{Y})$, by a Monte Carlo ensemble approximation with 100 samples.

For all classification ensemble methods, we temperature scale on validation cross entropy for 5 epochs with batch size 100 and learning rate 0.1, whereas for analytic NTKGP we temperature scale for 1000 epochs on full batch size 6000. Like above, we approximate the analytic NTKGP validation predictions (for temperature scaling) by a Monte Carlo ensemble, this time of size 10. We found the various temperature scaling training hyperparameter considerations here to be unimportant to achieve convergence, due to the fact that the temperature scale is a scalar value.

I.4 CIFAR-10 vs SVHN

Baselearners are Myrtle-10 CNNs [40] with 100 channel width and ReLU activations. We use $\sigma_b = 0.01$ and set observation noise $\sigma = 0.1$. Like for MNIST we tune σ_W^2 on a small linear scale around $\sigma_W^2 = 2$. We train using SGD, with momentum parameter 0.9, for 100 epochs and learning rate 0.001, which is decayed to 0.0002 after 80 epochs. In the first 5 epochs we raise the learning rate in linear increments from 0.0001 to 0.001. We use batch size 125. During training we apply random crops and horizontal flips before standardisation. We do not compare to the analytic NTKGP for the Myrtle-10 CNN due to resource requirements.

Figure 8 displays entropy histograms for ensembles trained on CIFAR-10 and tested on in distribution CIFAR-10 test data and out-of-distribution SVHN test data, corresponding to the same experiments as in Figure 5. As we can see, there is a much less noticeable difference between ensemble methods compared to the simpler MNIST vs NotMNIST case.

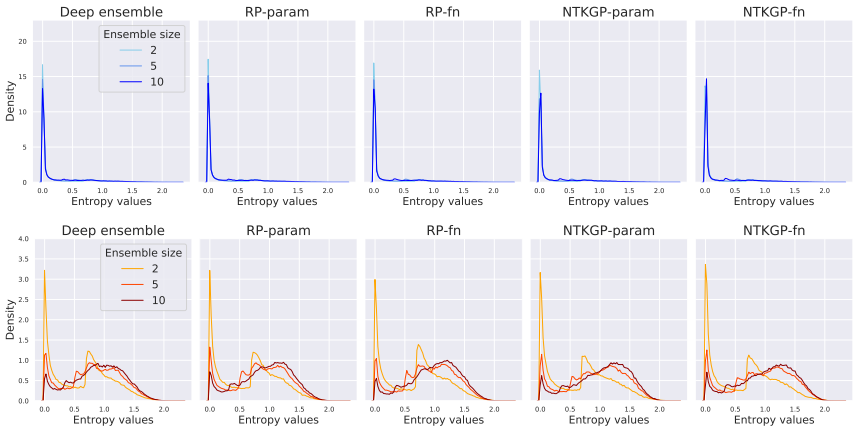


Figure 8: Histograms of predictive entropy on CIFAR-10 (top) and SVHN (bottom) test sets for different ensemble methods and for different ensemble sizes.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Bayesian Deep Ensembles via the Neural Tangent Kernel
Publication Status	Published
Publication Details	Bobby He, Balaji Lakshminarayanan, Yee Whye Teh. Bayesian Deep Ensembles via the Neural Tangent Kernel. In Advances in Neural Information Processing Systems 33, 2020.

Student Confirmation

Student Name:	Bobby He		
Contribution to the Paper	- Lead author with two advisors. - Derived solution to problem that was identified by Yee Whye. - Derived all theoretical results and proofs. - Performed all experiments, with help from advisors to design them. - Wrote the paper, with helpful feedback from advisors.		
Signature 		Date	8/1/23

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Yee Whye Teh		
Supervisor comments Bobby is lead author and did all the research work in this paper.		
Signature 	Date	8 Jan 2023

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 4

Feature Kernel Distillation

This paper was published as the following

> Bobby He and Mete Ozay. Feature Kernel Distillation. In Proceedings of the Tenth International Conference on Learning Representations, 2022.

FEATURE KERNEL DISTILLATION

Bobby He[1,2†] **& Mete Ozay**[2]
[1]Department of Statistics, University of Oxford
[2]Samsung Research UK

ABSTRACT

Trained Neural Networks (NNs) can be viewed as data-dependent kernel machines, with predictions determined by the inner product of last-layer representations across inputs, referred to as the *feature kernel*. We explore the relevance of the feature kernel for Knowledge Distillation (KD), using a mechanistic understanding of an NN's optimisation process. We extend the theoretical analysis of Allen-Zhu & Li (2020) to show that a trained NN's feature kernel is highly dependent on its parameter initialisation, which biases different initialisations of the same architecture to learn different data attributes in a multi-view data setting. This enables us to prove that KD using only pairwise feature kernel comparisons can improve NN test accuracy in such settings, with both single & ensemble teacher models, whereas standard training without KD fails to generalise. We further use our theory to motivate practical considerations for improving student generalisation when using distillation with feature kernels, which allows us to propose a novel approach: Feature Kernel Distillation (FKD). Finally, we experimentally corroborate our theory in the image classification setting, showing that FKD is amenable to ensemble distillation, can transfer knowledge across datasets, and outperforms both vanilla KD & other feature kernel based KD baselines across a range of standard architectures & datasets.

1 INTRODUCTION & BACKGROUND

A prevailing belief in the Deep Learning community is that feature learning, where data-dependent features are acquired during training, is crucial to explaining the empirical success of Neural Networks (NNs) (Fort et al., 2020; Baratin et al., 2021). A comparison in this regard is often made to kernel methods (Jacot et al., 2018), which can be thought of as feature selection methods from a *fixed* data-independent set of features. This separation has been caricaturised as a distinction between *feature learning* and *kernel learning* regimes (Chizat et al., 2019; Yang & Hu, 2020; Woodworth et al., 2020) of NN training. Though less amenable to theoretical analysis compared to kernel regimes, feature learning regimes have the capability to capture more of the complex empirical phenomenon that one can observe in NNs due to parameter-space non-convexity, such as: i) how ensembling trained NNs differing solely in their independent parameter initialisations can lead to improvements in predictive accuracy & uncertainty (Lakshminarayanan et al., 2017; Allen-Zhu & Li, 2020), or ii) the effectiveness of knowledge distillation with both single & ensemble teacher models (Buciluǎ et al., 2006; Hinton et al., 2015). This implies that in order to understand ensembling & knowledge distillation (KD) in NNs, we need to understand the mechanisms of NN feature learning regimes.

Ensembling can be loosely summarised as aggregating predictions from multiple models, & is used widely across machine learning (ML) to improve performance (Dietterich, 2000; Breiman, 2001). Conversely, knowledge distillation (KD), the idea of transferring knowledge from a teacher model to a student model, has garnered the most attention with NNs. Remarkably, via KD it is possible to

[†]Researched during internship at Samsung Research UK. Correspondence to bobby.he@stats.ox.ac.uk.

significantly improve a single student's generalisation with knowledge from a teacher model, or an ensemble of teachers. This means that the single student's model has enough flexibility to generalise well (relative to the teacher), thus one must factor in the optimisation process (such as the parameter initialisation) in order to explain the mechanisms of ensembling and KD in NNs.

To describe KD, suppose we have N input-target $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d \times \mathbb{R}^{C_S}$ data pairs $\hat{\mathcal{D}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ sampled i.i.d. (independent & identically distributed) from some distribution $\mathcal{D}$, and a student NN architecture $f_{\mathcal{S}}(\boldsymbol{x}) = W_{\mathcal{S}} \cdot h_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{S}})$. Here $h_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{S}}) \in \mathbb{R}^{m \times 1}$ is a student-specific feature extractor model (e.g. MLP, CNN, ResNet, or Transformer) with parameters $\boldsymbol{\theta}_{\mathcal{S}}$, and $W_{\mathcal{S}} \in \mathbb{R}^{C_{\mathcal{S}} \times m}$ is a parameter matrix for the last layer. Assume also that we have loss: $\mathcal{L}(\boldsymbol{\theta}_{\mathcal{S}}, W_{\mathcal{S}}) = \frac{1}{N} \sum_{i=1}^N L(f_{\mathcal{S}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$ which we seek to minimise over $\boldsymbol{\theta}_{\mathcal{S}}, W_{\mathcal{S}}$ in the hope that $f_{\mathcal{S}}$ can generalise to unseen $(\boldsymbol{x}, \boldsymbol{y})$ pairs. L is typically cross-entropy in the classification setting.

Vanilla KD (Hinton et al., 2015) distils knowledge from a trained teacher network $f_{\mathcal{T}}(\boldsymbol{x}) = W_{\mathcal{T}} \cdot h_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{T}}) \in \mathbb{R}^{C_{\mathcal{T}}}$ to a student by regularising student $f_{\mathcal{S}}$ towards the teacher $f_{\mathcal{T}}$:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}_{\mathcal{S}}, W_{\mathcal{S}}) = \mathcal{L}(\boldsymbol{\theta}_{\mathcal{S}}, W_{\mathcal{S}}) + \lambda_{\text{KD}} \frac{1}{N} \sum_{i=1}^N L\big(\frac{f_{\mathcal{S}}(\boldsymbol{x}_i)}{\tau}, \frac{f_{\mathcal{T}}(\boldsymbol{x}_i)}{\tau}\big), \tag{1}$$

for temperature $\tau > 0$ & regularisation $\lambda_{\text{KD}} > 0$ hyperparameters. Note, this is only valid if $C_{\mathcal{T}} = C_{\mathcal{S}}$.

Following Hinton et al. (2015), many methods have been proposed using different quirks of NNs to distil knowledge from teacher to student. A relevant line of work involves encouraging the student to match how similar/related the teacher views two inputs $\boldsymbol{x}, \boldsymbol{x}'$ to be (Passalis & Tefas, 2018; Tung & Mori, 2019; Park et al., 2019). These approaches have the benefit of being agnostic to teacher/student architectures & prediction spaces $C_{\mathcal{T}}$ & $C_{\mathcal{S}}$, but as of yet remain heuristically motivated. In this work, we explore such approaches under the more general framework of NN *feature kernel* (the kernel induced by the inner product of last-layer features h) learning, allowing us to provide the missing theoretical justification. Moreover, we use our theoretical insights to introduce practical improvements for FKD in Section 4, which we show outperform these previous works in Section 5.

Allen-Zhu & Li (2020) provide the first theoretical exposition of the mechanisms by which vanilla KD and ensembling improve generalisation in NNs. To this end, the authors introduce the notion of multi-view data, which is when a class in a multi-class classification problem has multiple identifying features/attributes. For example, an image of a car can be discerned by i) wheels, ii) windows, or iii) headlights. The key idea is that the NN parameter initialisation, and its random correlations with certain attributes, will bias the NN to learn only a subset of the entire set of attributes pertaining to a given class. When presented with single-view data lacking the class-identifying attribute that the NN has learnt, the NN will not generalise. For example, an NN that has learnt to classify cars based on if they have headlights will not generalise to a side-on image of a car that occludes headlights.

The implication then is that ensembling NNs works in part because independent parameter initialisations learn independent sets of attributes, so more data features will be learnt across the ensemble model. Moreover, it is argued that vanilla KD in NNs works because the features learnt by the teacher model (or models) are imparted to the student via soft teacher labels that capture ambiguity in a given data input (such as an image of a car whose headlights look like the eyes of a cat). This is fundamentally different to ensembling in strongly convex feature selection problems, such as linear or random features (Rahimi & Recht, 2007) models with ℓ_2 regularisation. In such cases, different initialisations reach the same unique optimum, and additional noise must be added to ensure predictive diversity in the ensemble (Matthews et al., 2017). These analyses suggest that it is not possible to fully explain KD or ensembling in NNs without feature learning, thus motivating our study of *Feature Kernel Distillation*, where one performs KD on NN features directly.

Our contributions Feature learning can be thought of as when the *feature kernel*, induced by the inner product of last-layer representations in a NN, changes during training (Yang & Hu, 2020), and kernel learning in NNs can be thought of as when this kernel is constant. In this work, we take a

feature learning perspective of knowledge distillation (KD). We first highlight the importance of the feature kernel by viewing trained NNs as data-dependent kernel machines, & use this to motivate *Feature Kernel Distillation* (FKD). In FKD, we aim to ensure that the student's feature kernel is well suited for improved generalisation, using both the teacher's data-dependent feature kernel as well as an understanding of the student NN's optimisation process. In Section 3, we adapt the framework of Allen-Zhu & Li (2020) to show that FKD offers the same generalisation benefits as found in vanilla KD in a multi-view data setting, and is further amenable to ensemble distillation. We then derive practical considerations from our insights in Section 4, to improve FKD through an understanding of the NN's feature learning optimisation process, compared to previous methods which implicitly used the feature kernel for KD. Finally, in Section 5, we provide experimental support that our theoretical claims extend to standard image classification settings, by: verifying that FKD is amenable to ensemble distillation; can transfer knowledge across datasets with different prediction spaces (unlike vanilla KD); and outperforms vanilla KD & previous feature kernel based distillation methods over a range of architectures on CIFAR-100 and ImageNet-1K.

2 Motivation for Feature Kernel Distillation

One obvious limitation of vanilla KD is that student $f_{\mathcal{S}}$ and teacher $f_{\mathcal{T}}$ need to share prediction spaces, i.e. $C_{\mathcal{S}}{=}C_{\mathcal{T}}$. In many situations, we may have a teacher network trained on a dataset with a different number of classes than the student's dataset, and it is not clear how one could apply vanilla knowledge distillation. One possibility could be to regularise directly in feature space by comparing $h_{\mathcal{S}}$ and $h_{\mathcal{T}}$ element-wise, but again this either requires same teacher-student last-layer sizes or additional projection layers.

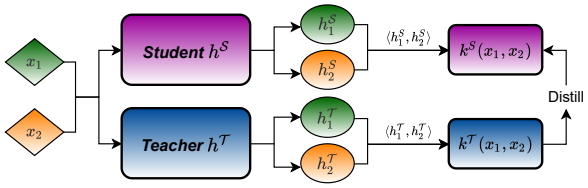


Figure 1: Feature Kernel Distillation (FKD) from the feature extractor of a teacher $h^{\mathcal{T}}$ to that of a student $h^{\mathcal{S}}$.

To eschew such unnecessary complications, we take the perspective of NNs as data-dependent kernel machines. Define an NN's *feature kernel* to be:

Definition 1 (Feature Kernel)**.** *Suppose we have parameters $\boldsymbol{\theta}$ and last-layer NN feature extractor $h(\cdot, \boldsymbol{\theta})$:$\mathbb{R}^d \mapsto \mathbb{R}^m$. For two inputs $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^m$, the feature kernel k is the kernel induced by the inner product of $h(\boldsymbol{x}_i, \boldsymbol{\theta})$ and $h(\boldsymbol{x}_j, \boldsymbol{\theta})$, that is: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) \stackrel{\text{def}}{=} \langle h(\boldsymbol{x}_i, \boldsymbol{\theta}), h(\boldsymbol{x}_j, \boldsymbol{\theta}) \rangle$.*

At initialisation, it is well known that in the infinite NN-width limit, with appropriate scaling, the feature kernel k converges almost surely to a deterministic kernel known as the Neural Network Gaussian Process (NNGP) kernel (Neal, 2012; Lee et al., 2018; Matthews et al., 2018; Yang, 2019). Yang & Hu (2020) show that there is a parameterisation-dependent dichotomy between kernel & feature learning regimes for infinite-width NNs, where the feature kernel k is constant or changes during training, respectively. It has been widely demonstrated that a crucial component of the success of finite-width NNs is their ability to flexibly learn features, and indeed the feature kernel, from data during training (Fort et al., 2020; Aitchison, 2020; Chen et al., 2020b; Maddox et al., 2021).

To see the importance of the feature kernel, note that for a fixed $\boldsymbol{\theta}$ with many common loss functions L, and some mild assumptions on strong convexity (which could be enforced e.g. with standard ℓ_2 regularisation), the optimal W is uniquely determined and k determines the entire predictive function $f^*(\cdot)$. For example, with squared error, $L(f(\boldsymbol{x}), \boldsymbol{y}) = \|f(\boldsymbol{x}) - \boldsymbol{y}\|_2^2$, and ℓ_2 regularisation strength $\lambda > 0$, a trained NN is precisely kernel ridge regression with the data-dependent feature kernel k, whose job is to measure how similar different inputs are. Thus, *all* teacher knowledge is contained in its feature kernel, $k^{\mathcal{T}}$, so the feature kernel can act as our primary distillation target, as depicted in Fig. 1. We show a corresponding result for cross-entropy loss in App. A.

Fig. 2 corroborates our claims. For a ResNet20v1 (He et al., 2016) 'reference' model trained on CIFAR10 with cross entropy, we plot test class prediction confusion matrices between said model and:

i) a retrained version where all but the last-layer parameters are fixed (hence fully determined by the reference model's feature kernel as per App. A), & ii) an independent model trained from a different initialisation. As expected, there is significantly more disagreement across test predictions for models with different initialisations than those which share feature kernels. This suggests: a) different initialisations bias the same architecture to learn different features, and b) the feature kernel (largely) determines a model's test predictions. Experimental details and a breakdown of the predictive disagreements can be found in App. F.

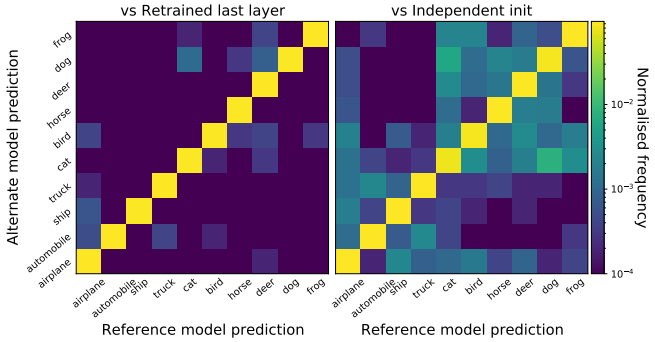


Figure 2: CIFAR10 test prediction confusion matrices between a fixed reference model and a model with: (left) retrained last layer, and (right) independent initialisation.

Having described the feature kernel as a central object in any NN, we use this to motivate our proposed FKD, where we treat the teacher's feature kernel, $k^{\mathcal{T}}$, as a key distillation target for the student's feature kernel, $k^{\mathcal{S}}$. Encouraging similarity across feature kernels shares useful features that the teacher has learnt with the student, which we theoretically show in Section 3.

We define the FKD student loss function via an additive regularisation term between feature kernels:[1]

$$\mathcal{L}^{\lambda_{\text{KD}}}(\boldsymbol{\theta}, W) \overset{\text{def}}{=} \mathcal{L}(\boldsymbol{\theta}, W) + \lambda_{\text{KD}} \cdot \mathbb{D}(k_{\boldsymbol{\theta}}, k^{\mathcal{T}}) \tag{2}$$

where $\lambda_{\text{KD}} > 0$ is the regularisation strength, $\mathbb{D}$ is some (pseudo-)distance over kernels, and the student feature kernel $k^{\mathcal{S}} = k_{\boldsymbol{\theta}}$ is written to make explicit the dependence on student feature extractor parameters $\boldsymbol{\theta}$. We stress that Eq. (2) does not require matching prediction (nor feature) spaces between teacher and student, allowing us to apply FKD across tasks, architectures, and datasets.

We consider Eq. (2) with $\mathbb{D}$ set to:

$$\mathbb{D}(k_{\boldsymbol{\theta}}, k^{\mathcal{T}}) = \mathbb{E}_{\boldsymbol{x}_1, \boldsymbol{x}_2 \overset{\text{i.i.d.}}{\sim} \hat{\mathcal{D}}} \left[\left(k_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \boldsymbol{x}_2) - k^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2) \right)^p \right], \tag{3}$$

with expectation approximated by an average over a minibatch. In this work we choose $p = 2$, so that $\sqrt{\mathbb{D}}$ gives the Frobenius norm of the difference in feature kernel gram matrices over a batch.

3 Theoretical analysis for FKD

We now adapt the theoretical framework of Allen-Zhu & Li (2020), which is restricted to vanilla KD, to demonstrate the generalisation benefits of FKD over standard training. Note that FKD distils knowledge by comparing different data points, whereas vanilla KD compares a single data point across classes: this core difference is reflected throughout our analysis relative to Allen-Zhu & Li (2020). We first describe the multi-view data setting & CNN architecture we consider, before recalling that standard training without KD fails to generalise well. We then provide our main theoretical result, Theorem 2, which shows that FKD improves student test performance. Though our theoretical results are limited to a specific scenario, inspired by real-world data (Allen-Zhu & Li, 2020), & NN architecture, we believe the setup we consider is apt: it is simple enough to be tractable, yet rich enough to display the merits of FKD. Moreover, we find in Section 5 that our conclusions generalise to standard architectures & image datasets. In the interest of space & readability, we focus on providing intuition in this section, and fill in remaining details/proofs in the appendix.

[1] We will sometimes drop the student $\mathcal{S}$ sub/superscript where obvious for clarity, like in Eq. (2). Any teacher specific object, e.g. $k^{\mathcal{T}}$, will always have corresponding $\mathcal{T}$ sub/superscript.

Multi-view data. We consider the data classification problem introduced by Allen-Zhu & Li (2020), with C classes and inputs $\boldsymbol{x}$ with P patches each of dimension d, meaning $\boldsymbol{x} \in (\mathbb{R}^d)^P$. For each class c, we suppose that there exist two attributes $v_{c,1}, v_{c,2} \in \mathbb{R}^d$. For $\boldsymbol{x}$ belonging to class c, the attributes found in patches of $\boldsymbol{x}$ will include $v_{c,1}$ and $v_{c,2}$, as well as a random selection of out-of-class attributes $\{v_{c',l}\}_{c' \neq c, l \in [2]}$.[2] This denotes the multi-view nature of the data distribution. In the true data-generating distribution $\mathcal{D}$, we suppose that a proportion μ of the data $(\boldsymbol{x}, \boldsymbol{y})$ is single-view, which means that only one of $v_{c,1}$ or $v_{c,2}$ is present in $\boldsymbol{x}$ when $(\boldsymbol{x}, \boldsymbol{y})$ is from class c. These will be the data for which standard training fails to generalise. A precise definition of multi-view data is presented in App. B.1.1. Allen-Zhu & Li (2020) argue that this multi-view setting provides a compelling proxy for standard image datasets such as CIFAR-10/100 Krizhevsky (2009).[3]

Intuition: FKD on multi-view data

Suppose we have an image classification task, with cat & car just two out of many classes. For the car class, $v_{c,1}$ could correspond to headlights, whilst $v_{c,2}$ could correspond to wheels. We would then expect $v_{c,1}$ to also appear in patches of an input image, $\boldsymbol{x}_{\text{cat}}$, corresponding to a cat with headlight-like eyes. Allen-Zhu & Li (2020) show that a single trained model is biased to learn exactly one of $v_{c,1}$ or $v_{c,2}$, depending on its parameter initialisation. W.L.O.G, suppose that the student is biased to learn $v_{c,2}$ & not $v_{c,1}$. If the teacher model has learnt $v_{c,1}$, this means that the teacher model knows there is a similarity between $\boldsymbol{x}_{\text{cat}}$ & any car image, $\boldsymbol{x}_{\text{car}}$, that displays headlights. Mathematically, we show that this corresponds to a large value for $k_{\mathcal{T}}(\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{car}})$. Our FKD regularisation forces the student to also have a large value for $k_{\mathcal{S}}(\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{car}})$, ensuring that attribute $v_{c,1}$ is also learnt by the student network. Without distillation, a student NN which has learnt $v_{c,2}$ & not $v_{c,1}$ will not generalise to front-on images of cars that hide wheels.

Convolutional NN & corresponding feature kernel. Like Allen-Zhu & Li (2020), for our theoretical analyis we consider a single hidden-layer convolutional NN (CNN) with sum-pooling.[4] For each class $c \in [C]$, we suppose that the CNN has m channels, giving Cm channels in total. For channel r and class c, we suppose that we have weights $\boldsymbol{\theta}_{c,r} \in \mathbb{R}^d$. This gives output for class c by

$$f_c(\boldsymbol{x}) \stackrel{\text{def}}{=} \sum_{r=1}^{m} \sum_{p=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p \rangle) \tag{4}$$

where for ease of analysis $\widetilde{\text{ReLU}}$ is ReLU-like but with continuous gradient, see App. B.2.

Before we consider FKD, we must first define the feature kernel for this CNN f. To do so, we recast $f(\boldsymbol{x}) = W \cdot h(\boldsymbol{x}, \boldsymbol{\theta})$, where $h(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^{Cm}$ and $W \in \mathbb{R}^{C \times Cm}$ satisfying, for $r \in [m], c \in [C], c' \in [C]$:

$$h(\boldsymbol{x}, \boldsymbol{\theta})_{r+(c-1)m} \stackrel{\text{def}}{=} \sum_{p=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p \rangle), \quad \text{and} \quad W_{c',r+(c-1)m} \stackrel{\text{def}}{=} \mathbb{1}\{c = c'\}. \tag{5}$$

Given that the feature kernel is $k(\boldsymbol{x}, \boldsymbol{x}') \stackrel{\text{def}}{=} \langle h(\boldsymbol{x}, \boldsymbol{\theta}), h(\boldsymbol{x}', \boldsymbol{\theta}) \rangle$, we now have that:[5]

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{c=1}^{C} \sum_{r=1}^{m} \sum_{p,p'=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p \rangle) \cdot \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}'_{p'} \rangle). \tag{6}$$

We first recall that standard training of the model f with gradient descent and cross entropy loss fails to generalise on half the μ proportion of data that is single-view.

[2] It is straightforward to extend to the case of more than two views per class if need be.

[3] https://www.microsoft.com/en-us/research/blog/three-mysteries-in-deep-learning-ensemble-knowledge-distillation-and-self-distillation/

[4] It is straightforward to extend our analysis for max-pooling.

[5] The feature kernel defined in Eq. (6) corresponds to the Global Average Pooling CNN-GP kernel in Novak et al. (2018) in the infinite-channel limit, which captures intra-patch correlations unlike the vectorised CNN-GP, which corresponds to vectorising the spatial dimensions to give CmP rather than Cm channels.

Theorem 1 (Standard training fails, Theorem 1 of Allen-Zhu & Li (2020))**.** *For sufficiently many classes C and channels $m \in [\text{polylog}(C), C]$, with learning rate $\eta \leq \frac{1}{\text{poly}(C)}$, training time $T^* = \frac{\text{poly}(C)}{\eta}$, and multi-view data distribution (App. B.1.1), the trained model $f^{(T^*)}$ satisfies with probability at least $1 - e^{-\Omega(\log^2(C))}$:*

- *Training accuracy is perfect: For all $(\boldsymbol{x}, \boldsymbol{y}) \in \hat{\mathcal{D}}$, $\boldsymbol{y} = \text{argmax}_c f_c^{(T^*)}(\boldsymbol{x})$.*
- *Test accuracy is bad but consistent: $\mathbb{P}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\left[\boldsymbol{y} \neq \text{argmax}_c f_c^{(T^*)}(\boldsymbol{x})\right] \in [0.49\mu, 0.51\mu]$.*

Now we are ready to show that regularising the student model towards the teacher model's feature kernel, as in FKD, improves test accuracy. We suppose our teacher model is an ensemble of $E \geq 1$ models $\{f_e\}_{e=1}^E$, each with corresponding feature kernel k_e, trained as standard on the same data with independent initialisations $\boldsymbol{\theta}_0^e$. We average k_e over $e \in [E]$ to obtain our teacher feature kernel:

$$k_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{E}\sum_{e=1}^{E} k_e(\boldsymbol{x}, \boldsymbol{x}'). \tag{7}$$

This is akin to concatenating all features in $\{h_e\}_{e=1}^E$ into a ECm-dimensional feature vector, albeit without the additional computational baggage. We then have our main theoretical result:

Theorem 2 (FKD improves student generalisation and is better with larger ensemble)**.** *Given an arbitrary $\epsilon > 0$. For any ensemble size E of teacher NNs trained as in Theorem 1 and sufficiently many classes C, for $m = \text{polylog}(C)$, with learning rate $\eta \leq \frac{1}{\text{poly}(C)}$, and training time $T^* = \frac{\text{poly}(C)}{\eta}$, the ensemble teacher knowledge can be distilled into a single student model $f^{(T^*)}$ using only teacher feature kernel $k_{\mathcal{T}}$, Eq. (7), such that with probability at least $1 - e^{-\Omega(\log^2(C))}$:*

- *Training accuracy is perfect: For all $(\boldsymbol{x}, \boldsymbol{y}) \in \hat{\mathcal{D}}$, $\boldsymbol{y} = \text{argmax}_c f_c^{(T^*)}(\boldsymbol{x})$.*
- *Test accuracy is good: $\mathbb{P}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\left[\boldsymbol{y} \neq \text{argmax}_c f_c^{(T^*)}(\boldsymbol{x})\right] \leq (\frac{1}{2^{E+1}} + \epsilon)\mu$.*

Proof outline for Theorem 2, we first show, in Lemma 1, that a single trained NN's feature kernel (which we defined in Eq. (6)) can detect if two inputs share a data attribute that the NN has learnt due to its weight initialisation. We extend this result to an ensemble teacher in App. C.2, showing that the ensemble teacher feature kernel detects the union of all data attributes learnt by individual $\{k_e\}_{e=1}^E$. This simplifies our calculations when showing that our distillation regulariser, Eq. (3), is effective for improved student generalisation. The full proof can be found in App. C.

Intuition: FKD with ensemble of teachers

To parse Theorem 2, suppose we have a single teacher i.e. E=1. Then, the test error is essentially 0.25μ in Theorem 2. The explanation is that both the student & teacher networks independently learn one of $\{v_{c,l}\}_{l=1}^2$ for each class c. Either vanilla KD (Theorem 4 of Allen-Zhu & Li (2020)) or our feature kernel approach allow the student to learn the union of the independent attributes learnt by the student and the teacher, so for only a quarter of the single-view test data $\boldsymbol{x}_s$ will the student not have learnt the useful class attribute present in $\boldsymbol{x}_s$. For general ensemble size E, the story is the same: the student & E teachers each independently learn one of the two useful attributes $\{v_{c,l}\}_{l\in 2}$ for all $c \in [C]$. Distilling allows the student to learn the union of these attributes, which means that the student will fail on only $\frac{1}{2^{E+1}}$ of the single-view data.

4 FKD IN PRACTICE

Next, we highlight practical considerations for implementing FKD derived from our theory. Pseudo-code and PyTorch-style code for our FKD implementation are given in Algs. 1 and 2 respectively.

Correlation kernel. We propose $\mathbb{D}(\rho_{\boldsymbol{\theta}}, \rho_{\mathcal{T}})$ as a regulariser in FKD instead of $\mathbb{D}(k_{\boldsymbol{\theta}}, k_{\mathcal{T}})$, where:

$$\rho_z(\boldsymbol{x}, \boldsymbol{x}') \stackrel{\text{def}}{=} \frac{k_z(\boldsymbol{x}, \boldsymbol{x}')}{\sqrt{k_z(\boldsymbol{x}, \boldsymbol{x}) k_z(\boldsymbol{x}', \boldsymbol{x}')}}, \quad \text{and} \quad \rho_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}') \stackrel{\text{def}}{=} \frac{1}{E} \sum_{e=1}^{E} \rho_e(\boldsymbol{x}, \boldsymbol{x}')$$

defines *feature correlation kernel* ρ_z, corresponding to feature kernel k_z, for $z \in [E] \cup \{\boldsymbol{\theta}\}$. The reason we use correlation kernels is that they normalise data, so that $\rho(\boldsymbol{x}, \boldsymbol{x}) = 1 \ \forall \boldsymbol{x}$, which zeros diagonal differences in Eq. (3): we hope FKD allows the student to learn from the teacher features shared between *different* inputs. Non-zero diagonal differences, like in Similarity-Preserving (SP) KD (Tung & Mori, 2019), encourage the student to learn noise as we show in App. D.1, and we hypothesise that this contributes to the improved performance we observe of FKD over SP in Section 5. Moreover, this normalisation helps balance individual teacher's influence in an ensemble teacher, and ensures that FKD does not need a temperature hyperparameter τ, which produces soft labels in vanilla KD.

Feature regularisation. One downside to using the correlation kernel is that our FKD regularisation, $\mathbb{D}(\rho_{\boldsymbol{\theta}}, \rho_{\mathcal{T}})$, becomes invariant to the scale of $k_{\boldsymbol{\theta}}$. For example, replacing $k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$ with $\sqrt{M(\boldsymbol{x}) M(\boldsymbol{x}')} k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$, for any $M(\boldsymbol{x}) : \mathbb{R}^d \mapsto \mathbb{R}^+$, leaves the student correlation kernel unchanged. This may lead to degeneracies when training $\boldsymbol{\theta}$, & large variations in $k(\boldsymbol{x}, \boldsymbol{x})$ over $\boldsymbol{x}$ may harm generalisation (as evidenced by the fact that input normalisation is common across ML, from linear models to NNs). Moreover, our proof of Theorem 2 is not quantitative, in that we only show $k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \tilde{\Theta}(1)$ in the number of classes C *up to* poly-logarithmic factor in C, when $\boldsymbol{x} \neq \boldsymbol{x}'$ share a data attribute $v_{c,l}$ that has been learned by parameters $\boldsymbol{\theta}$. These insights motivate us to regularise the student feature h during FKD training to control the norm of $k(\boldsymbol{x}, \boldsymbol{x})$ across inputs $\boldsymbol{x}$. We use an additive ℓ_2 regularisation $\frac{1}{B} \sum_{b=1}^{B} \|h(\boldsymbol{x}_b, \boldsymbol{\theta})\|_2^2 = \frac{1}{B} \sum_{b=1}^{B} k(\boldsymbol{x}_b, \boldsymbol{x}_b)$ with regularisation strength $\lambda_{\text{FR}} > 0$, for each minibatch $\{(\boldsymbol{x}_b, \boldsymbol{y}_b)\}_{b=1}^{B}$. Fig. 3 shows that using Feature Regularisation (FR) encourages a more even spread of $k(\boldsymbol{x}, \boldsymbol{x})$ across inputs for a student VGG8 network trained with a VGG13 teacher model on CIFAR-100. Corresponding plots for other architectures can be found in Fig. 7. Similar to Dauphin & Cubuk (2021), we find that FR improves generalisation & provide ablations in Section 5.

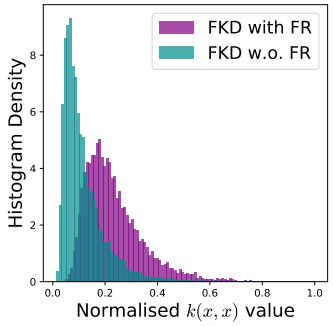


Figure 3: Histogram of normalised feature kernel values, $\frac{k(\boldsymbol{x}, \boldsymbol{x})}{\max_{\boldsymbol{x}'} k(\boldsymbol{x}', \boldsymbol{x}')}$, over the CIFAR-100 test set.

Algorithm 1 Feature Kernel Distillation with SGD.

Require: Maximum number of iterations T^*, batch size B, learning rate η, teacher correlation kernel $\rho_{\mathcal{T}}$, FKD regularisation $\lambda_{\text{KD}} > 0$, Feature regularisation $\lambda_{\text{FR}} > 0$. Initialise student parameters $\boldsymbol{\theta}_0, W_0$.
for iteration $t = 0, \ldots, T^*$ **do**
  Sample minibatch $(\boldsymbol{x}_i^B, y_i^B)_{i=1}^B \overset{\text{i.i.d.}}{\sim} \hat{\mathcal{D}}$.
  Compute loss $\mathcal{L} = \frac{1}{B} \sum_{i=1}^{B} L(f(\boldsymbol{x}_i^B), y_i^B) + \frac{2\lambda_{\text{KD}}}{B(B-1)} \sum_{i \neq j}^{B} \left(\rho_{\boldsymbol{\theta}_t}(\boldsymbol{x}_i^B, \boldsymbol{x}_j^B) - \rho_{\mathcal{T}}(\boldsymbol{x}_i^B, \boldsymbol{x}_j^B)\right)^2$.
  Add feature regularisation $\mathcal{L} = \mathcal{L} + \frac{\lambda_{\text{FR}}}{B} \sum_{i=1}^{B} \left\|h(\boldsymbol{x}_i^B, \boldsymbol{\theta}_t)\right\|_2^2$, (optional).
  Update parameters $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}$, $W_{t+1} \leftarrow W_t - \eta \nabla_W \mathcal{L}$.
end for
return $\{\boldsymbol{\theta}_{T^*}, W_{T^*}\}$

5 Experiments

Due to space concerns, App. F contains further experiments & any missing experimental details.

Ensemble distillation. We first verify that larger ensemble teacher size, E, further improves FKD student performance as suggested by Theorem 2. This is confirmed in Fig. 4, using VGG8 for all student & teacher networks on the CIFAR-100 dataset.

We also plot the test accuracy of the ensemble teacher across sizes E, whose predictive probabilities are averaged over individual teachers, as well as the test accuracy of an undistilled student model. We see that FKD consistently outperforms vanilla KD, and both distillation methods outperform the teacher in the 'self-distillation' setting of E=1 (Furlanello et al., 2018; Zhang et al., 2019). Moreover, FKD allows a single student to match teacher performance when E=2, before positive but diminishing returns with larger E relative to the teacher ensemble.

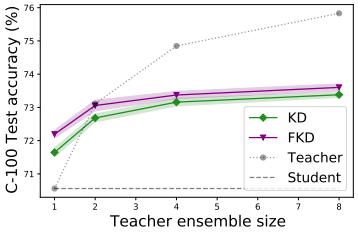


Figure 4: FKD as teacher ensemble size changes. Error bars denote 95% confidence for mean of 10 runs.

Dataset Transfer. We next show FKD can transfer knowledge across similar datasets. From a fixed VGG13 teacher network trained on CIFAR-100, we distil to student VGG8 NNs on CIFAR-10, STL-10 & Tiny-ImageNet. As no student dataset has 100 classes, unlike CIFAR-100, it is not clear how one can use vanilla KD (Hinton et al., 2015) in this case. We thus compare FKD to other feature kernel based KD methods: Relational KD (RKD) (Park et al., 2019) & Similarity-Preserving (SP) KD (Tung & Mori, 2019). In Table 1, we see that FKD without feature regularisation outperforms both baselines across all datasets, and that feature regularisation (FR) further improves FKD performance, highlighting the benefit of our practical considerations in Section 4. The improved performance is particularly stark on STL-10 (which we downsize to 32x32 resolution), where FKD improves student performance by 2.75%. STL-10 is well suited for FKD as it has only 5K labeled inputs but 100K unlabeled datapoints, which can be used in our feature kernel regulariser, $\mathbb{D}(\rho_{\boldsymbol{\theta}}, \rho_{\mathcal{T}})$.

Table 1: Test accuracy (%) of FKD & baselines in a dataset transfer distillation setting. Error bars indicate 95% confidence for mean of 10 runs.

Dataset	Student	RKD	SP	FKD w.o. FR	FKD
C-100 → C-10	91.56	$91.74_{\pm 0.09}$	$92.21_{\pm 0.07}$	$92.33_{\pm 0.07}$	$\mathbf{92.61}_{\pm 0.07}$
C-100 → STL-10	72.69	$72.86_{\pm 0.27}$	$73.88_{\pm 0.17}$	$\mathbf{75.17}_{\pm 0.30}$	$\mathbf{75.44}_{\pm 0.31}$
C-100 → Tiny-I	48.53	$48.74_{\pm 0.17}$	$48.73_{\pm 0.14}$	$48.85_{\pm 0.09}$	$\mathbf{50.67}_{\pm 0.12}$

Table 2: CIFAR-100 and ImageNet-1K accuracies (%) comparing FKD with KD baselines. * denotes result from Tian et al. (2020); FKD uses the same teacher checkpoints provided by the authors,[6] with error bars denoting 95% confidence for the mean over 5 students.

	CIFAR-100				**ImageNet-1K**
Teacher	ResNet32x4	VGG13	ResNet32x4	ResNet50	ResNet34
Student	ResNet8x4	VGG8	ShuffleNetV1	VGG8	ResNet18
Teacher*	79.42	74.64	79.42	79.34	73.26
Student*	72.50	70.36	70.50	70.36	69.97
KD* (Hinton et al., 2015)	$73.33_{\pm 0.22}$	$72.98_{\pm 0.17}$	$74.07_{\pm 0.17}$	$73.81_{\pm 0.11}$	70.66
RKD* (Park et al., 2019)	$71.90_{\pm 0.10}$	$71.48_{\pm 0.04}$	$72.28_{\pm 0.34}$	$71.50_{\pm 0.06}$	N/A
SP* (Tung & Mori, 2019)	$72.94_{\pm 0.20}$	$72.68_{\pm 0.17}$	$73.48_{\pm 0.37}$	$73.34_{\pm 0.30}$	70.62
CRD* (Tian et al., 2020)	$\mathbf{75.51}_{\pm \mathbf{0.16}}$	$\mathbf{73.94}_{\pm \mathbf{0.19}}$	$\mathbf{75.11}_{\pm \mathbf{0.28}}$	$74.30_{\pm 0.12}$	71.17
FKD w.o. FR	$74.89_{\pm 0.24}$	$73.08_{\pm 0.16}$	$74.66_{\pm 0.23}$	$73.99_{\pm 0.15}$	70.84
FKD	$\mathbf{75.57}_{\pm \mathbf{0.22}}$	$\mathbf{73.78}_{\pm \mathbf{0.17}}$	$\mathbf{75.00}_{\pm \mathbf{0.30}}$	$\mathbf{74.61}_{\pm \mathbf{0.28}}$	**71.23**

Comparison on CIFAR-100 and ImageNet. Finally, we compare FKD to various knowledge distillation baselines on CIFAR-100 and ImageNet, across a selection of teacher/student architectures. We see in Table 2 that FKD consistently outperforms: vanilla KD (Hinton et al., 2015), RKD (Park et al., 2019), and SP (Tung & Mori, 2019). Moreover, FKD either matches or outperforms the high-performing Contrastive Representational Distillation (Tian et al., 2020). We use the exact same teacher checkpoints used by Tian et al. (2020) and Chen et al. (2021) for CIFAR-100 and ImageNet respectively to ensure fair comparison. We find, like in Table 1, that feature regularisation consistently improves FKD performance and that even without feature regularisation, FKD outperforms all feature kernel based KD methods. This implies that using the correlation kernel to zero out diagonal differences, as described in Section 4, indeed helps improve student performance.

[6] Apart from ImageNet-1K which used the pretrained ResNet34 from torchvision, like Chen et al. (2021)

6 RELATED WORK

NN Knowledge Distillation. Following Hinton et al. (2015), there has been much interest in expanding KD in NNs (Romero et al., 2014; Zagoruyko & Komodakis, 2016; Passalis & Tefas, 2018; Zhang et al., 2018; Yu et al., 2019; Chen et al., 2020a; Tian et al., 2020). Most similar to FKD are Park et al. (2019); Tung & Mori (2019) who also use relations between inputs to distil knowledge (albeit not from the feature kernel learning perspective and without our theoretical justification), as well as Qian et al. (2020) who focus on reducing computational costs of full-batch feature kernel matrix operations. App. D highlights in more detail the differences of FKD compared to previous pairwise feature kernel based KD methods. Allen-Zhu & Li (2020) made the first theoretical connection using the mechanisms of ensembling in NNs to explain the success of vanilla KD in NNs, which we extend for feature kernel based KD.

Ensembling NNs. Ensembling NNs has long been studied for improving predictive accuracy (Hansen & Salamon, 1990; Krogh et al., 1995) with particular recent focus towards uncertainty quantification & Bayesian inference (Lakshminarayanan et al., 2017; Ovadia et al., 2019; Zaidi et al., 2020; Pearce et al., 2020; He et al., 2020; Wilson & Izmailov, 2020; Wenzel et al., 2020; D'Angelo & Fortuin, 2021; Schut et al., 2021) and predictive diversity (Fort et al., 2019; D'Amour et al., 2020). On the topic of Bayesian inference, the feature kernel has also been studied under the name of Neural Linear Model (Riquelme et al., 2018; Ober & Rasmussen, 2019), and extensions treating features h as inputs to standard Gaussian Process kernels are known by the name of Deep Kernel Learning (Wilson et al., 2016; Ober et al., 2021; van Amersfoort et al., 2021).

NN Feature learning. A recent flurry of work has focussed on characterising & understanding the importance of feature learning in NNs (Chizat et al., 2019; Fort et al., 2020; Baratin et al., 2021; Lee et al., 2020; Aitchison, 2020; Ghorbani et al., 2020), fuelled in part by the development that wide NNs become (Neural Tangent) Kernel machines in certain regimes (Jacot et al., 2018; Lee et al., 2019; Yang & Littwin, 2021), thus forgoing feature learning. The consensus in these works is that there are gaps between NTK theory & practical NNs that cannot be explained without feature learning. However, Yang & Hu (2020) proved that feature-learning is still possible with infinite-width NNs, and also that feature learning is equivalent to feature kernel learning in infinite-width NNs. This motivates our study of the feature kernel as a key object for distillation. Regularising the feature kernel to the true target covariance kernel was suggested by Yoo et al. (2021).

7 CONCLUSION

We have theoretically shown that the feature kernel is a valid object for Knowledge Distillation (KD) in Neural Networks (NNs) by extending the analysis of Allen-Zhu & Li (2020), which focused on vanilla KD (Hinton et al., 2015). Further, we used our theoretical insights to motivate practical considerations when using feature kernels for distillation, such as using the feature correlation kernel & using feature regularisation, to improve on previous feature based KD methods. We term our approach Feature Kernel Distillation (FKD), and note that FKD is more widely applicable than vanilla KD, as it benefits from being agnostic to teacher and student prediction spaces. Experimentally, we have demonstrated that FKD is amenable to ensemble distillation as suggested by our theory, is able to transfer knowledge across similar datasets and that FKD outperforms vanilla KD & previous feature kernel based KD methods across a variety of architectures on CIFAR-100, and ImageNet-1K.

Limitations & future work. Though feature learning is central to our results, we stress that there are still gaps between our theory & practice to understanding NN ensembling & KD, demonstrated by the divergence between ensemble teacher & FKD in Fig. 4 for larger ensemble size. This could be due to: the multi-view data setting not being able to capture the full complexity of real-world data, the role of hierarchical feature learning between layers in a deep NN, or the importance of mini-batching in stochastic gradient descent. Other future work could apply the multi-view data setting to analyse uncertainty quantification in NN ensembles, assess the impact of different FKD regularisation metrics in Eq. (3), or improve FKD further to compete with state-of-the-art KD methods.

Acknowledgements We thank Emilien Dupont, Yee Whye Teh, and Sheheryar Zaidi for helpful feedback on this work.

REFERENCES

Laurence Aitchison. Why bigger is not always better: on finite and infinite neural networks. In *International Conference on Machine Learning*, pp. 156–164. PMLR, 2020.

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pp. 2269–2277. PMLR, 2021.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Defang Chen, Jian-Ping Mei, Can Wang, Yan Feng, and Chun Chen. Online knowledge distillation with diverse peers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3430–3437, 2020a.

Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Zhe Wang, Yan Feng, and Chun Chen. Cross-layer distillation with semantic calibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7028–7036, 2021.

Shuxiao Chen, Hangfeng He, and Weijie Su. Label-aware neural tangent kernel: Toward better generalization and local elasticity. *Advances in Neural Information Processing Systems*, 33, 2020b.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32:2937–2947, 2019.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*, 2020.

Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *arXiv preprint arXiv:2106.11642*, 2021.

Yann Dauphin and Ekin Dogus Cubuk. Deconstructing the regularization of batchnorm. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=d-XzF81Wg1`.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33, 2020.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pp. 1607–1616. PMLR, 2018.

Yan Gao, Titouan Parcollet, and Nicholas D. Lane. Distilling knowledge from ensembles of acoustic models for joint ctc-attention end-to-end speech recognition. volume abs/2005.09310, 2020.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *arXiv preprint arXiv:2006.13409*, 2020.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian deep ensembles via the neural tangent kernel. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1010–1022. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0b1ec366924b26fc98fa7b71a9c249cf-Paper.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8580–8589, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7:231–238, 1995.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep Neural Networks as Gaussian Processes. In *International Conference on Learning Representations*, 2018.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32:8572–8583, 2019.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33, 2020.

Wesley Maddox, Shuai Tang, Pablo Moreno, Andrew Gordon Wilson, and Andreas Damianou. Fast adaptation with linearized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 2737–2745. PMLR, 2021.

Alexander G de G Matthews, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Sample-then-optimize posterior sampling for bayesian linear models. 2017.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian Process Behaviour in Wide Deep Neural Networks. In *International Conference on Learning Representations*, volume 4, 2018.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2018.

Sebastian W Ober and Carl Edward Rasmussen. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*, 2019.

Sebastian W Ober, Carl E Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. *arXiv preprint arXiv:2102.12108*, 2021.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 32:13991–14002, 2019.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3967–3976, 2019.

Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 268–284, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019a. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019b.

Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, pp. 234–244. PMLR, 2020.

Qi Qian, Hao Li, and Juhua Hu. Efficient kernel transfer in knowledge distillation. *arXiv preprint arXiv:2009.14416*, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pp. 1177–1184, 2007.

Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, Elena Rastorgueva, François Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, and Yoshua Bengio. SpeechBrain: A general-purpose speech toolkit. 2021.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Lisa Schut, Edward Hu, Greg Yang, and Yarin Gal. Deep Ensemble Uncertainty Fails as Network Width Increases: Why, and How to Fix It. In *Workshop on Uncertainty & Robustness in Deep Learning*, 2021.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. Multilingual neural machine translation with knowledge distillation. In *International Conference on Learning Representations*, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SkgpBJrtvS`.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1365–1374, 2019.

Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021.

Kees Van Den Doel, Uri M Ascher, and Eldad Haber. The lost honor of ℓ_2-based regularization. In *Large scale inverse problems*, pp. 181–203. De Gruyter, 2013.

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*, 2020.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pp. 3635–3673. PMLR, 2020.

Greg Yang. Tensor Programs I: Wide Feedforward or Recurrent Neural Networks of Any Architecture are Gaussian Processes. *arXiv preprint arXiv:1910.12478*, 2019.

Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. *arXiv preprint arXiv:2105.03703*, 2021.

Boseon Yoo, Jiwoo Lee, Janghoon Ju, Seijun Chung, Soyeon Kim, and Jaesik Choi. Conditional temporal neural processes with covariance loss. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12051–12061. PMLR, 18–24 Jul 2021. URL `http://proceedings.mlr.press/v139/yoo21b.html`.

Lu Yu, Vacit Oguz Yazici, Xialei Liu, Joost van de Weijer, Yongmei Cheng, and Arnau Ramisa. Learning metrics from teachers: Compact networks for image embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2907–2916, 2019.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris Holmes, Frank Hutter, and Yee Whye Teh. Neural ensemble search for uncertainty estimation and dataset shift. *arXiv preprint arXiv:2006.08573*, 2020.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3713–3722, 2019.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4320–4328, 2018.

APPENDIX: FEATURE KERNEL DISTILLATION

A FEATURE KERNEL DEPENDENCE WITH CROSS-ENTROPY LOSS

Suppose, we have n data points with fixed feature extractor $h = h(\boldsymbol{X}) \in \mathbb{R}^{n \times p}$, trainable last layer weights $W \in \mathbb{R}^{p \times C}$ and targets $Y \in \mathbb{R}^{n \times C}$.

In Section 2, we described how with Mean Squared Error loss the trained NN is precisely kernel ridge regression using the feature kernel, when there is ℓ_2 regularisation on the last layer weights. Hence, the feature kernel exactly determines the predictions. We now prove the same for cross entropy loss. That is to say, we wish to show that given the feature kernel k, the predictive probabilities/logits at the optimal W^* are independent of both W^* and h.

We define the loss, for some regularisation $\lambda > 0$ (purely to enforce strong convexity) by

$$\mathcal{L}(W) = \sum_{i=1}^{n} CE(y_i, h_i W) + \frac{\lambda}{2} \|W\|_2^2 ,$$

where $h_i \in \mathbb{R}^{1 \times p}$ is the i^{th} row of h and

$$CE(y, f) = -\log \frac{e^{f_y}}{\sum_{c \in [C]} e^{f_c}}, \tag{8}$$

is cross entropy loss.

Proposition 1. *Let feature extractor $h(\cdot)$, training inputs $\boldsymbol{X}$, training targets Y, ℓ_2 regularisation $\lambda > 0$ all be fixed. Suppose we are given a test point $\boldsymbol{x}'$ with features h', then the test prediction logits $h'W^*$ at optimal W^* can be expressed solely in terms of feature kernel evaluations $k(\cdot, \cdot) = \langle h(\cdot), h(\cdot) \rangle$.*

Proof. If we differentiate $\mathcal{L}(W)$ and set to zero we get:

$$\lambda W^*_{j,l} = \sum_{i=1}^{n} h_{i,j} \big[\mathbb{1}\{y_i = l\} - p_{i,l} \big] \tag{9}$$

where $p \in \mathbb{R}^{n \times C}$ is the result of applying softmax to each row of the logits $h'W^*$.

Let h_i denote the extracted features for training input $\boldsymbol{x}_i$. Now recall that $\langle h_i, h_j \rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the feature kernel evaluated at $\boldsymbol{x}_i, \boldsymbol{x}_j$. We multiply Eq. (9) by the test-data feature vector h' to give:

$$\lambda (h'W^*)_l = \sum_{i=1}^{n} k(\boldsymbol{x}_*, \boldsymbol{x}_i) \big[\mathbb{1}\{y_i = l\} - p_{i,l} \big] \tag{10}$$

but $h'W^*$ are precisely the logits for test point $\boldsymbol{x}_*$, and likewise $p_{i,l}$ is the vector of probability predictions at training point i. Hence, we could solve Eq. (10) numerically for logits/predictive probabilities given only the feature kernel (without $h(\cdot)$ or W^*), and we see that the feature kernel once again determines logit/prediction probabilities at the optimal last layer parameters like for squared error loss, albeit this time implicitly for cross entropy loss. □

B SETUP FOR TRAINING ON MULTI-VIEW DATA

Before we prove Theorem 2, which demonstrates the generalisation benefits of FKD theoretically, we need to recall the data and training setup of Allen-Zhu & Li (2020) for completeness and self-containedness. Unless otherwise stated, everything in this section (App. B) is a simplified version of the setup in Allen-Zhu & Li (2020). Our results hold in the more general version too, but we present the setup in a simplified setting here for readability and convenience, without sacrificing the key messages and intuitions: our focus is for our theoretical and practical contributions to guide each other and align as much as possible, as opposed to e.g. maximising the generality of our theory.

B.1 MULTI-VIEW DATA DISTRIBUTION

Recall that we consider a C-class classification problem over P-patch inputs, where each patch has dimension d, so our inputs are described by $\boldsymbol{x} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^P) \in (\mathbb{R}^d)^P$. For simplicity, we take $P = C^2$ and $d = \text{poly}(C)$ for a large polynomial. Like Allen-Zhu & Li (2020), we use $\tilde{O}, \tilde{\Theta}, \tilde{\Omega}$ to hide polylogarithmic factors in the number of classes, C, which we take to be sufficiently large.

We assume that each class $c \in [C]$ has exactly two attributes $v_{c,1}, v_{c,2} \in \mathbb{R}^d$, which are orthonormal for simplicity,[7] such that:

$$\mathcal{V} = \{v_{c,1}, v_{c,2}\}_{c\in[C]}$$

is the set of all attributes.

B.1.1 DATA GENERATING MECHANISM

Let our data distribution for a data pair $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$ be defined as $\mathcal{D} = \mu\mathcal{D}_s + (1-\mu)\mathcal{D}_m$, for multi-view & single-view distributions $\mathcal{D}_m$ & $\mathcal{D}_s$ respectively. $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$ are generated as follows:

1. Sample $\boldsymbol{y} \in [C]$ uniformly at random.
2. Sample a set $\mathcal{V}'(\boldsymbol{x})$ of attributes uniformly at random from $\{v_{c',1}, v_{c',2}\}_{c'\neq y}$ each with probability $\frac{s}{C}$ and denote $\mathcal{V}(X) = \mathcal{V}'(\boldsymbol{x}) \cup \{v_{y,1}, v_{y,2}\}$ as the set of attribute vectors used in data $\boldsymbol{x}$. We take $s = C^{0.2}$.

   Intuition:

   $\mathcal{V}'(\boldsymbol{x})$ correspond to the ambiguous attributes present in $\boldsymbol{x}$, such as the cat whose eyes look like car headlights. These $\mathcal{V}'$ are crucial in our proofs as the FKD regularisation between ambiguous images ensures that the student learns attributes it would have otherwise missed.

   For each $v \in \mathcal{V}(\boldsymbol{x})$, pick C_p (where C_p is a global constant) many disjoint patches in $[P]$ uniformly at random and denote this set as $\mathcal{P}_v(\boldsymbol{x})$. Denote $\mathcal{P}(\boldsymbol{x}) = \cup_{v\in\mathcal{V}(\boldsymbol{x})}\mathcal{P}_v(\boldsymbol{x})$: all other patches $p \notin \mathcal{P}(\boldsymbol{x})$ will contain noise only.
4. If $\boldsymbol{x}$ is **single** view, pick a value $\hat{l} = \hat{l}(\boldsymbol{x}) \in \{1, 2\}$ uniformly at random. $\hat{l}$ corresponds to the attribute $v_{y,\hat{l}}$ that is present in $\boldsymbol{x}$, with $v_{y,3-\hat{l}}$ missing.
5. For each $p \in \mathcal{P}_v(\boldsymbol{x})$ for some $v \in \mathcal{V}(\boldsymbol{x})$:

$$\boldsymbol{x}_p = z_p v + \sum_{v'\in\mathcal{V}} \alpha_{p,v'} v' + \xi_p$$

[7] As $d = \text{poly}(C)$ for a large polynomial, this isn't too far-fetched an assumption.

where $\alpha_{p,v'} \in [0, \frac{1}{C^{1.5}}]$ represents feature noise and $\xi_p \sim \mathcal{N}(0, \sigma_p^2 \mathbf{I}_d)$ is independent random noise, with $\sigma_p = \frac{1}{\sqrt{d}\text{polylog}(C)}$. The coefficients $z_p \geq 0$ satisfy:

(a) If $\boldsymbol{x}$ is **multi** view,

- When $v \in \{v_{y,1}, v_{y,2}\}$,

$$\begin{cases} \sum_{p \in \mathcal{P}_v(\boldsymbol{x})} z_p \in [1, 2) \\ \sum_{p \in \mathcal{P}_v(\boldsymbol{x})} z_p^4 = 1 \end{cases} \tag{11}$$

> **Intuition:**
> These conditions ensure that both attributes for class y are equally likely to be learnt, averaged over different random initialisations of parameters.

- When $v \in \mathcal{V}'(\boldsymbol{x})$,

$$\begin{cases} \sum_{p \in \mathcal{P}_v(\boldsymbol{x})} z_p = 0.4 \\ \sum_{p \in \mathcal{P}_v(\boldsymbol{x})} z_p^4 = \Theta(1) \end{cases} \tag{12}$$

(b) If $\boldsymbol{x}$ is **single** view,

- When $v = v_{y,\hat{l}}$, $\sum_{p \in \mathcal{P}_v(\boldsymbol{x})} z_p = 1$
- When $v = v_{y,3-\hat{l}}$, $\sum_{p \in \mathcal{P}_v(\boldsymbol{x})} z_p = C^{-0.2}$
- When $v \in \mathcal{V}'(\boldsymbol{x})$, $\sum_{p \in \mathcal{P}_v(\boldsymbol{x})} z_p = \Gamma$

where $\Gamma = \frac{1}{\text{polylog}(C)}$

> **Intuition:**
> This is where the single view name comes from, as $C^{-0.2} \ll 1$ we see that $v_{y,3-\hat{l}}$ is barely present in $\boldsymbol{x}$.

6. For each $p \in [P] \backslash \mathcal{P}_v(\boldsymbol{x})$:

$$\boldsymbol{x}_p = \sum_{v' \in \mathcal{V}} \alpha_{p,v'} v' + \xi_p$$

for feature noise $\alpha_{p,v'} \in [0, \frac{1}{C^{1.5}}]$ and $\xi_p \sim \mathcal{N}(0, \frac{1}{Cd}\mathbf{I}_d)$ is independent random noise.

> **Intuition:**
> One can think of the zero feature noise setting $\alpha_{p,v'} = 0 \ \forall p, v'$ for simplicity. But the general formulation above renders the problem unlearnable by linear classifier, as the maximum permissible feature noise across patches dominates the minimum possible signal: $\frac{P}{C^{1.5}} = C^{0.5} \gg 1$

Remark It is possible to allow more relaxed assumptions, e.g. on z_p, as in Allen-Zhu & Li (2020).

Training data Recall we have $\mathcal{D} = \mu\mathcal{D}_s + (1-\mu)\mathcal{D}_m$, so that a proportion $(1-\mu)$ of the data is multi-view. Our training data, $\hat{\mathcal{D}}$, is N independent samples from $\mathcal{D}$. Letting $\hat{\mathcal{D}} = \hat{\mathcal{D}}_m \cup \hat{\mathcal{D}}_s$ denote the split into multi and single view training data. We let $\mu = \frac{1}{\text{poly}(C)}$ and we suppose $|N| = \frac{C^{1.2}}{\mu}$ so that each label c appears at least $\Omega(1)$ in $\hat{\mathcal{D}}_s$.

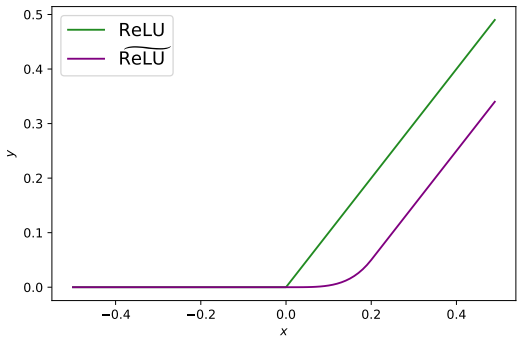


Figure 5: Comparison between ReLU & $\widetilde{\text{ReLU}}$, for $\varrho = 0.2$.

B.2 Smoothed ReLU CNN

Our theoretical analysis considers a single hidden layer CNN with sum-pooling, f, such that for class c:

$$f_c(\boldsymbol{x}) = \sum_{r=1}^{m} \sum_{p=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p \rangle), \quad \forall c \in [C].$$

For a threshold $\varrho = \frac{1}{\text{polylog}(C)}$, we define the smoothed ReLU function as:

$$\widetilde{\text{ReLU}}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \frac{z^4}{4\varrho^3} & \text{if } z \in [0, \varrho] \\ z - \frac{3}{4}\varrho & \text{if } z \geq \varrho \end{cases}$$

which has continuous monotonic gradient denoted as $\widetilde{\text{ReLU}}'$.

B.3 Hyperparameter values in setup

Hyperparameter values used in the theoretical setup are given in Table 3.

B.4 Standard training

We define our empirical loss by:

$$\mathcal{L} = \frac{1}{N} \sum_{i \in [N]} L(f(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

where L is the cross entropy loss defined in Eq. (8). We randomly initialise parameters $\boldsymbol{\theta}_{c,r}^0 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_0^2)$ for $\sigma_0^2 = \frac{1}{C}$.

Standard training in the theoretical analysis comprises of full-batch gradient descent on the empirical loss $\mathcal{L}$ with learning rate $\eta \leq \frac{1}{\text{poly}(C)}$ and for $T = \frac{\text{poly}(C)}{\eta}$ iterations.

C Proof of Theorem 2

We restate Theorem 2, where recall E denotes the teacher ensemble size and C is the number of classes:

Table 3: Hyperparameter values used in our theoretical analysis, corresponding to the setup of Allen-Zhu & Li (2020). (*) denotes undefined in our presentation, but appearing in Allen-Zhu & Li (2020), because the hyperparameter takes only one value in this work. We note that m is restricted to be $\text{polylog}(C)$ in the setting of Theorem 2.

Hyperparameter	Description	Value(s)
N	Training set size	$\frac{C^{1.2}}{\mu}$
μ	Proportion of single-view data	$\frac{1}{\text{poly}(C)}$
d	Input patch dimension	$\text{poly}(C)$
P	Number of patches per input	C^2
m	Number of channels per class	$[\text{polylog}(C), C]$
s	Out-of-class attribute sparsity	$C^{0.2}$
σ_0	Parameter initialisation standard deviation (std)	$C^{-0.5}$
σ_p	Input patch additive noise std	$\frac{1}{\sqrt{d}\text{polylog}(C)}$
Γ	Out-of-class attribute strength in single-view data.	$\frac{1}{\text{polylog}(C)}$
ϱ	$\widetilde{\text{ReLU}}$ threshold	$\frac{1}{\sqrt{d}\text{polylog}(C)}$
q (*)	$\widetilde{\text{ReLU}}$ mid-section exponent	4
ρ (*)	In-class weaker attribute strength in single-view data	$C^{-0.2}$

Theorem 2 (FKD improves student generalisation and is better with larger ensemble)**.** *Given an arbitrary $\epsilon > 0$. For any ensemble size E of teacher NNs trained as in Theorem 1 and sufficiently many classes C, for $m = \text{polylog}(C)$, with learning rate $\eta \leq \frac{1}{\text{poly}(C)}$, and training time $T^* = \frac{\text{poly}(C)}{\eta}$, the ensemble teacher knowledge can be distilled into a single student model $f^{(T^*)}$ using only teacher feature kernel $k_{\mathcal{T}}$, Eq.* (7)*, such that with probability at least $1 - e^{-\Omega(\log^2(C))}$:*

- *Training accuracy is perfect: For all $(\boldsymbol{x}, \boldsymbol{y}) \in \hat{\mathcal{D}}$, $\boldsymbol{y} = \text{argmax}_c\, f_c^{(T^*)}(\boldsymbol{x})$.*
- *Test accuracy is good: $\mathbb{P}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\left[\boldsymbol{y} \neq \text{argmax}_c\, f_c^{(T^*)}(\boldsymbol{x})\right] \leq (\frac{1}{2^{E+1}} + \epsilon)\mu$.*

Outline of proof of Theorem 2

1. We first analyse the feature kernel k for a single trained model, in App. C.1.
2. We then extend our results to an ensembled teacher model feature kernel in App. C.2.
3. We next outline our kernel distillation training scheme and provide a key result concerning how the student's parameters become increasingly correlated with the teacher's learnt views in Apps. C.3 and C.4
4. We combine all threads to prove the final result in App. C.5.

C.1 Feature kernel for a single trained model

To prove Theorem 2, we first calculate the feature kernel k_* for a single trained model with trained parameters $\boldsymbol{\theta}^*$ from initialisation $\boldsymbol{\theta}^0$. The point of this exercise is to show that the feature kernel k_* is able to detect whether two inputs $\boldsymbol{x}, \boldsymbol{x}'$ share a common attribute which is in the subset of attributes that was learnt by the trained parameters $\boldsymbol{\theta}^*$, due to random correlation with initialised parameters $\boldsymbol{\theta}^0$. This is formally shown in Lemma 1.

Recall the definition of the feature kernel k for our CNN architecture:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{c=1}^{C} \sum_{r=1}^{m} \sum_{p,p'=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p \rangle) \cdot \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}'_{p'} \rangle)$$

Let us first make a few more useful definitions. For $l = 1, 2$, we have:

Definition 2.

$$Z_{c,l}(\boldsymbol{x}) \stackrel{\text{def}}{=} \mathbb{1}\{v_{c,l} \in \mathcal{V}(\boldsymbol{x})\} \sum_{p \in \mathcal{P}_{v_{c,l}}(\boldsymbol{x})} z_p.$$

Intuition:

$Z_{c,l}(\boldsymbol{x})$ is a scalar constant describing the strength, in $\boldsymbol{x}$, of the presence of attribute $v_{c,l}$ (where c may either correspond to true class y or to a different class; the latter has the effect of producing 'soft labels' when using vanilla knowledge distillation).

Definition 3.

$$\Phi^*_{c,l} \stackrel{\text{def}}{=} \sum_{r \in [m]} [\langle \boldsymbol{\theta}^t_{c,r}, v_{c,l} \rangle]^+.$$

Definition 4.

$$\Upsilon^*_{c,l} \stackrel{\text{def}}{=} \sum_{r \in [m]} [\langle \boldsymbol{\theta}^*_{c,r}, v_{c,l} \rangle]^{+^2}.$$

Intuition:

$\Upsilon^*_{c,l}, \Phi^*_{c,l}$ are both parameter-dependent, data-independent scalars that describe the amount that attribute $v_{c,l}$ has been 'learnt' by the network parameters. $\Upsilon^*_{c,l}$ appears in the feature kernel whereas $\Phi^*_{c,l}$ appears in the function predictions f_c directly, for our particular architecture in Eq. (4) (Allen-Zhu & Li, 2020).

Lemma 1. *We have, for $\boldsymbol{x}, \boldsymbol{x}' \sim \hat{\mathcal{D}}_m$,*

$$k_*(\boldsymbol{x}, \boldsymbol{x}') = \sum_{(c,l):(c,3-l) \notin \mathcal{M}} \Upsilon^*_{c,l} Z_{c,l}(\boldsymbol{x}) Z_{c,l}(\boldsymbol{x}') \pm O\Big(\frac{s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}')}{\text{polylog}(C)}\Big) + \tilde{O}\Big(\frac{1}{C^{0.8}}\Big)$$

where

$$s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}') = |\{(c,l) : \ v_{c,l} \in \mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}') \text{ and } (c, 3-l) \notin \mathcal{M}\}|$$

*is the number of shared attributes of $\boldsymbol{x}$ and $\boldsymbol{x}'$, that are also members of the set of attributes learnt by the single trained model with parameters $\boldsymbol{\theta}^*$, and $\mathcal{M}$ is defined in **Fact A.e** below.*

Proof. We have

$$k_*(\boldsymbol{x}, \boldsymbol{x}') = \sum_{c=1}^{C} \sum_{r=1}^{m} \Big[\sum_{p=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}^*_{c,r}, \boldsymbol{x}_p \rangle) \cdot \sum_{p'=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}^*_{c,r}, \boldsymbol{x}'_{p'} \rangle) \Big].$$

First, define the hidden-layer activations for class c and channel r to be:

Definition 5.

$$\Psi_{c,r}(\boldsymbol{x}) \stackrel{\text{def}}{=} \sum_{p=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}^*_{c,r}, \boldsymbol{x}_p \rangle).$$

We know from Theorem C.2. of Allen-Zhu & Li (2020) that, for multi-view $\boldsymbol{x} \in \hat{\mathcal{D}}_m$ and some $c \in [C]$:

Fact A.a For every $p \in \mathcal{P}_{v_{c,l}}(\boldsymbol{x})$ and for $l = 1, 2$, we have: $\langle \boldsymbol{\theta}^*_{c,r}, \boldsymbol{x}_p \rangle = \langle \boldsymbol{\theta}^*_{c,r}, v_{c,l} \rangle z_p \pm \tilde{o}(\sigma_0)$

Fact A.b For every $p \in \mathcal{P}(\boldsymbol{x}) \backslash (\mathcal{P}_{v_{c,1}}(\boldsymbol{x}) \cup \mathcal{P}_{v_{c,2}}(\boldsymbol{x}))$, we have $|\langle \boldsymbol{\theta}^*_{c,r}, \boldsymbol{x}_p \rangle| \leq \tilde{O}(\sigma_0)$

Fact A.c For every $p \in [P] \backslash \mathcal{P}(\boldsymbol{x})$, we have $|\langle \boldsymbol{\theta}^*_{c,r}, \boldsymbol{x}_p \rangle| \leq \tilde{O}(\frac{\sigma_0}{\sqrt{C}})$

Fact A.d For every $r \in [m] \backslash \mathcal{M}^0_c$, every $l \in [2]$, it holds that $\langle \boldsymbol{\theta}^*_{c,r}, v_{c,l} \rangle \leq \tilde{O}(\sigma_0)$, where:

$$\mathcal{M}^0_c \stackrel{\text{def}}{=} \left\{ r \in [m] \middle| \exists l \in [2] : \langle \boldsymbol{\theta}^0_{c,r}, v_{c,l} \rangle \geq (1 - O(\frac{1}{\log(C)})) \cdot \max_{r \in [m]} [\langle \boldsymbol{\theta}^0_{c,r}, v_{c,l} \rangle]^+ \right\}.$$

Note from Proposition B.1 of Allen-Zhu & Li (2020), that $m_0 \stackrel{\text{def}}{=} |\mathcal{M}^0_c| = O(\log^5(C))$ with probability at least $1 - e^{-\Omega(\log^5(C))}$.

> **Intuition:**
> $\mathcal{M}^0_c$ denotes the key channels in $[m]$ which have 'won the lottery' and are relevant for class c, in that in the $C \to \infty$ limit the predictions for f_c are the same as if one forgets the other channels, as shown in Allen-Zhu & Li (2020).

Fact A.e For every $p \in \mathcal{P}_{v_{c,l}}(\boldsymbol{x})$ and $r \in [m]$, if $(c, 3-l) \in \mathcal{M}$, then we have: $|\langle \boldsymbol{\theta}^*_{c,r}, x_p \rangle| \leq \tilde{O}(\sigma_0)$, where:

$$\mathcal{M} \stackrel{\text{def}}{=} \left\{ (c, l) \in [C] \times [2] \middle| \max_{r \in [m]} [\langle \boldsymbol{\theta}^0_{c,r}, v_{c,l} \rangle]^+ \geq \big(1 + \frac{1}{\log^2(m)}\big) \cdot \max_{r \in [m]} [\langle \boldsymbol{\theta}^0_{c,r}, v_{c,3-l} \rangle]^+ \right\}$$

> **Intuition:**
> $\mathcal{M}$ denotes the data attributes $v_{c,l}$ which are more likely to be learnt by the NN parameters (compared to their fellow class attributes $v_{c,3-l}$) because of correlations of the initial parameters with such attributes. So, **Fact A.e** is saying that if $(c, 3-l) \in \mathcal{M}$, then the attribute (c, l) is not learnt at all during standard single model training.

Thus,

$$\Psi_{c,r}(\boldsymbol{x}) = \sum_{l=1}^{2} \mathbb{1}\{v_{c,l} \in \mathcal{V}(\boldsymbol{x})\} \cdot \sum_{p \in \mathcal{P}_{v_{c,l}}(\boldsymbol{x})} \widetilde{\text{ReLU}}\big(\langle \boldsymbol{\theta}^*_{c,r}, \boldsymbol{x}_p \rangle\big) \tag{13}$$

$$+ \sum_{p \in \mathcal{P}(\boldsymbol{x}) \backslash \bigcup_l \{\mathcal{P}_{v_{c,l}}(\boldsymbol{x})\}} \widetilde{\text{ReLU}}\big(\tilde{O}(\sigma_0)\big) \tag{14}$$

$$+ \sum_{p \in [P] \backslash \mathcal{P}(\boldsymbol{x})} \widetilde{\text{ReLU}}\big(\tilde{O}(\frac{\sigma_0}{\sqrt{C}})\big). \tag{15}$$

First, note that $|\mathcal{P}_{v_{c',l'}}(\boldsymbol{x})| = C_p$ is constant $\forall c', l'$. From **Fact A.b**, Eq. (14) can be easily seen to be $\tilde{O}(\sigma_0^4 s) = \tilde{O}(C^{-1.8})$ as $\sigma_0^2 = \frac{1}{C}$ and $s = C^{0.2}$, and likewise Eq. (15) can be seen to be $\tilde{O}((\frac{\sigma_0}{\sqrt{C}})^4 P) = \tilde{O}(C^{-2})$ as $P = C^2$, by **Fact A.c**. We note here that summing these equations over m and C will be bounded above by $O(\frac{1}{\text{polylog}(C)})$.

Now, let's consider Eq. (13). Let notation $v_{c,1}, v_{c,2} \in S$ denote either $v_{c,1}$ or $v_{c,2} \in S$ for some set S, and $v_{c,1}, v_{c,2} \notin S$ denote neither $v_{c,1}$ nor $v_{c,2} \in S$.

There are three cases to consider:

1. If $v_{c,1}, v_{c,2} \notin \mathcal{V}(\boldsymbol{x})$, then Eq. (13) is zero.
2. Else if $\exists l \in [2]$ such that $v_{c,l} \in \mathcal{V}(\boldsymbol{x})$ and $(c, 3-l) \in \mathcal{M}$, then by **Fact A.e**, we have Eq. (13) is $\tilde{O}(\sigma_0^4)$=$\tilde{O}(C^{-2})$.

   Intuition:
   This setting is when only attribute (c, l) appears in $\boldsymbol{x}$, but attribute $(c, 3-l)$ was dominant at initialisation so (c, l) has not been learnt by $\boldsymbol{\theta}^*$.

3. Else, we have that:

$$\text{Eq. (13)} = \sum_{l=1}^{2} \mathbb{1}\{v_{c,l} \in \mathcal{V}(\boldsymbol{x})\} \cdot \sum_{p \in \mathcal{P}_{v_{c,l}}(\boldsymbol{x})} \widetilde{\text{ReLU}}\big(\langle \boldsymbol{\theta}^*_{c,r}, \boldsymbol{x}_p \rangle\big).$$

Putting this all together with **Fact A.a**, we see that

$$\Psi_{c,r}(\boldsymbol{x}) = \sum_{l=1}^{2} \mathbb{1}\{(c,l) \in s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x})\} \sum_{p \in \mathcal{P}_{v_{c,l}}(\boldsymbol{x})} \widetilde{\text{ReLU}}\big(\langle \boldsymbol{\theta}^*_{c,r}, v_{c,l} \rangle z_p + \tilde{o}(\sigma_0)\big) + \tilde{O}(\frac{1}{C^{1.8}}).$$

Now, if $r \notin \mathcal{M}_c^0$ (i.e. neuron r is not dominant at initialisation), and $(c,1), (c,2) \in s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x})$, the by **Fact A.d**, we have that $\Psi_{c,r}(\boldsymbol{x})$=$\tilde{O}(C^{-1.8})$.

On the other hand, recall $\varrho = \frac{1}{\text{polylog}(C)}$ and $m_0 = O(\log^5(C))$. And also, recall that if $z > \varrho$, then $\widetilde{\text{ReLU}}(z) = z + O(\varrho)$. Moreover, by Claim C.11 of Allen-Zhu & Li (2020) we know that $\exists r', l'$ such that $\langle \boldsymbol{\theta}^*_{c,r'}, v_{c,l'} \rangle = \Omega(\frac{1}{m_0})$. Hence, for any $r \in \mathcal{M}_c^0$, we know that either:

1. $\Psi_{c,r}(\boldsymbol{x}) = \sum_{l=1}^{2} \mathbb{1}\{(c,l) \in s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x})\}\big(\langle \boldsymbol{\theta}^*_{c,r}, v_{c,l} \rangle Z_{c,l}(\boldsymbol{x}) \pm O(\varrho)\big) + \tilde{O}(\frac{1}{C^{1.8}})$, or
2. $\Psi_{c,r}(\boldsymbol{x}) = \mathbb{1}\{(c,1), (c,2) \in s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x})\} O(\varrho) + \tilde{O}(\frac{1}{C^{1.8}})$.

Moreover, $\Psi_{c,r}(\boldsymbol{x}) = \tilde{O}(1)\ \forall r$, by e.g. Lemma C.21 of Allen-Zhu & Li (2020). And so it can be seen, for ϱ small enough we have:

$$\begin{aligned}
&\sum_{r=1}^{m} \Psi_{c,r}(\boldsymbol{x}) \Psi_{c,r}(\boldsymbol{x}') \\
&= \sum_{l=1}^{2} \mathbb{1}\{(c, 3-l) \notin \mathcal{M}\} Z_{c,l}(\boldsymbol{x}) Z_{c,l}(\boldsymbol{x}') [\sum_{r=1}^{m} \big[\langle \boldsymbol{\theta}^*_{c,r}, v_{c,l} \rangle\big]^{+^2} \pm O(\frac{1}{\text{polylog}(C)})] + \tilde{O}(\frac{1}{C^{1.8}}) \\
&= \sum_{l=1}^{2} \mathbb{1}\{(c, 3-l) \notin \mathcal{M}\} Z_{c,l}(\boldsymbol{x}) Z_{c,l}(\boldsymbol{x}') [\Upsilon^*_{c,l} \pm O(\frac{1}{\text{polylog}(C)})] + \tilde{O}(\frac{1}{C^{1.8}})
\end{aligned}$$

where we also use **Fact A.e** above, such that it is not possible for both $\langle \boldsymbol{\theta}^*_{c,r}, v_{c,l}\rangle^+$ and $\langle \boldsymbol{\theta}^*_{c,r}, v_{c,3-l}\rangle^+$ to be large.[8] Note also from e.g. the proof of Allen-Zhu & Li (2020) Theorem 1, that if $(c, 3-l) \notin \mathcal{M}$, then $\max_r \langle \boldsymbol{\theta}^*_{c,r}, v_{c,l}\rangle = \tilde{\Theta}(1)$.

Thus, we see that the contribution to k from the m class-c channels/neurons is:

$$k_c(\boldsymbol{x}, \boldsymbol{x}') \stackrel{\text{def}}{=} \sum_{r=1}^{m} \Psi_{c,r}(\boldsymbol{x})\Psi_{c,r}(\boldsymbol{x}') \tag{16}$$

$$= \begin{cases} \tilde{\Theta}(1) & \text{if } (c,1),(c,2) \in s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}') \\ \tilde{O}(\frac{1}{C^{1.8}}) & \text{else} \end{cases} \tag{17}$$

Summing k_c over $[C]$ completes the proof of the lemma. □

We now use Lemma 1 to analyse k_* and $\rho_*(\boldsymbol{x}, \boldsymbol{x}') = \frac{k_*(\boldsymbol{x},\boldsymbol{x}')}{\sqrt{k_*(\boldsymbol{x},\boldsymbol{x})k_*(\boldsymbol{x}',\boldsymbol{x}')}}$.

First we look at $\Upsilon^*_{c,l}$. Recall that $\langle \boldsymbol{\theta}^*_{c,r}, v_{c,l}\rangle = \tilde{O}(1)$ for all $c \in [C], l \in [2], r \in [m]$, and likewise so is $\mathcal{M}^0_c$.

More specifically,

- If $(c,l) \in \mathcal{M}$, then we have from the proof of Theorem 1 in Allen-Zhu & Li (2020) that $\sum_r \langle \boldsymbol{\theta}^*_{c,r}, v_{c,l}\rangle^+ \geq \Omega(\log(C))$, and so $\max_r \langle \boldsymbol{\theta}^*_{c,r}, v_{c,l}\rangle^+ = \tilde{\Theta}(1)$.
- If neither $(c,1)$ nor $(c,2)$ are in $\mathcal{M}$, then Claim C.10 of Allen-Zhu & Li (2020) shows us that both $\max_r \langle \boldsymbol{\theta}^*_{c,r}, v_{c,1}\rangle^+, \max_r \langle \boldsymbol{\theta}^*_{c,r}, v_{c,2}\rangle^+$ are $\tilde{\Theta}(1)$.

Combining these facts, we have that:

$$\Upsilon^*_{c,l} = \tilde{\Theta}(1) \qquad \text{if } (c, 3-l) \notin \mathcal{M}$$

and so from Lemma 1, we have:

$$k_*(\boldsymbol{x}, \boldsymbol{x}') = \sum_{(c,l):(c,3-l)\notin\mathcal{M}} \Upsilon^*_{c,l} Z_{c,l}(\boldsymbol{x}) Z_{c,l}(\boldsymbol{x}') \pm O\big(\frac{s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}')}{\text{polylog}(C)}\big) + \tilde{O}(C^{-0.8}) \tag{18}$$

$$= \tilde{\Theta}(1) \sum_{(c,l):(c,3-l)\notin\mathcal{M}} Z_{c,l}(\boldsymbol{x}) Z_{c,l}(\boldsymbol{x}') \pm O\big(\frac{s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}')}{\text{polylog}(C)}\big) + \tilde{O}(C^{-0.8}) \tag{19}$$

$$= \tilde{\Theta}(s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}')) \pm O\big(\frac{s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}')}{\text{polylog}(C)}\big) + \tilde{O}(C^{-0.8}). \tag{20}$$

Finally, we arrive at an expression for the correlation kernel ρ_* of a trained single model by

$$\rho_*(\boldsymbol{x}, \boldsymbol{x}') = \tilde{\Theta}\big(\frac{s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}')}{\sqrt{s_{\mathcal{M}}(\boldsymbol{x}) s_{\mathcal{M}}(\boldsymbol{x}')}}\big)\big(1 + O(\frac{1}{\text{polylog}(C)})\big) + \tilde{O}(\frac{1}{C^{0.8}}) \tag{21}$$

where we define that $s_{\mathcal{M}}(\boldsymbol{x}) = s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x})$ is the number of attributes in $\boldsymbol{x}$ that have been learnt by the trained network $\boldsymbol{\theta}^*$, and is $s(1 \pm o(1))$ with high probability.

[8] Unless neither $(c,1)$ nor $(c,2) \in \mathcal{M}$ for a given c, but that only occurs in $o(C)$ classes, and does not change the order of e.g. $s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}')$ which is what we really care about.

Intuition:

Compare ρ_* in Eq. (21) to the 'soft' probability labels $p^\tau \in \mathbb{R}^C$ of Allen-Zhu & Li (2020) (Claim F.4) with temperature $\tau = \frac{1}{\log^2(C)}$:

$$p_c^\tau(\boldsymbol{x}) = \begin{cases} \frac{1}{s(\boldsymbol{x})} & \text{if } v_{c,1} \text{ or } v_{c,2} \text{ is in } \mathcal{V}(\boldsymbol{x}) \\ 0 & \text{else} \end{cases}$$

where $s(\boldsymbol{x})$ is the number of indices $c \in [C]$ such that $v_{c,1}$ or $v_{c,2}$ is in $\mathcal{V}(\boldsymbol{x})$. Note, the setting of Allen-Zhu & Li (2020) is with a large $\tilde{\Omega}(1)$ ensemble, so every attribute is learnt (akin to $\mathcal{M}$ being empty for us). In the case that $\mathcal{M} = \{\}$ being empty, if $\boldsymbol{x} \neq \boldsymbol{x}'$ and they share at least one feature, then from App. B.1.1 with high probability they will share exactly one feature, so that $s_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{x}') = 1$. Moreover, $s(\boldsymbol{x}) = s_{\mathcal{M}}(\boldsymbol{x}) \approx s_{\mathcal{M}}(\boldsymbol{x}')$, hence we see that Eq. (21) matches roughly with $p_c^\tau(\boldsymbol{x})$, but without the need for a temperature hyperparameter.

We see that vanilla KD learns new attributes by comparing a single data point $\boldsymbol{x}$ between classes, and giving larger target labels to the classes where ambiguous attributes learnt by the teacher are present in $\boldsymbol{x}$. On the other hand, ρ_* gives higher values to data pairs $\boldsymbol{x}, \boldsymbol{x}'$ that share attributes that have been learnt by the trained model, and as we will see later this is how FKD learns new attributes in the student.

C.2 Ensembled teacher

To summarise what we have done so far in App. C.1, we have seen in Eqs. (20) and (21) that it is possible, for a single trained model $\boldsymbol{\theta}^*$, to simplify both the feature kernel $k_*(\boldsymbol{x}, \boldsymbol{x}')$ and correlation kernel $\rho_*(\boldsymbol{x}, \boldsymbol{x}')$ in terms of the number of shared attributes between $\boldsymbol{x}, \boldsymbol{x}'$ which are also learnt by the trained model. The set of attributes learnt by the single trained model is captured by the set

$$\mathcal{M} = \left\{ (c,l) \in [C] \times [2] \,\middle|\, \max_{r \in [m]} [\langle \boldsymbol{\theta}_{c,r}^0, v_{c,l} \rangle]^+ \geq \left(1 + \frac{1}{\log^2(m)}\right) \cdot \max_{r \in [m]} [\langle \boldsymbol{\theta}_{c,r}^0, v_{c,3-l} \rangle]^+ \right\}$$

where $\boldsymbol{\theta}_0$ was the random parameter initialisation for $\boldsymbol{\theta}^*$. From **Fact A.e**, we know that if $(c, 3-l) \in \mathcal{M}$, then the attribute $v_{c,l}$ has not been learnt by the network.

Consider now an ensemble of $E = \Theta(1)$ independently trained networks, $\{\boldsymbol{\theta}_e^*\}_{e=1}^E$, with an averaged feature kernel:

$$k_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{E} \sum_{e=1}^{E} k_e^*(\boldsymbol{x}, \boldsymbol{x}').$$

Suppose $\{\boldsymbol{\theta}^{e,0}\}_{e=1}^E$ denotes the corresponding independent parameter initialisations. Then, for each $e \in [E]$, let us define:

$$\mathcal{M}_e \stackrel{\text{def}}{=} \left\{ (c,l) \in [C] \times [2] \,\middle|\, \max_{r \in [m]} [\langle \boldsymbol{\theta}_{c,r}^{e,0}, v_{c,l} \rangle]^+ \geq \left(1 + \frac{1}{\log^2(m)}\right) \cdot \max_{r \in [m]} [\langle \boldsymbol{\theta}_{c,r}^{e,0}, v_{c,3-l} \rangle]^+ \right\}.$$

Note that these $\mathcal{M}_e$ are completely independent sets due to the independent initialisations, and also by Proposition B.2 of Allen-Zhu & Li (2020), we know that

$$\mathbb{P}\big[(c,1) \text{ or } (c,2) \in \mathcal{M}_e\big] \geq 1 - o(1) \qquad \forall c \in [C], e \in [E].$$

Therefore,

$$C \geq |\mathcal{M}_e| \geq C(1 - o_p(1)) \qquad \forall e \in [E].$$

Moreover, Eq. (11) & Proposition B.2 of Allen-Zhu & Li (2020) tell us that each of the two attributes $v_{c,1}, v_{c,2}$ are equally likely to be in $\mathcal{M}_e$ (and so learnt in the multi-view setup), This means that:

$$|\bigcap_{e=1}^{E} \mathcal{M}_e| = \frac{1}{2^{E-1}} C(1 - o_p(1)).$$

Define $\mathcal{M}_{\mathcal{T}} = \bigcap_{e=1}^{E} \mathcal{M}_e$. From Eq. (19) and the definition of $k_{\mathcal{T}}$, we see that:

$$k_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}') = \tilde{\Theta}(1) \sum_{(c,l)} \mathbb{1}\{(c, 3-l) \notin \mathcal{M}_{\mathcal{T}}\} Z_{c,l}(\boldsymbol{x}) Z_{c,l}(\boldsymbol{x}') \pm O\big(\frac{s_{\mathcal{M}_{\mathcal{T}}}(\boldsymbol{x}, \boldsymbol{x}')}{\text{polylog}(C)}\big) + \tilde{O}(C^{-0.8}) \tag{22}$$

$$= \tilde{\Theta}(s_{\mathcal{M}_{\mathcal{T}}}(\boldsymbol{x}, \boldsymbol{x}')) \pm O\big(\frac{s_{\mathcal{M}_{\mathcal{T}}}(\boldsymbol{x}, \boldsymbol{x}')}{\text{polylog}(C)}\big) + \tilde{O}(C^{-0.8}) \tag{23}$$

where for the reader's convenience, we redefine:

$$s_{\mathcal{M}_{\mathcal{T}}}(\boldsymbol{x}, \boldsymbol{x}') = \{(c, l) : \, v_{c,l} \in \mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}') \text{ and } (c, 3-l) \notin \mathcal{M}_{\mathcal{T}}\}.$$

Intuition:

We see that only for those attributes (c, l) such that $(c, 3-l) \in \mathcal{M}_{\mathcal{T}}$ does the ensembled teacher $k_{\mathcal{T}}$ miss the fact that we should have a strong $\tilde{\Theta}(1)$ kernel value between $\boldsymbol{x}, \boldsymbol{x}'$. This is when $|\mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}')| = \{v_{c,l}\}$ is non-empty (or in other words, when $s_{\mathcal{M}_{\mathcal{T}}}(\boldsymbol{x}, \boldsymbol{x}') \neq |\mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}')|$), and hence there should be a large kernel value $k_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}')$.
So we see that only $|\mathcal{M}_{\mathcal{T}}|$ of the attributes are not learnt by the teacher, which is a fraction $\frac{|\mathcal{M}_{\mathcal{T}}|}{2C} = \frac{1}{2^E}(1 - o(1))$ of all the attributes. These missed attributes are where the $\frac{1}{2^{E+1}}$ test error in Theorem 2 comes from (teacher ensemble of size E, and plus 1 for the attributes learnt from the student's initialisation too).

What's more, we can decompose the teacher's feature kernel $k_{\mathcal{T}} = \sum_c k_c^{\mathcal{T}}$ into contributions $k_c^{\mathcal{T}}$ from each class c, like in Eq. (16). From Eq. (17), we see that the contribution to the teacher's feature kernel from class , for $\boldsymbol{x} \neq \boldsymbol{x}'$;

$$k_c^{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} \tilde{\Theta}(1) & \text{if } (c,1), (c,2) \in s_{\mathcal{M}_{\mathcal{T}}}(\boldsymbol{x}, \boldsymbol{x}') \\ \tilde{O}(\frac{1}{C^{1.8}}) & \text{else,} \end{cases} \tag{24}$$

is able to decipher between whether or not $\boldsymbol{x}, \boldsymbol{x}'$ share an attribute from class c for all attributes apart from those (c, l) such that $(c, 3-l) \in \mathcal{M}_{\mathcal{T}}$.

C.3 Training scheme for FKD

We note at this point that we are morally done in terms of proving Theorem 2, with Eqs. (23) and (24), our key results telling us that the (ensemble) teacher kernel $k_{\mathcal{T}}$ can identify when two inputs share common attributes that have been learnt by the (ensemble) teacher, and more specifically that $k_c^{\mathcal{T}}$ can do so when said common attribute is from class c.

What remains is a repackaging of the proof techniques of Allen-Zhu & Li (2020) (particularly for their Theorem 4 regarding self-distillation), that knowledge distillation (this time only using feature kernels instead of temperature-scaled logits, and with explicit dependence on teacher ensemble size) can improve generalisation performance of a student.

For convenience, the theoretical analysis of Allen-Zhu & Li (2020) introduces some slight discrepancies between the actual practical weight updates of vanilla KD Hinton et al. (2015), i.e. the gradients of:

$$\tilde{\mathcal{L}} = \mathcal{L} + \lambda \frac{1}{N} \sum_i L(\frac{f(\boldsymbol{x}_i)}{\tau}, \frac{f_{\mathcal{T}}(\boldsymbol{x}_i)}{\tau}),$$

and the weight updates in their theoretical exposition. Namely,

1. The authors assume that a temperature-dependent threshold caps the logits to give soft labels:
$$p_c^{\tau}(\boldsymbol{x}) = \frac{e^{\min\{\tau^2 f_c(\boldsymbol{x}), 1\}/\tau}}{\sum_{j\in[C]} e^{\min\{\tau^2 f_j(\boldsymbol{x}), 1\}/\tau}}.$$
2. The authors truncate the negative part of the gradient of the KD regularisation to only encourage logits to increase not decrease, with weight updates for $\boldsymbol{\theta}_{c,r}$ on input $\boldsymbol{x}$:
$$-\Delta\boldsymbol{\theta}_{c,r}^t \stackrel{\text{def}}{=} \boldsymbol{\theta}_{c,r}^t - \boldsymbol{\theta}_{c,r}^{t+1} \propto \nabla_{\boldsymbol{\theta}_{c,r}} \mathcal{L} + \eta \frac{1}{N} \sum_i \left(p_c^{\tau}(\boldsymbol{x}) - p_c^{\tau,\mathcal{T}}(\boldsymbol{x})\right)^{-} \nabla_{\boldsymbol{\theta}_{c,r}} f_c(\boldsymbol{x})$$
where $p^{\tau,\mathcal{T}}$ are the temperature-scaled teacher labels.
3. The authors scale the output of both student and teacher models by a (polylogarithmic) factor, in order to ensure that both reach the threshold to give soft labels in Item 1. above.
4. Self-distillation (Furlanello et al., 2018; Zhang et al., 2019) distils a single teacher and a single student of same architecture into the student, like an ensemble of size 2 (student+teacher). Allen-Zhu & Li (2020) modify the training scheme for their theoretical analysis of self-distillation so that the student is first trained on its own in order separate learning its own attributes/features from those of the teacher. Our analysis covers a similar scheme.

These modifications are justified in that they make the theoretical analysis more convenient, whilst illustrating the main mechanisms by which KD works, which is to share 'dark knowledge' that is held in the teacher (in the form of the multi-view attributes that have been acquired by the teacher due to its parameter initialisation), with the student.

In the same vein, we now introduce some modifications to the practical implementation of FKD we propose in Alg. 1 to aid our theoretical analysis, and describe the main mechanisms by which FKD works, corroborating our initial analyses in Section 2 and App. A about how the feature kernel is a crucial object in any NN and captures all the 'dark knowledge' that a teacher network could possess in the multi-view data setting.

It is likely possible to extend our proof of Theorem 2 with different modifications/training schemes, but given that the focus of this work is to introduce FKD as a principled alternative to vanilla KD with certain advantages such as prediction-space independence, and that the multi-view setting we consider is a plausible simplification of real world data (as demonstrated in Allen-Zhu & Li (2020)), we leave this to future work. We stress that any simplifications to the update rule in Alg. 1 for this section can be efficiently computed, only requiring access to pairwise evaluations of the student and teacher feature kernels, if need be.

Modified training regime for FKD

1. We first suppose that the student is trained as standard (as in App. B.4) for $T_1 = \frac{\text{poly}(C)}{\eta}$ steps, and learns its own subset of attributes $\mathcal{M}_{\mathcal{S}}$, dependent on its initialisation $\boldsymbol{\theta}_s^0$, before being trained with the FKD objective:

Intuition:

This mirrors the self-distillation setup of Allen-Zhu & Li (2020) Theorem 4. The idea being that the student first learns $\mathcal{M}_{\mathcal{S}}$ before picking up the other attributes that the teacher has access to.

2. For a given feature kernel k, we threshold the feature kernel k based on value, to define a modification $\tilde{k}$ such that

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} 1 & \text{if } k(\boldsymbol{x}, \boldsymbol{x}') \geq \frac{1}{m^2} \\ 0 & \text{else} \end{cases}$$

Intuition:

This condition delineates between the setting where $\boldsymbol{x}, \boldsymbol{x}'$ share common attributes learnt by student parameters $\boldsymbol{\theta}^{T_1}$ in the initial phase of training (i.e. delineates between whether $s_{\mathcal{M}_s}(\boldsymbol{x}, \boldsymbol{x}')$ nonempty or empty).

To see this: note that if $v_{c,l} \in \mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}')$ and $(c, 3-l) \notin \mathcal{M}_{\mathcal{S}}$ then we know from Allen-Zhu & Li (2020) that $\Phi_{c,l}^{T_1} \geq \Omega(\log(C))$.
Hence $\max_r \langle \boldsymbol{\theta}_{c,r}^*, v_{c,l} \rangle \geq \Omega(\log^{-4}(C))$ as the number of active neurons $m_0 = |\mathcal{M}_c^0| = O(\log^5 C)$, and so it's easy to see that for large enough m we have $k(\boldsymbol{x}, \boldsymbol{x}') \geq \frac{1}{m^2}$ via Lemma 1.
On the other hand, if $\{v_{c,l}\} = \mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}')$ and $(c, 3-l) \in \mathcal{M}_{\mathcal{S}}$ then from Lemma 1 we know that $k(\boldsymbol{x}, \boldsymbol{x}') = \tilde{O}(C^{-0.8}) \ll \frac{1}{m^2}$.

3. Similar to Allen-Zhu & Li (2020), we also truncate our FKD regularisation to only encourage kernel values to increase, and not decrease. For any input pair $\boldsymbol{x}_1, \boldsymbol{x}_2$, we have parameter update:

$$-\Delta \boldsymbol{\theta}_{c,r}(\boldsymbol{x}_1, \boldsymbol{x}_2) \propto \big(\tilde{k}_c(\boldsymbol{x}_1, \boldsymbol{x}_2) - \tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2)\big)^{-} \big[\sum_{j \in \{1,2\}} \Psi_{c,r}(\boldsymbol{x}_j) \nabla_{\boldsymbol{\theta}_{c,r}} \Psi_{c,r}(\boldsymbol{x}_{3-j}) \big] \tag{25}$$

where recall

$$\Psi_{c,r}(\boldsymbol{x}) \stackrel{\text{def}}{=} \sum_{p=1}^{P} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p \rangle) \quad \text{so that} \quad \nabla_{\boldsymbol{\theta}_{c,r}} \Psi_{c,r}(\boldsymbol{x}) = \sum_{p=1}^{P} \widetilde{\text{ReLU}}'(\langle \boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p \rangle) \boldsymbol{x}_p$$

and

$$k_c(\boldsymbol{x}, \boldsymbol{x}') \stackrel{\text{def}}{=} \sum_{r=1}^{m} \Psi_{c,r}(\boldsymbol{x}) \Psi_{c,r}(\boldsymbol{x}')$$

were defined in Definition 5 and Eq. (16).

Intuition:

If the loss was

$$(k_c(\boldsymbol{x}_1, \boldsymbol{x}_2) - k_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2))^2$$

then the gradient with respect to $\boldsymbol{\theta}_{c,r}$ would be:

$$\big(k_c(\boldsymbol{x}_1, \boldsymbol{x}_2) - k_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2)\big) \big[\sum_{j \in \{1,2\}} \Psi_{c,r}(\boldsymbol{x}_j) \nabla_{\boldsymbol{\theta}_{c,r}} \Psi_{c,r}(\boldsymbol{x}_{3-j}) \big]$$

so the only differences with Eq. (25) are truncating $\big(\tilde{k}_c(\boldsymbol{x}_1, \boldsymbol{x}_2) - \tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2)\big)^{-}$ and also the thresholding to obtain $\tilde{k}$.

To summarise, after training the student on its own for T_1 steps (such that we are in the setting of Theorem 1) to reach parameters $\boldsymbol{\theta}_s^{T_1}$, we update for $T_2 = \frac{\text{poly}(C)}{\eta}$ steps as (hiding $\mathcal{S}$ subscript):

$$\Delta\boldsymbol{\theta}_{c,r}^t = -\eta\mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{x}_2\sim\hat{\mathcal{D}}^2}\left[\left(\tilde{k}_c(\boldsymbol{x}_1,\boldsymbol{x}_2) - \tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1,\boldsymbol{x}_2)\right)^- \sum_{j\in\{1,2\}} \Psi_{c,r}(\boldsymbol{x}_j)\nabla_{\boldsymbol{\theta}_{c,r}}\Psi_{c,r}(\boldsymbol{x}_{3-j})\right] \tag{26}$$

C.4 Feature correlation growths

We now seek to analyse to what extent the attributes $\{v_{c,l}\}_{c,l}$ are learnt during our T_2 FKD training steps. The central objects describing how much $v_{c,l}$ has been learnt by parameters $\boldsymbol{\theta}$ are:

$$\Phi_{c,l}^t \overset{\text{def}}{=} \sum_{r\in[m]} [\langle\boldsymbol{\theta}_{c,r}^t, v_{c,l}\rangle]^+ \qquad \text{and} \qquad \Phi_c^t \overset{\text{def}}{=} \sum_{l\in[2]} \Phi_{c,l}^t$$

as well as $\Psi_{c,r}(\boldsymbol{x})$ as defined above.

Intuition:

$\Phi_{c,l}$ is a data-independent quantity that reflects the strength of correlation with feature $v_{c,l}$ by parameters $\boldsymbol{\theta}$. On the other hand, $\Psi_{c,r}(\boldsymbol{x})$ is a data-dependent quantity that reflects the activation of channel r for class c with input $\boldsymbol{x}$.

Also recall that

$$Z_{c,l}(\boldsymbol{x}) \overset{\text{def}}{=} \mathbb{1}\{v_{c,l}\in\mathcal{V}(\boldsymbol{x})\} \sum_{p\in\mathcal{P}_{v_{c,l}}(\boldsymbol{x})} z_p$$

and define $V_{c,r,l}(\boldsymbol{x})$ (which is convenient for calculating the size of gradient updates for $\boldsymbol{\theta}_{c,r}$):

Definition 6.

$$V_{c,r,l}(\boldsymbol{x}) \overset{\text{def}}{=} \mathbb{1}\{v_{c,l}\in\mathcal{V}(\boldsymbol{x})\} \sum_{p\in\mathcal{P}_{v_{c,l}(\boldsymbol{x})}} \widetilde{\text{ReLU}}'(\langle\boldsymbol{\theta}_{c,r},\boldsymbol{x}_p\rangle) z_p$$

Like how Lemma 1 simplified the feature kernel in terms of data-dependent $Z_{c,l}(\boldsymbol{x})$ and data-independent $\Upsilon_{c,l}$, we have a result from Allen-Zhu & Li (2020) to simplify function predictions f_c in terms of $Z_{c,l}(\boldsymbol{x})$ and $\Phi_{c,l}$:

Claim 1 (Claim F.7 from Allen-Zhu & Li (2020))**.** *For every $t \le T_1 + T_2$, every $c \in [C]$, every $(\boldsymbol{x},\boldsymbol{y}) \in \hat{\mathcal{D}}$ (or every test sample $(\boldsymbol{x},\boldsymbol{y}) \sim \mathcal{D}$ with probability $1 - e^{-\Omega(\log^2(C))}$):*

$$f_c^t(\boldsymbol{x}) = \sum_{l\in[2]} \left(\Phi_{c,l}^t \times Z_{c,l}^t(\boldsymbol{x})\right) \pm O(\frac{1}{\text{polylog}(C)})$$

We also have the following facts from Allen-Zhu & Li (2020) regarding the correlation of gradient $\nabla_{\boldsymbol{\theta}_{c,r}}\Psi_{c,r}^t(\boldsymbol{x})$ with $v_{c,l}$ for $(\boldsymbol{x},\boldsymbol{y}) \in \hat{\mathcal{D}}, l \in [2]$ and $r \in [m]$:

Claim 2 (c.f. Claim F.6 of Allen-Zhu & Li (2020))**.** *For every $t \le T_1 + T_2$, for every $(\boldsymbol{x},\boldsymbol{y}) \in \hat{\mathcal{D}}$, every $c \in [C]$, $r \in [m]$ and $l \in [2]$:*

- *If $v_{c,1}, v_{c,2} \in \mathcal{V}(\boldsymbol{x})$, then $\langle\nabla_{\boldsymbol{\theta}_{c,r}}\Psi_{c,r}^t(\boldsymbol{x}), v_{c,l}\rangle \ge \left(V_{c,r,l}(\boldsymbol{x}) - \tilde{O}(\sigma_p P)\right)$*
- $\langle\nabla_{\boldsymbol{\theta}_{c,r}}\Psi_{c,r}^t, v_{c,l}\rangle \le \left(\mathbb{1}\{(v_{c,l}\in\mathcal{V}(\boldsymbol{x})\}V_{c,r,l}(\boldsymbol{x}) + \tilde{O}(C^{-2})\right)$
- *For every $i \ne c$, $|\langle -\nabla_{\boldsymbol{\theta}_{c,r}}\Psi_{c,r}^t(\boldsymbol{x}), v_{i,l}\rangle| \le \tilde{O}(C^{-1.5})$*

Disclaimer Technically, Allen-Zhu & Li (2020) only show Claims 1 and 2 for $t \leq T_1$ and one would need to use similar proof techniques (such as their inductive hypothesis F.1) to show the case for $T_1 \leq t \leq T_2$, which we skip for conciseness.

We now study the growth of the student's $\Phi_{c,l}$, for those (c,l) which have been learnt by the teacher but not the student:

Lemma 2 (Correlation Growth for attributes learnt by teacher)**.** *For every $c \in [C], l \in [2]$, $T_2 \geq t \geq T_1$, such that $(c, 3-l) \notin \mathcal{M}_{\mathcal{T}}$, suppose $\Phi_{c,l}^t \leq \frac{1}{2m}$, then we have:*

$$\Phi_{c,l}^{t+1} \geq \Phi_{c,l}^t + \tilde{\Omega}(\frac{\eta s^2}{C^2}) \cdot {\Phi_{c,l}^t}^4 \cdot \widetilde{\text{ReLU}}'(\Phi_{c,l}^t)$$

Proof. For any $c \in [C], r \in [m], l \in [2]$, we have from Claim 2'

$$\langle \Delta \boldsymbol{\theta}_{c,r}^t, v_{c,l} \rangle = \eta \mathbb{E}_{\boldsymbol{x}_1, \boldsymbol{x}_2 \sim \hat{\mathcal{D}}^2} \left[\left(\tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \tilde{k}_c(\boldsymbol{x}_1, \boldsymbol{x}_2) \right)^+ \left[\sum_{j \in \{1,2\}} \Psi_{c,r}(\boldsymbol{x}_j) \left(V_{c,r,l}(\boldsymbol{x}_{3-j}) - \tilde{O}(\sigma_p P) \right) \right] \right]$$

Note that as $\mu \leq \frac{1}{\text{poly}(C)}$, we can suppose that both $\boldsymbol{x}_1, \boldsymbol{x}_2$ are multi-view data. Using Claim 2, we have that

$$\langle \Delta \boldsymbol{\theta}_{c,r}^t, v_{c,l} \rangle \geq \eta \mathbb{E}_{\boldsymbol{x}_1, \boldsymbol{x}_2 \sim \hat{\mathcal{D}}^2} \left[\left(\tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \tilde{k}_c(\boldsymbol{x}_1, \boldsymbol{x}_2) \right)^+ \left[\sum_{j \in \{1,2\}} \Psi_{c,r}(\boldsymbol{x}_j) \left(V_{c,r,l}(\boldsymbol{x}_{3-j}) - \tilde{O}(\sigma_p P) \right) \right] \right]$$

Let $r = \text{argmax}_{r' \in [m]} \{ \langle \boldsymbol{\theta}_{c,r'}^t, v_{c,l} \rangle \}$, such that definitely $\langle \boldsymbol{\theta}_{c,r}^t, v_{c,l} \rangle \geq \tilde{\Omega}(\Phi_{c,l}^t)$ because $m = \text{polylog}(C)$.

But if $\boldsymbol{x}$ is multi-view and $v_{c,l} \in \mathcal{V}(\boldsymbol{x})$ such that $\sum_{p \in \mathcal{P}_{v_{c,l}}} z_p^4 = \Theta(1)$, and also by **Fact A.a** we have that:

$$V_{c,r,l}(\boldsymbol{x}) \geq \Omega(1) \cdot \widetilde{\text{ReLU}}' \left(\langle \boldsymbol{\theta}_{c,r}^t, v_{c,l} \rangle \right) \geq \tilde{\Omega} \left(\widetilde{\text{ReLU}}'(\Phi_{c,l}^t) \right)$$
$$\Psi_{c,r}(\boldsymbol{x}) \geq \sum_{p \in \mathcal{P}_{v_{c,l}}} \widetilde{\text{ReLU}}(\langle \boldsymbol{\theta}_{c,r}^t, v_{c,l} \rangle z_p - \tilde{o}(\sigma_0)) \geq \tilde{\Omega}({\Phi_{c,l}^t}^4)$$

Now, we have assumed that $(c, 3-l) \notin \mathcal{M}_{\mathcal{T}}$, such that the teacher model has learnt attribute (c,l) and satisfies $\tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}') = 1$ when $v_{c,l} \in \mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}')$ (for large enough polylogarithmic m).

Also, it is simple to see that when $v_{c,l} \in \mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}')$ & $v_{c,3-l} \notin \mathcal{V}(\boldsymbol{x}) \cap \mathcal{V}(\boldsymbol{x}')$, for large enough m, that $\Phi_{c,l}^t \leq \frac{1}{2m}$ implies that $k_c(\boldsymbol{x}, \boldsymbol{x}') \leq \frac{1}{m^2}$ by Lemma 1, and so $\tilde{k}_c(\boldsymbol{x}, \boldsymbol{x}') = 0$, i.e. the student has not (yet) learnt $v_{c,l}$.

So we see there are two more conditions that must be satisfied in order for $\left(\tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \tilde{k}_c(\boldsymbol{x}_1, \boldsymbol{x}_2) \right)^+ = 1 > 0$:

1. $v_{c,l} \in \mathcal{V}(\boldsymbol{x}_1) \cap \mathcal{V}(\boldsymbol{x}_2)$ so that $\tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = 1$
2. $v_{c,3-l} \notin \mathcal{V}(\boldsymbol{x}_1) \cap \mathcal{V}(\boldsymbol{x}_2)$ so that $\tilde{k}_c(\boldsymbol{x}_1, \boldsymbol{x}_2) = 0$.

Going back to App. B.1.1, we know that these conditions occur with probability $\frac{s^2}{C^2}(1 - o(1))$ for independently sampled $\boldsymbol{x}_1, \boldsymbol{x}_2 \sim \hat{\mathcal{D}}$. Finally, putting everything together we have that:

$$\langle \boldsymbol{\theta}_{c,r}^{t+1}, v_{c,l} \rangle^+ - \langle \boldsymbol{\theta}_{c,r}^t, v_{c,l} \rangle^+ \geq \tilde{\Omega}(\frac{\eta s^2}{C^2}) {\Phi_{c,l}^t}^4 \cdot \widetilde{\text{ReLU}}'(\Phi_{c,l}^t)$$

summing over $r' \in [m]$ and noting $\langle \Delta \boldsymbol{\theta}_{c,r'}^t, v_{c,l} \rangle \geq 0 \ \ \forall r'$, up to small error (as $\sigma_p P = \frac{1}{\text{poly}(C)}$ for a large polynomial), gives us our result.

□

Lemma 2 immediately gives us the following corollaries, because $\Phi_{c,l}^{T_1} \geq \tilde{\Omega}(\sigma_0)$ from Allen-Zhu & Li (2020) Induction Hypothesis F.1.g and $\widetilde{\text{ReLU}}'$ is increasing.

Corollary 1. *Define iteration threshold $T_2 = \tilde{\Theta}(\frac{C^2}{\eta s^2 \sigma_0^7}) = \tilde{\Theta}(\frac{C^{5.1}}{\eta})$, then for every (c, l) such that $(c, 3-l) \notin \mathcal{M}_{\mathcal{T}}$ we have:*

$$\Phi_{c,l}^{T_1+T_2} \geq \frac{1}{4m}$$

But likewise, we can also bound the growth of $\Phi_{c,l}$

Lemma 3. *If $(c, 3-l) \in \mathcal{M}_{\mathcal{S}} \backslash \mathcal{M}_{\mathcal{T}}$, once $\Phi_{c,l}^t \geq \sqrt{\frac{\text{loglog}(C)}{m}}$, it no longer gets updated (for large C):*

Proof. Recall Definitions 3 and 4 that $\Phi_{c,l}^* = \sum_{r \in [m]} [\langle \boldsymbol{\theta}_{c,r}^t, v_{c,l} \rangle]^+$ and $\Upsilon_{c,l}^t = \sum_{r \in [m]} [\langle \boldsymbol{\theta}_{c,r}^t, v_{c,l} \rangle]^{+^2}$

Hence by Cauchy-Schwarz we have:

$$\Phi_{c,l}^{t\ 2} \leq m \Upsilon_{c,l}^t \implies \Upsilon_{c,l}^t \geq \frac{\text{loglog}(C)}{m^2} \geq \frac{2}{0.4^2 m^2}$$

for large enough C. Thus, if x_1, x_2 are both multi-view and $v_{c,l} \in \mathcal{V}(\boldsymbol{x}_1) \cap \mathcal{V}(\boldsymbol{x}_2)$, by Lemma 1 we must have

$$k_c(\boldsymbol{x}_1, \boldsymbol{x}_2) \geq \frac{1}{m^2}$$

It's also not difficult to check that any other possible setting for $\boldsymbol{x}_1, \boldsymbol{x}_2,$ and $\mathcal{V}(\boldsymbol{x}_1) \cap \mathcal{V}(\boldsymbol{x}_2)$ will lead to $\left(\tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \tilde{k}_c(\boldsymbol{x}_1, \boldsymbol{x}_2)\right)^+ = 0$, and hence

$$\left(\tilde{k}_c^{\mathcal{T}}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \tilde{k}_c(\boldsymbol{x}_1, \boldsymbol{x}_2)\right)^+ = 0 \qquad \forall \boldsymbol{x}_1, \boldsymbol{x}_2$$

□

C.5 Wrapping up proof of Theorem 2

Proof. We are now ready to wrap up our proof. Recall from the proof of Theorem 1 in Allen-Zhu & Li (2020), that after the initial phase of T_1 steps of student training on its own:

$$\Phi_c^{T_1} \geq \Omega(\log(C)) \ \ \forall c \in [C],$$

and more specifically:

- If $(c, 3-l) \notin \mathcal{M}_{\mathcal{S}}$, then

  $$\Phi_{c,l}^{T_1} \geq \Omega(\log(C))$$

  This gives us perfect test accuracy on the multi-view data, and 50% accuracy on the single-view data, so 0.5μ test accuracy overall without distillation, as per Theorem 1.

- Moreover, we have that if $(c, 3-l) \notin \mathcal{M}_{\mathcal{T}}$ and $(c, 3-l) \in \mathcal{M}_{\mathcal{S}}$, then by Corollary 1 and Lemma 3:

$$\sqrt{\frac{\text{loglog}(C)}{m}} \geq \Phi_{c,l}^{T_1+T_2} \geq \frac{1}{4m}$$
$$\implies \sqrt{\text{loglog}(C)} \geq \Phi_{c,l}^{T_1+T_2} \geq \frac{1}{4m}$$

  We see that this change in $\Phi_{c,l}^{T_1+T_2}$ after FKD training is much smaller than $\Omega(\log(C))$ and so the student after FKD training still has perfect multi-view accuracy, as well as correct predictions on any single-view data that possess the attributes learnt in the initial phase of training.

  On the other hand, for single-view data, we know that if we have data point $\boldsymbol{x}, y$ and attribute $v_{c,l} \in \mathcal{V}(\boldsymbol{x})$, such that $c \neq y$, then $\sum_{p \in \mathcal{P}_{v_{c,l}}(\boldsymbol{x})} z_p = \Gamma = O(\frac{1}{\text{polylog}(C)})$, as defined in App. B.1.1.

  Hence for small enough Γ ($\ll \frac{1}{m}$) we have that if $(c, 3-l) \notin \mathcal{M}_{\mathcal{T}}$ and the single view data $\boldsymbol{x}$ is of class c with $\hat{l}(\boldsymbol{x}) = l$ then we have correct prediction, as per Claim 1.

Combining these means that we have correct prediction for any single-view data $\boldsymbol{x}$ of class c, and $\hat{l}(\boldsymbol{x}) = l$ such that $(c, 3-l) \notin \mathcal{M}_{\mathcal{T}} \cap \mathcal{M}_s$.

By the independence of these sets we have that $|\mathcal{M}_{\mathcal{T}} \cap \mathcal{M}_s| = (2^{-E})k(1-o(1))$ this means we have test error less than $(2^{-E-1} + \epsilon)\mu$ for any $\epsilon > 0$, for large enough C as required.

□

D DIFFERENCES BETWEEN FKD & OTHER FEATURE KERNEL BASED KD METHODS

In this section, we highlight how our FKD approach overcomes some of the shortcomings of previous feature kernel based KD methods which only use pairwise evaluations of the feature kernel: SP (Tung & Mori, 2019) and RKD (Park et al., 2019). One advantage of FKD relative to these previous works is that we have shown FKD is amenable to ensemble distillation. Moreover, it goes without saying that Feature Regularisation, which arises naturally thanks to our feature kernel learning perspective in Section 4, is already a significant departure that improves FKD relative to SP & RKD. However, even without FR we observe in Section 5 that FKD outperforms both RKD & SP across different datasets and architectures, which warrants explanation.

D.1 IMPORTANCE OF ZERO DIAGONAL DIFFERENCES: SP (TUNG & MORI, 2019)

First, we consider diagonal kernel differences, $k^{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{x}) - k^{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x})$ for fixed $\boldsymbol{x}$, and motivate using zero diagonal differences, which is not present in SP (Tung & Mori, 2019) but is in FKD thanks to our use of the correlation kernel. Fig. 6 displays this comparison between FKD & SP graphically.

Intuition: Downside of non-zero diagonal differences

The key intuition, which we detail below using our theoretical setup, is that non-zero diagonal differences $k(\boldsymbol{x}, \boldsymbol{x}) - k^{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}) \neq 0$ encourage the student to learn noise in input $\boldsymbol{x}$, compared to when we have zero diagonal differences $k(\boldsymbol{x}, \boldsymbol{x}) - k^{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}) = 0$. In the latter case, we only have non-zero differences for $k(\boldsymbol{x}, \boldsymbol{x}') - k^{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}')$ where $\boldsymbol{x} \neq \boldsymbol{x}'$.

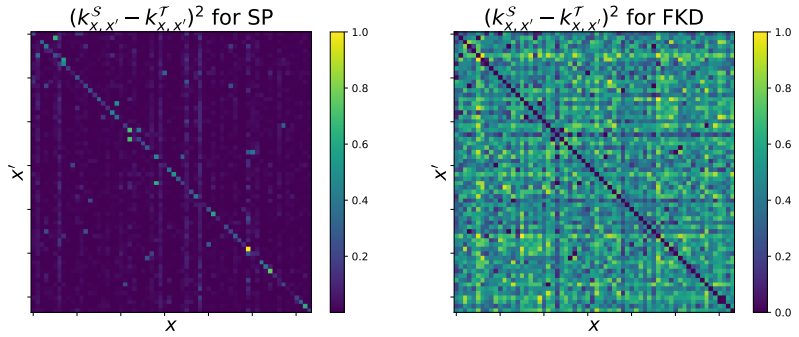


Figure 6: Comparison of (normalised) squared differences in $k_{\boldsymbol{x},\boldsymbol{x}'} = k(\boldsymbol{x}, \boldsymbol{x}')$ between student $\mathcal{S}$ & teacher $\mathcal{T}$, across a minibatch of size 64 of CIFAR-100 training data, for SP (left) and FKD (right). We see that whereas FKD has zero diagonal differences, SP is largely dominated by non-zero diagonal differences. Note there is a slight abuse of notation here, in that we plot squared differences in *normalised* kernels, so that FKD uses the correlation kernel and SP uses row-normalisation (Tung & Mori, 2019).

Diagonal updates Consider our parameter update Eq. (25) when $\boldsymbol{x}_1 = \boldsymbol{x}_2 = \boldsymbol{x}$:
If we didn't have zero diagonal differences and instead $\tilde{k}_c(\boldsymbol{x}, \boldsymbol{x}) - \tilde{k}^{\mathcal{T}}_c(\boldsymbol{x}, \boldsymbol{x}) = -1$, then:

$$\Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}, \boldsymbol{x}) = 2\eta\Psi_{c,r}(\boldsymbol{x}) \sum_{p=1}^{P} \widetilde{\text{ReLU}}'(\langle\boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p\rangle)\boldsymbol{x}_p,$$

Now suppose $v = v_{c,1} \in \mathcal{V}(\boldsymbol{x})$. For each $p \in \mathcal{P}_v(\boldsymbol{x})$, recall (from App. B.1.1) that:

$$\boldsymbol{x}_p = z_p v + \xi_p,$$

where we assume zero feature noise for simplicity.
We then see that (c.f. Claim C.13 of Allen-Zhu & Li (2020)):

$$\langle\Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}, \boldsymbol{x}), \xi_p\rangle = 2\tilde{\Theta}(\eta)\Psi_{c,r}(\boldsymbol{x})\widetilde{\text{ReLU}}'(\langle\boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p\rangle) + O(\frac{1}{\sqrt{d}}),$$

as $\langle v, \xi_p\rangle = O(\frac{1}{\sqrt{d}})$ with high probability.
But at the same time (by e.g. Claim F.6 of Allen-Zhu & Li (2020)):

$$\langle\Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}, \boldsymbol{x}), v\rangle = 2\eta\Psi_{c,r}(\boldsymbol{x})V_{c,r,1}(\boldsymbol{x})(1 \pm o(1))$$

and note that that $V_{c,r,1}(\boldsymbol{x}) \leq \Theta(1)$ by definition. Moreover, in order for the student network to learn attribute $v_{c,1}$, then eventually it must satisfy $\max_{r'}\langle\boldsymbol{\theta}_{c,r'}, v\rangle = \tilde{O}(1)$ from Allen-Zhu & Li (2020). Thus, for ϱ small enough, if $r = \text{argmax}_{r'}\langle\boldsymbol{\theta}_{c,r'}, v\rangle$, we have $\widetilde{\text{ReLU}}'(\langle\boldsymbol{\theta}_{c,r}, \boldsymbol{x}_p\rangle) = 1$.

Non-diagonal updates On the other hand, if $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ and we have $v \in \mathcal{V}(\boldsymbol{x}_1) \cap \mathcal{V}(x_2)$:

$$\Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \eta \sum_{j=1,2} \Psi_{c,r}(\boldsymbol{x}_j) \sum_{p=1}^{P} \widetilde{\text{ReLU}}'(\langle\boldsymbol{\theta}_{c,r}, \boldsymbol{x}_{3-j,p}\rangle)\boldsymbol{x}_{3-j,p},$$

and so if $\boldsymbol{x}_1 = \boldsymbol{x}$ and ξ_p denotes the random noise in $\boldsymbol{x}_{1,p}$, then:

$$\langle\Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}_1, \boldsymbol{x}_2), \xi_p\rangle = \tilde{\Theta}(\eta)\Psi_{c,r}(\boldsymbol{x}_2)\widetilde{\text{ReLU}}'(\langle\boldsymbol{\theta}_{c,r}, \boldsymbol{x}_{1,p}\rangle) + O(\frac{1}{\sqrt{d}}),$$

as $\langle \boldsymbol{x}_{2,p}, \xi_p \rangle = O(\frac{1}{\sqrt{d}})$. But this time we have

$$\langle \Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}_1, \boldsymbol{x}_2), v \rangle = \eta\big(\Psi_{c,r}(\boldsymbol{x}_1)V_{c,r,1}(\boldsymbol{x}_2) + \Psi_{c,r}(\boldsymbol{x}_2)V_{c,r,1}(\boldsymbol{x}_1)\big)(1 \pm o(1)),$$

We see that whereas in diagonal updates $\Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}, \boldsymbol{x})$ the increments for $\langle \Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}, \boldsymbol{x}), \xi_p \rangle$ and $\langle \Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}, \boldsymbol{x}), v \rangle$ are 1:1, for non-diagonal updates $\Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ they are 1:2 respectively.

Thus, the parameter updates for $\boldsymbol{\theta}_{c,r}$ with non-zero diagonal differences in feature kernels are more likely to learn noise, ξ_p, compared to our zero diagonal updates which rely only on non-diagonal $\Delta\boldsymbol{\theta}_{c,r}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ for $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$. This is why we zero out diagonal differences for FKD, using the feature correlation matrix in practice.

D.2 Problem of homogeneous NNs in RKD (Park et al., 2019)

The *distance-wise* version of RKD (Park et al., 2019) is as follows: for $\boldsymbol{x}, \boldsymbol{x}'$, we calculate $\psi_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}') = \|h_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{T}}) - h_{\mathcal{T}}(\boldsymbol{x}', \boldsymbol{\theta}_{\mathcal{T}})\|_2$, where recall $h_{\mathcal{T}}$ is the last-layer teacher feature extractor. Likewise, we also calculate $\psi_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{x}') = \|h_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{S}}) - h_{\mathcal{S}}(\boldsymbol{x}', \boldsymbol{\theta}_{\mathcal{S}})\|_2$. The RKD loss adds $\lambda_{\text{KD}}\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[(\psi_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}') - \psi_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{x}'))^2]$ to the student's training loss.[9] While RKD (Park et al., 2019) does ensure zero diagonal differences, i.e. that $\psi_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{x}) - \psi_{\mathcal{S}}(\boldsymbol{x}, \boldsymbol{x}) = 0$, it suffers from a related issue, due to the homogeneity of NNs that use ReLU nonlinearity, which is ubiquitous in image classification tasks.

Suppose we take $\boldsymbol{x}$ and define $\boldsymbol{x}' = M\boldsymbol{x}$ for some $M > 0$. For example, think of taking a cat image and multiplying all the pixel values by M. For ReLU (C)NNs without bias parameters, we have that $h(\boldsymbol{x}, \boldsymbol{\theta})$ is 1-homogeneous: $h(\boldsymbol{x}', \boldsymbol{\theta}) = Mh(\boldsymbol{x}, \boldsymbol{\theta})$. This means that it is likely (depending on the norms of the features $h_{\mathcal{S}}$ and $h_{\mathcal{T}}$) that we will have $\psi_{\mathcal{T}}(\boldsymbol{x}, M\boldsymbol{x}) - \psi_{\mathcal{S}}(\boldsymbol{x}, M\boldsymbol{x}) \neq 0$. But a cat image multiplied by some scalar M is still a cat image, hence RKD runs into the same problems as in App. D.1 of learning noise in $\boldsymbol{x}$.

On the other hand for FKD: correlation kernel $\rho_{\mathcal{S}}(\boldsymbol{x}, M\boldsymbol{x}){=}\rho_{\mathcal{T}}(\boldsymbol{x}, M\boldsymbol{x}){=}1$, hence $\rho_{\mathcal{S}}(\boldsymbol{x}, M\boldsymbol{x}){-}\rho_{\mathcal{T}}(\boldsymbol{x}, M\boldsymbol{x}){=}0 \;\; \forall \boldsymbol{x} \in \mathbb{R}^d, M > 0$.

E PyTorch-style pseudocode for FKD

In Alg. 2, we provide PyTorch-style Paszke et al. (2019b) pseudocode for the distillation and feature regularisation losses in FKD. We note that FKD only requires pairwise computations of feature (correlations) kernels. This alleviates the need for matrix multiplication/inversion operations with batch-by-batch size matrices, which is beneficial for scalability.

F Experimental details and further results

F.1 Fig. 2: predictive disagreement across independent initialisations vs retrained last layer

All models are ResNet20v1 trained with standard hyperparameters:

- 160 epochs training time with batch size 128 and learning rate 0.1 which is decayed by a factor of 10 after epochs 80 and 120.
- SGD optimiser with momentum 0.9 and weight decay of 0.0001.

[9] We do not consider the angle-wise RKD loss here, but there will be similar issues due to homogeneity.

Algorithm 2 PyTorch-style pseudocode for Feature Kernel Distillation (FKD).

```
# B:     Batch size.
# L_FKD: FKD regularisation strength.
# L_FR:  Feature regularisation strength.
# D_s:   Student feature dimension.
# D_t:   Teacher feature dimension.

# f_s:   Student features B x D_s
# f_t:   Teacher features B x D_t

# mm: matrix-matrix multiplication

# Compute student feature correlation kernel matrix s_c
s_k = mm(f_s, f_s.T)                                    # B x B
s_k_diag_inv_sqrt = torch.diag(s_k).pow(-1/2)
s_k_diag_inv_sqrt = s_k_diag_inv_sqrt.reshape(-1, 1)    # B x 1
s_c = s_k_diag_inv_sqrt * s_k * s_k_diag_inv_sqrt.T     # B x B

# Compute teacher feature correlation kernel matrix t_c
with torch.no_grad():
    t_k = mm(f_t, f_t.T)                                # B x B
    t_k_diag_inv_sqrt = torch.diag(t_k).pow(-1/2)
    t_k_diag_inv_sqrt = t_k_diag_inv_sqrt.reshape(-1, 1) # B x 1
    t_c = t_k_diag_inv_sqrt * t_k * t_k_diag_inv_sqrt.T  # B x B

distil_loss = ((t_c - s_c).pow(2)).mean()
feat_reg_loss = (f_s.pow(2)).mean()

# FKD loss to be added to supervised loss
loss_fkd = L_FKD * distil_loss + L_FR * feat_reg_loss
```

- CIFAR10 data is normalised in each channel such that the training data is zero mean and unit standard deviation. Random crops and horizontal flips used as data augmentation.
- All models are initialised with Kaiming initialisation He et al. (2015).

Out of 10000 test points, the predictive disagreements between a 'reference' model and either: independent initialisations (top row) or retrained last layers (bottom row) are depicted in Table 4. We see two clear trends. First, the retrained last layer has much fewer disagreements with the reference model than an independent initialisation model, highlighting the importance of the feature kernel. Secondly, the vast majority of disagreements between independent initialisations are where one of the models is correct. This reinforces our intuition/theoretical analysis that ensembling NN works because different initialisations bias the models to capture different useful features, and hence ensemble distillation (via feature kernels) can improve student performance.

Table 4: Breakdown of predictive disagreements between reference and alternate models over 10000 CIFAR10 test points, in terms of which model (if any) was correct. Mean $\pm$ standard deviations over 3 independent initialisations for top row, and over 3 independent reference models for bottom row. All models achieved between 8.0%-8.5% test error.

Alternate model	Reference correct	Alternate correct	Neither correct	Total disagreement
Independent Init	350 ± 14.6	383 ± 15.9	124 ± 3.7	857 ± 29.8
Retrained LL	30 ± 2.9	35 ± 1.7	15 ± 5.4	80 ± 4.1

F.2 Fig. 3: Additional feature kernel histograms

In Fig. 7, we provide additional plots to Fig. 3 that depict the difference in distribution (over $\boldsymbol{x}$) of feature kernel values $k(\boldsymbol{x}, \boldsymbol{x})$, for FKD with and without Feature Regularisation (FR).

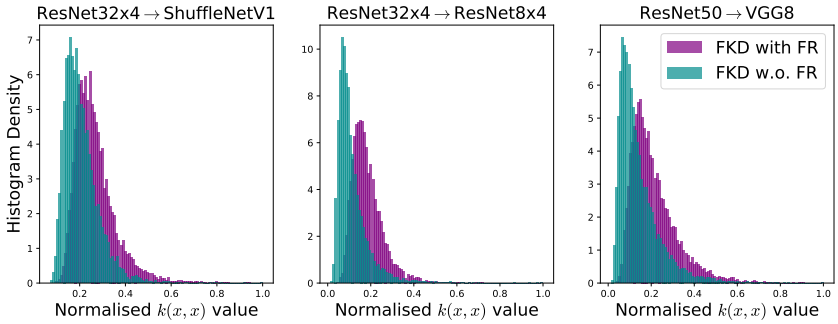


Figure 7: Comparison of normalised $k(\boldsymbol{x}, \boldsymbol{x})$ values between FKD with & without Feature Regularisation (FR), across different Teacher→Student architectures, on CIFAR-100 test set. We see, like in Fig. 3 that FR encourages a more even distribution of $k(\boldsymbol{x}, \boldsymbol{x})$ across $\boldsymbol{x}$, for all architectures.

Negative hypothesis. We originally hypothesised that FR could benefit FKD, in addition to balancing the distribution of $k(\boldsymbol{x}, \boldsymbol{x})$, as it could reduce the sparsity in the NN last-layer representation activations, which is consistent with the proofs of Theorems 1 and 2. Indeed, the NN predictions are dominated by a select few neurons who 'have won the lottery' (Allen-Zhu & Li, 2020) on account of being most correlated with one of the attributes $\{v_{c;l}\}_{l\in[2],c\in[C]}$ at random initialisation. In our analysis, as few as $O(\log^5(m)$ out of m neurons could be inactive. From our feature kernel learning perspective, this seems like a highly undesirable phenomenon, because if $k(\boldsymbol{x}, \boldsymbol{x}') = \langle h(\boldsymbol{x}; \boldsymbol{\theta}), h(\boldsymbol{x}'; \boldsymbol{\theta})\rangle = \sum_{r=1}^{Cm} h_r(\boldsymbol{x}, \boldsymbol{\theta}) h_r(\boldsymbol{x}', \boldsymbol{\theta})$ only has useful contributions from a dominant minority of $r \in [Cm]$, then we are not utilising the full capacity of the model. FR seemed appropriate to reduce this sparsity as ℓ_2 regularisation is known to promote non-sparse solutions (Van Den Doel et al., 2013). However, we experimentally found the opposite to our hypothesis: that FR trained FKD student had more inactive neurons relative to FKD students trained without FR. This again highlights that, while (we believe) our results in this work provide compelling evidence to highlight the validity of feature kernel based distillation, there are still gaps between our theory and practice, and further questions to be answered in future work.

F.3 Fig. 4: Ensemble distillation

All individual VGG8 networks that made up the teacher ensemble were trained using the default training regime from Tian et al. (2020), with independent parameter initialisations. Indeed, all of our experiments in Section 5 used Tian et al. (2020)'s excellent open-source PyTorch codebase (Paszke et al., 2019a).[10]

For student networks, we used the training regime for vanilla KD from Tian et al. (2020) for all ensemble sizes. For FKD, we used the hyperparameters from our ResNet50→VGG8 experiment in Table 2 for all ensemble sizes.

F.4 Table 1: Dataset Transfer

The VGG13 teacher checkpoint is provided by Tian et al. (2020). For both CIFAR-10 and STL-10, all student networks are trained for 160 epochs with batch size 64 using SGD with momentum, with learning rate decays at epochs 80, 120, 150. The student trained without KD used default hyperparameters from Tian et al. (2020), which are indeed strong hyperparameters for standard training. For FKD, RKD (Park et al., 2019) and SP (Tung & Mori, 2019), we tuned the learning

[10] `https://github.com/HobbitLong/RepDistiller`

rate, learning rate decay, and KD regularisation strength λ_{KD} on a labeled validation set of size 5000 for CIFAR-10 and 1000 for STL-10, before retraining using best hyperparameters on the full training(+unlabeled) dataset. We also tuned the FR regularisation strength, λ_{FR} for FKD when FR was used. All RKD and SP hyperparameters were tuned in a large window around their default values from Tian et al. (2020), which were all author recommended. For FKD, we allowed λ_{KD} to range in [1,1000], and λ_{FR} to range in [0,20]. All hyperparameters sweeps were conducted using Bayes search.

For STL-10, we used a batch size of 512 for all KD methods' regularisation terms, compared to 64 for the standard cross-entropy loss. This was due to the fact that STL-10 has only 5K labeled datapoints, and we wanted to ensure that the student used as much of the unlabeled data as possible for each feature-kernel based KD method's additional regularisation term during 160 epochs of training. 512 batch size was the maximum power of 2 before we ran into memory issues on a 11GB VRAM GPU, which occured for the RKD method.

Both CIFAR-10 and STL-10 data are normalised in each channel such that the training data is zero mean and unit standard deviation. Random crops and horizontal flips used as data augmentation. STL-10 images are downsized from 96x96 to 32x32 resolution.

F.5 Table 2: CIFAR-100 and ImageNet comparison

CIFAR-100. All networks were trained for 240 epochs with batch size 64, with learning rate decay at epochs 150, 180, 210 using SGD with momentum. All teacher networks use the exact same checkpoints as provided by Tian et al. (2020). Learning rate, learning rate decay, λ_{KD}, and λ_{FR} (when used) were tuned as in App. F.4 on a validation of size 5000. All other hyperparameters were set to the default values used by Tian et al. (2020). The CIFAR-100 data is normalised in each channel such that the training data is zero mean and unit standard deviation, with random crops and horizontal flips used for data augmentation. All results provided denote the test set accuracy at the end of the 240 epochs of training.

ImageNet. The ImageNet dataset (ILSVRC-2012) consists of about 1.3 million training images and 50,000 validation images from 1,000 classes. Each training image is extracted as a randomly sampled 224x224 crop or its horizontal flip without any padding operation. All teacher networks use the exact same checkpoints as provided by Chen et al. (2021). The initial learning rate is 0.1 and divided by 10 at 30 and 60 of the total 90 training epochs. We set the mini-batch size to 256 and the weight decay to 10^{-4}. λ_{KD}, and λ_{FR} (when used) were tuned as in App. F.4 on a validation of size 5000 except using Bayes search. All results are reported in a single trial. All other hyperparameters were set to the default values used by Chen et al. (2021). All results provided denote the Top-1 test accuracy (%). Accuracy of baselines were reported in Tian et al. (2020).

F.6 Sensitivity to λ_{KD}

In Fig. 8, we plot the sensitivity of FKD to the strength of the distillation regularisation λ_{KD} in Eq. (2) for the VGG13→VGG8 experiment on CIFAR-100. We see that a well tuned $\lambda_{\text{KD}}(\approx 300$ here) is important for best student generalisation. Feature regularisation $\lambda_{\text{FR}} = 20$ in Fig. 8.

F.7 Table 5: Analyses in Neural Machine Translation

In this section, We performed analyses for a neural machine translation (NMT) task proposed by Tan et al. (2019). In the analyses, we could only obtain data for En-De (from English to German) translation since links to the datasets for other languages are broken. Therefore, we employed a self-distillation method on a pre-trained English model for En-De translation as follows:

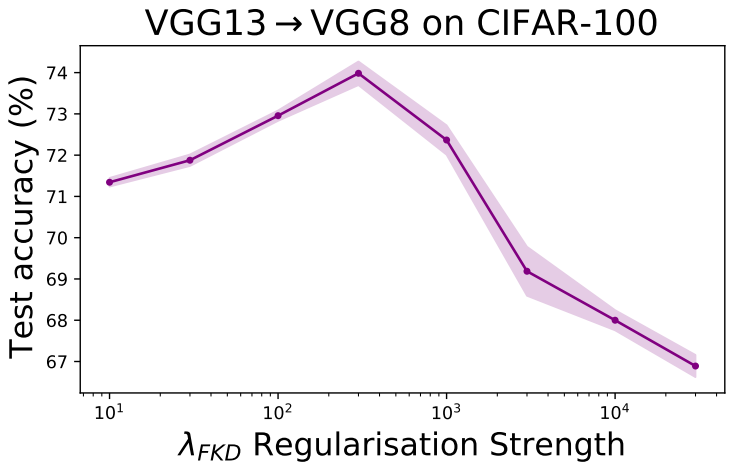


Figure 8: Comparison of normalised $k(\boldsymbol{x}, \boldsymbol{x})$ values between FKD with & without Feature Regularisation (FR), across different Teacher→Student architectures, on CIFAR-100 test set. We see, like in Fig. 3 that FR encourages a more even distribution of $k(\boldsymbol{x}, \boldsymbol{x})$ across $\boldsymbol{x}$, for all architectures.

- We train a single teacher transformer model on the IWSLT dataset for English (Tan et al., 2019).
- We perform self-distillation on the teacher model for En-De translation (Tan et al., 2019).
- We did not search for optimal hyperparameters, and used default parameters of the code provided by the authors of Tan et al. (2019). The results are given in Table 5.

Table 5: BLEU of the teacher model of (Tan et al., 2019) (Teacher), self-distillation of (Tan et al., 2019) (SD), SD with KD of (Hinton et al., 2015), SD with FKD, and SD with FKD loss obtained by replacing distillation loss (2) of (Tan et al., 2019) with FKD, in En - De neural machine translation tasks.

Teacher (Tan et al., 2019)	**SD (Tan et al., 2019)**	**SD with KD**	**SD with FKD**	**SD with FKD loss**
27.32	27.49	27.51	**27.64**	**27.79**

We first note that, we adapted vanilla KD and our FKD for sequential data in the KD loss (2) of (Tan et al., 2019) in this task. More precisely, we first computed vanilla KD and FKD on token probabilities, and added these loss functions to the KD loss (eq 2 of (Tan et al., 2019)) in KD and FKD. In the results, aggregating vanilla KD with the KD loss (eq 2 of (Tan et al., 2019)) improved accuracy from 27.49 to 27.51. However, FKD further boosted BLEU to 27.64. We then replaced KD loss (Eq. 2 of (Tan et al., 2019)) with FKD for training. Remarkably, FKD further boosted the BLEU to 27.79. These results suggest that the proposed FKD can be applied in NMT tasks, successfully. We hope that these results will motivate researchers to employ FKD in various different NLP tasks including but not limited to multilingual NMT, named entity recognition and question answering.

F.8 Table 6: Analyses in Automatic Speech Recognition

In this section, we used a CRDNN model (VGG + LSTM,GRU,LiGRU+ DNN) on the TIMIT dataset. In this experiment, we used a distillation approach proposed by Gao et al. (2020) for ASR tasks as follows:

- We train a single teacher model on the TIMIT dataset Ravanelli et al. (2021).

- We perform self-distillation on the teacher Gao et al. (2020).
- We did not search for optimal hyperparameters, and used default parameters of the SpeechBrain Library.
- In this task, replacing CTC/NLL distillation losses with KD (Hinton et al., 2015) did not converge. Additional investigation with hyperparameter search is needed. We used phoneme error rate (PER) to measure accuracy of models.

Table 6: Phoneme error rate (PER) of methods in automatic speech recognition tasks.

Teacher	**Distilled Teacher (Gao et al., 2020)**	**KD (Hinton et al., 2015)**	**FKD**
13.26	12.80	12.86	**12.59**

The results are given in Table 6. Similar to the NMT task, we adapted vanilla KD and our FKD for sequential data as follows: We first computed vanilla KD and FKD loss functions on token probabilities, and then added to the total loss (eq 7 of Gao et al. (2020)). In the analyses, Vanilla KD (Hinton et al., 2015) increased the PER from 12.80 to 12.86. However, FKD further improved the PER from 12.80 to 12.59. In this task, training models by replacing CTC/NLL distillation losses (eq 4 or 5 of Gao et al. (2020)) with KD (Hinton et al., 2015) and FKD did not converge. In conclusion, these results propound that FKD can be applied for different tasks, i.e., image classification, NMT and ASR, boosting accuracy of baseline distillation methods. We hope that these initial results will motivate researchers in different communities (computer vision, NLP, and ASR) to further expound and apply FKD in additional sub-tasks.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

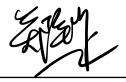
Title of Paper	Feature Kernel Distillation
Publication Status	Published
Publication Details	Bobby He and Mete Ozay. Feature Kernel Distillation. In Proceedings of the Tenth International Conference on Learning Representations, 2022.

Student Confirmation

Student Name:	Bobby He		
Contribution to the Paper	- Lead author with one advisor. - Identified problem tackled in paper. - Derived all theoretical results and proofs, which were checked by advisor. - Performed all experiments apart from ImageNet, NMT and ASR. - Wrote the paper, with helpful feedback from advisor.		
Signature 		Date	8/1/23

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Yee Whye Teh		
Supervisor comments Bobby is lead author and did all the research work in this paper.		
Signature 	Date	8 Jan 2023

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 5

Exploring the Gap between Collapsed & Whitened Features in Self-Supervised Learning

This paper was published as the following

> Bobby He and Mete Ozay. Exploring the Gap between Collapsed & Whitened Features in Self-Supervised Learning. In Proceedings of the 39th International Conference on Machine Learning, 2022.

Exploring the Gap between Collapsed & Whitened Features in Self-Supervised Learning

Bobby He [1 †] **Mete Ozay** [2]

Abstract

Avoiding feature collapse, when a Neural Network (NN) encoder maps all inputs to a constant vector, is a shared implicit desideratum of various methodological advances in self-supervised learning (SSL). To that end, whitened features have been proposed as an explicit objective to ensure uncollapsed features (Zbontar et al., 2021; Ermolov et al., 2021; Hua et al., 2021; Bardes et al., 2022). We identify power law behaviour in eigenvalue decay, parameterised by exponent $\beta \geq 0$, as a spectrum that bridges between the collapsed & whitened feature extremes. We provide theoretical & empirical evidence highlighting the factors in SSL, like projection layers & regularisation strength, that influence eigenvalue decay rate, & demonstrate that the degree of feature whitening affects generalisation, particularly in label scarce regimes. We use our insights to motivate a novel method, Post-hoc Manipulation of the Principal Axes & Trace (PostMan-Pat), which efficiently post-processes a pretrained encoder to enforce eigenvalue decay rate with power law exponent β, & find that PostMan-Pat delivers improved label efficiency and transferability across a range of SSL methods and encoder architectures.

1. Introduction

As label procurement can be expensive relative to the availability of unlabelled data, self-supervised learning (SSL), where a learning algorithm operates without access to labels, has grown both in importance & interest in recent years. Without labels, a general recipe that has produced impressive results is: 1) learning NN features/representations that are invariant to transformations of the same input, whilst 2) avoiding a completely collapsed representation, when all inputs map to a constant feature vector.

A variety of approaches have been proposed to successfully avoid feature collapse, including: contrastive learning (Chen et al., 2020a; He et al., 2020b); clustering (Caron et al., 2018; 2020); non-contrastive learning (Grill et al., 2020; Chen & He, 2021); and kernel dependence maximisation (Li et al., 2021). Of particular relevance to this work are feature decorrelation/whitening SSL methods (Ermolov et al., 2021; Zbontar et al., 2021; Hua et al., 2021; Bardes et al., 2022), which promote whitened/decorrelated features as a sufficient condition to avoid collapse.

Existing theoretical analyses into feature collapse & its mechanisms in SSL have focused on explaining why it does not occur in non-contrastive SSL (Tian et al., 2021; Zhang et al., 2022) or how a related notion of *dimensional collapse* (Hua et al., 2021), where features span a low-dimension subspace of the entire feature space, occurs (Jing et al., 2021). In these works, (dimensional) feature collapse can be interpreted in terms of a binary outcome for each dimension of the encoder NN: collapsed or uncollapsed. Thus, the size of uncollapsed feature dimensions and the importance of their rate of decay have so far been unexplored in SSL.

In this work, we examine the gap between collapsed & whitened features, highlighting its significance by first identifying power law behaviour in eigenvalue decay as a bridge between collapse & whitening in Section 2. We theoretically & empirically study elements of SSL that affect eigenvalue decay rate in Section 3, including projector head depth & regularisation strength, before demonstrating that generalisation performance in SSL is non-monotonic in the degree of feature whitening/collapse in Section 4. We show that the extent of feature whitening has implications for generalisation in low-labelled data regimes in Section 4.1, & use this to motivate our methodological contribution in Section 5: Post-hoc Manipulation of the Principal Axes & Trace (PostMan-Pat or PMP), which takes a pretrained SSL encoder and enforces a power law in its feature eigenspectrum. In Section 6, we show that PMP improves label-efficiency and transferability of pretrained SSL encoders under linear evaluation & often outperforms semi-supervised finetuning. For example, a pretrained Barlow Twins (Zbontar et al., 2021) encoder is improved by over 1% top-1 accuracy (56.2% vs 55.0%) on ImageNet-1K with only 1% of labels.

[†]Researched while interning at Samsung Research UK. [1]University of Oxford, [2]Samsung Research UK. Correspondence to: Bobby He <bobby.he@stats.ox.ac.uk>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

2. Background, Related Work, & Notation

Suppose we have a large unlabelled dataset $\boldsymbol{X}=\{\boldsymbol{x}_n\}_{n=1}^N$, with N samples of dimension d. We denote the empirical distribution over $\boldsymbol{X}$ by $\hat{p}(\boldsymbol{x})$. The SSL setting we consider is to learn a useful feature encoder NN, $h_{\boldsymbol{\theta}}$: $\mathbb{R}^d \to \mathbb{R}^{d_e}$ from $\boldsymbol{X}$, with trainable parameters $\boldsymbol{\theta}$, for downstream tasks. If appropriate, we drop the $\boldsymbol{\theta}$ subscript for clarity.

Self-supervised representations are generally evaluated with a labelled/supervised dataset of samples $\boldsymbol{X}_S \in \mathbb{R}^{S\times d}$ and labels $\boldsymbol{Y}_S \in \mathbb{R}^{S\times C}$. We assume $\boldsymbol{X}_S$ are sampled i.i.d. from the same marginal distribution $p(\boldsymbol{x})$ as our unlabelled $\boldsymbol{X}$, and that there is an additional conditional $q(\cdot|\boldsymbol{x})$ such that $\boldsymbol{y}_s|\boldsymbol{x}_s \sim q(\boldsymbol{y}|\boldsymbol{x}_s)$. Typically the task is C-class classification, & a linear layer $W_C \in \mathbb{R}^{d_e\times C}$ is composed on top of the encoder $h_{\boldsymbol{\theta}}$ to give predictor $f(\boldsymbol{x}) = h_{\boldsymbol{\theta}}(\boldsymbol{x})W_C$.

Evaluation is usually via: 1) non-linear finetuning by training both W_C & $h_{\boldsymbol{\theta}}$ (*finetuning*) or, 2) linear training of W_C only (*linear probe*). Linear probes train the following loss, where l is typically cross-entropy & $\frac{\sigma^2}{S}$ is weight decay:

$$\mathcal{L}(W_C) = \sum_{s=1}^{S} l(f(\boldsymbol{x}_s), \boldsymbol{y}_s) + \sigma^2 \|W_C\|_2^2. \qquad (1)$$

As mentioned, recent approaches (Chen et al., 2020a; Caron et al., 2020; Zbontar et al., 2021; Grill et al., 2020) all adopt the idea of training $\boldsymbol{\theta}$ to be invariant to a distribution of $\mathbb{R}^d \to \mathbb{R}^d$ transformations $\mathcal{T}$ that preserves semantic content (like random crops). In other words, given an image $\boldsymbol{x} \in \mathbb{R}^d$, we have $h_{\boldsymbol{\theta}}(T_1(\boldsymbol{x})) \approx h_{\boldsymbol{\theta}}(T_2(\boldsymbol{x}))$ for $T_1, T_2 \overset{\text{i.i.d.}}{\sim} \mathcal{T}$, where the joint-embeddings $T_1(\boldsymbol{x})$ & $T_2(\boldsymbol{x})$ are known as a *positive pair*. Figure 1 visualises this general approach.

Where these methods differ is in how they avoid the trivial (& useless) solution of *collapsed features*: $h_{\boldsymbol{\theta}}(\boldsymbol{x}') \triangleq \mathbf{c}$, $\forall \boldsymbol{x}' \in \mathbb{R}^d$, for constant $\mathbf{c} \in \mathbb{R}^{d_e}$. We highlight two popular approaches to avoid collapse:

Contrastive SSL methods avoid collapse by simultaneously encouraging representations of different images (*negative pairs*) $\boldsymbol{x} \neq \boldsymbol{x}'$ to be further apart in contrast to positive pairs' representations through the InfoNCE loss (Oord et al., 2018). SimCLR (Chen et al., 2020a;b) demonstrates the benefit of scaling to large batch sizes for contrastive SSL & introduces several techniques that seem to improve SSL performance in practice, such as stronger data augmentation, and the use of trainable (nonlinear) MLP projector heads $g(\cdot) : \mathbb{R}^{d_e} \to \mathbb{R}^{d_p}$. The projection $g(\cdot)$ takes encoder outputs $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ as input, so the InfoNCE loss actually acts on projections $z(\boldsymbol{x}) \triangleq g(h_{\boldsymbol{\theta}}(\boldsymbol{x}))$, not encodings $h_{\boldsymbol{\theta}}(\boldsymbol{x})$. It has been suggested that projectors serve to prevent encoder dimensional collapse (Jing et al., 2021) & ease the encoder's constraints on transformation invariance (Bordes et al., 2021). We see in Section 3 that another effect of projectors could be to whiten representations. MoCo (He et al., 2020b; Chen et al., 2020c) uses a memory bank to ease the large batch size bottleneck of contrastive SSL.

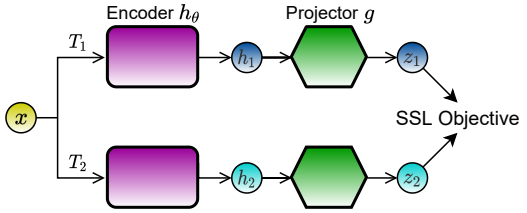


Figure 1. Schematic of joint-embedding approach in SSL.

Feature decorrelation SSL removes the need for negative pairs (hence large batch size) instead by encouraging whitened/decorrelated projections $z(\boldsymbol{x})$ over $\boldsymbol{x} \sim \hat{p}(\boldsymbol{x})$, as a sufficient condition to avoid feature collapse. W-MSE (Ermolov et al., 2021) uses an explicit Cholesky transformation to enforce an exactly whitened representation, by which we mean that the empirical distribution of representations $\{z(\boldsymbol{x}_n)\}_{n=1}^N$ has mean $\mathbf{0}$ & identity covariance $\Sigma \triangleq \frac{1}{N} z(\boldsymbol{X})^T z(\boldsymbol{X}) = I_{d_p \times d_p}$. Hua et al. (2021) considered *dimensional collapse*, where Σ has $m < d_p$ non-zero eigenvalues, as a milder but also undesirable form of collapse, thus motivating their study of feature whitening SSL.

We see that the degree of (projection) feature whitening or collapse can be defined through the eigenvalues of the covariance matrix Σ: identical non-zero eigenvalues give entirely whitened representations, whereas a collapsed representation has all-zero eigenvalues (or a single non-zero eigenvalue if the representation is uncentred). Between these two extremes there is a spectrum of possibilities for how the eigenvalues $\{\lambda_i\}_{i=1}^{d_p}$ of covariance Σ decay, which we characterise as follows using power law behaviour:

Definition 2.1 (β-power law of eigenvalues)**.** Let $\boldsymbol{\phi}(\boldsymbol{X}) \in \mathbb{R}^{N\times d_\phi}$ be a representation of $\boldsymbol{X}$ with feature-wise covariance matrix $\Sigma^\phi \in \mathbb{R}^{d_\phi \times d_\phi}$ and corresponding sorted eigenvalues $\{\lambda_i\}_{i=1}^{d_\phi}$. We say $\boldsymbol{\phi}(\boldsymbol{X})$ follows an *eigenvalue power law with exponent* $\beta \geq 0$ if there exist[1] $\Theta_{d_\phi}(1)$ positive constants $a \leq b$, such that $\forall i \geq 1$: $\frac{a}{i^\beta} \leq \lambda_i \leq \frac{b}{i^\beta}$.

Definition 2.2 (Whitened & collapsed representation)**.** We say a representation $\boldsymbol{\phi}(\boldsymbol{X})$ is *whitened* if it follows an eigenvalue power law with exponent $\beta = 0$, & *collapsed* if it follows an eigenvalue power law with exponent $\beta = \infty$.

Remark 2.3. Def. 2.1 is a potentially softer definition than one where the eigenvalues follow a strict power law (i.e. $a = b > 0$), and is found elsewhere in the literature, e.g. Jin et al. (2021).

In Def. 2.1, $\boldsymbol{\phi}(\boldsymbol{X})$ can be any function of $\boldsymbol{X}$. At various

[1] We presume d_ϕ to be a large but finite feature dimension; in an NN d_ϕ corresponds to the NN width.

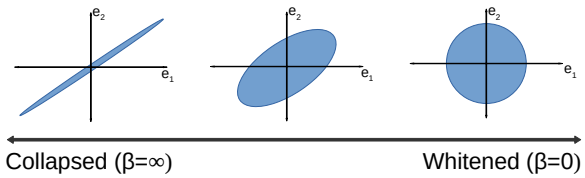


Figure 2. Toy illustration of the gap between collapsed & whitened features as two extremes of a spectrum. The 2D feature covariances above share eigenvectors, but have different eigenvalues.

points, we consider: the identity function; a hidden layer of an encoder h; an encoder h; or an encoder h + projection g. Figure 2 depicts a 2D visualisation of the range between collapse and whitened features, which may be parameterised by power-law exponent β in eigenvalue decay, as in Def. 2.1.

In lieu of explicit whitening, Barlow Twins (Zbontar et al., 2021) introduces a regularised loss $\mathcal{L}_{\text{BT}}$, with strength $\rho{>}0$ providing a soft constraint on feature correlation:

$$\mathcal{L}_{\text{BT}} = \sum_{i=1}^{d_p} \Big[(1 - C_{ii})^2 + \rho \sum_{j \neq i} C_{ij}^2 \Big], \tag{2}$$

where for $\boldsymbol{x}{\sim}\hat{p}(\boldsymbol{x})$ & $\{T_a\}_{a\in[2]} \overset{\text{i.i.d.}}{\sim} \mathcal{T}$, we have correlations:

$$C_{ij} = \frac{\mathbb{E}\big[\bar{z}_i(T_1(\boldsymbol{x}))\bar{z}_j(T_2(\boldsymbol{x}))\big]}{\mathbb{E}\big[\bar{z}_i(T_1(\boldsymbol{x}))^2\big] \cdot \mathbb{E}\big[\bar{z}_j(T_2(\boldsymbol{x}))^2\big]}, \tag{3}$$

with $\bar{z}_i(\boldsymbol{x}) \triangleq z_i(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim\hat{p}}[z_i(\boldsymbol{x})]$ defined to be centred. In practice, the expectations in Eq. (3) are estimated with mini-batches & random transformations sampled from $\mathcal{T}$. Note that, up to input transformations, we have $C_{ij}{=}\frac{\Sigma^z_{i,j}}{\sqrt{\Sigma^z_{i,i}\Sigma^z_{j,j}}}$.

In $\mathcal{L}_{\text{BT}}$, the on-diagonal elements of C encourage $z(\boldsymbol{x})$ to be invariant to transformations of the same image, whereas the off-diagonal contributions encourage the d_p individual projection features to be pairwise uncorrelated across inputs, preventing collapse (of both the encoder and projection).

Key takeaways: β-power law of eigenvalues in SSL.

- Power law behaviour, determined by exponent $\beta > 0$, in feature eigenspectrum decay is one *possible* way to bridge the gap between collapsed & whitened features.

3. Projection Layers Whiten Eigenspectra

Having highlighted the gap between collapsed and whitened features, we now study the factors, such as projector layers and choice of SSL method, that influence where a pretrained encoder lies along this spectrum.

Intuitively, low Barlow Twins training loss results in whitened projections, if individual neurons' variance across inputs satisfies $\Sigma^z_{ii} = \Theta(1)$ for $i \in [d_p]$. This is because if $\mathcal{L}_{\text{BT}}$ is small, then Σ^z is approximately diagonal, and we can read off the eigenvalues $\{\Sigma^z_{ii}\}_i$, as follows:

Proposition 3.1. *Suppose an NN encoder+projection trained via Barlow Twins (Zbontar et al., 2021) achieves (i) training loss $\mathcal{L}_{BT} = \epsilon$, & ii) $\exists a \leq b$ positive constants such that $a \leq \Sigma^z_{ii} \leq b \ , \forall i{\in}[d_p]$. Then, if $\epsilon \leq \frac{a^2\rho}{b^2}$, the projector has a whitened eigenspectrum.*

From Proposition 3.1 (proof in Appendix A), a successfully trained Barlow Twin encoder obtains a whitened projection eigenspectrum. Moreover, the larger the regularisation strength ρ, the more likely the condition $\epsilon \leq \frac{a^2\rho}{b^2}$ is to be satisfied, so larger ρ leads to more whitened projector eigenspectra, which we later empirically confirm in Figure 4.

To justify our assumptions on feature variances, we note that ensuring $\Sigma^z_{ii}{=}\Theta(1), \forall i{\in}[d_p]$, is one motivation for the VICReg (Bardes et al., 2022) extension of the Barlow Twins loss function $\mathcal{L}_{\text{BT}}$. Having said that, we empirically verify Proposition 3.1 in Figure 3 (center left) for a Barlow Twin ResNet-18 on CIFAR-10, where we plot (normalised) projection eigenvalues by size as training progresses.

As seen in Figure 3, at initialisation, the projection eigenspectrum is dominated by one eigenvalue. Through training, the relative size of smaller eigenvalues increases, such that after 100 epochs, we have around 120 dominant eigenvalues within an order of magnitude of the largest eigenvalue. We note some dimensional collapse is still observed for Barlow Twins projectors, as found for BYOL (Grill et al., 2020; Tian et al., 2021) & SimCLR (Chen et al., 2020a; Jing et al., 2021), as the encoder dimension is 512 with projector width of 1024. We leave an exploration of dimensional collapse in feature decorrelation methods like Barlow Twins for future work, & focus here on the decay rate of the dominant eigenvalues, both for Barlow Twins and in general in SSL.

This whitened eigenspectrum property of feature decorrelation SSL projections is somewhat at odds with findings from neuroscience (Stringer et al., 2019), where it has been observed empirically that neuronal population responses in the visual cortex of mice follow an eigenspectrum power law decay with $\beta{=}1$ in Def. 2.1. In all subplots of Figure 3, we plot the line $y = \frac{1}{x}$ for reference, observing that the dominant projection eigenvalues decay much slower than a $\beta{=}1$ power law. However, in Figure 3 (left) we also see that the eigenvalues for the corresponding encoder decay much faster compared to the projector (noting the log-log scale).

To provide theoretical support for this observation that projection layers change the eigenspectra to encourage faster eigenspectrum decay in encoder layers, we consider the setting of a deep linear MLP, with widths d_l at layer l satisfying $d_l > d \ , \forall l \in [L]$ (so that the MLP is wider than input dimension). Under an additional assumption of alignment between the first layer weight matrix $W_1 \in \mathbb{R}^{d_1 \times d}$ and the

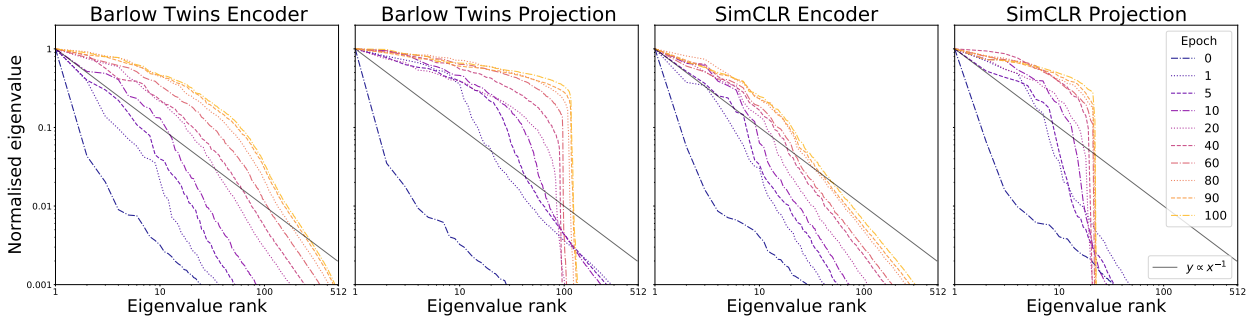


Figure 3. Eigenspectra for features & projections of both Barlow Twins & SimCLR networks. Projector MLPs use ReLU & have depth 2.

input covariance matrix $\Sigma^{\boldsymbol{x}} \in \mathbb{R}^{d\times d}$, we show that changes in the eigenspectrum decay between input and output layers are evenly spaced amongst hidden layers in Corollary 3.4.

Definition 3.2. $A{\in}\mathbb{R}^{a\times c}$ & $B{\in}\mathbb{R}^{c\times b}$ are *aligned* if there exist Singular Value Decompositions (SVDs) $A{=}U_A D_A V_A^T$ and $B{=}U_B D_B V_B^T$ such that $V_A^T U_B{=}I_{c\times c}$.

We note that this input-layer alignment phenomenon has been shown for the top principal component for linear NNs (Ji & Telgarsky, 2018), has been proven & observed empirically for deep non-linear NNs trained on random labels (Maennel et al., 2020), & that first layer alignment has been exploited to design Bayesian NN priors that are robust to covariate shift (Izmailov et al., 2021). Adjacent-layer alignment has been shown for contrastive SSL (Jing et al., 2021).

Proposition 3.3. *Suppose we have an L-layer linear MLP, $f(\boldsymbol{x}){=}\prod_{l=1}^{L} W_l \cdot \boldsymbol{x} \in \mathbb{R}^{d_L}$, trained to convergence using gradient flow & no bias terms on some loss $\mathcal{L}(f(\boldsymbol{X}))$ with weight decay $\eta > 0$. Assume further that the first layer matrix $W_1 \in \mathbb{R}^{d_1\times d}$ & input covariance matrix $\Sigma^{\boldsymbol{x}} \in \mathbb{R}^{d\times d}$ are aligned as in Def. 3.2. Then:*

1. *Adjacent layers' matrices W_l & W_{l-1} become aligned during training for $1 < l \leq L$ so that principal components can be grouped together across layers. If $\lambda_{l,j}$ denotes the j^{th} eigenvalue of the empirical covariance of features at layer l, then:*
2. *For any uncollapsed output eigenvalue j with $\lambda_{L,j}{>}0$ & any two layers $0{\leq}k{<}l$ (where $k = 0$ denotes the input layer), we have:*

$$\lambda_{l,j}{=}(\lambda_{k,j})^{\frac{L-l}{L-k}}(\lambda_{L,j})^{\frac{l-k}{L-k}},$$

i.e. $\lambda_{l,j}$ is a weighted geometric mean between $\lambda_{k,j}$ & $\lambda_{L,j}$, with weighting specified by closeness to k or L.

The proof of Proposition 3.3 can be found in Appendix A, & is largely inspired from previous work (Saxe et al., 2013; Ji & Telgarsky, 2018; Tian et al., 2021; Jing et al., 2021). However, it allows us to deduce that deeper projection MLPs result in a faster decaying encoder eigenspectrum:

Corollary 3.4. *In the setting of Proposition 3.3, suppose we have a fixed encoder depth l_e, & that for some encoder layer $k < l_e$, the feature eigenspectrum at layer k follows power law decay with exponent $\beta > 0$. Then, deeper projection MLPs result in faster encoder eigenvalue decay, if projection outputs are whitened (as in Proposition 3.1).*

Proof. Suppose the projector has depth l_p giving combined depth $L{=}l_e{+}l_p$. We are given $\lambda_{k,j}{=}j^{-\beta}$ & $\lambda_{L,j}{=}1$ up to constant, $\forall j$ (for simplicity of argument here we suppose $a = b$ in Def. 2.1). Applying Proposition 3.3 at the encoder layer, we conclude $\lambda_{l_e,j}{=}j^{-\frac{l_p}{L-k}\beta}$, i.e. power law behaviour with exponent $\frac{l_p\beta}{l_p+l_e-k}$, which is increasing in l_p. □

Remark 3.5. We use power law behaviour in Corollary 3.4 as a convenient medium to express eigenspectrum decay rate, though our conclusion extends to representations without an obvious eigenvalue power law (c.f. Figures 3 and 4).

We now justify our assumptions on eigenspectra in Corollary 3.4. For hidden layers, there are at least two phenomena that encourage hidden feature eigenspectra to decay in practice: 1) the fact that, at the input layer, natural images inherently possess such a power law decaying eigenspectrum (Field, 1987; Ruderman & Bialek, 1994) and 2) it is well known in the signal propagation/wide NN literature that common NN initialisation schemes (e.g. Kaiming (He et al., 2015)) converge to collapsed representations (with an at best polynomial rate) in depth (Schoenholz et al., 2016; Hayou et al., 2019; 2021; Martens et al., 2021). This is corroborated by Figure 3, where both projection & encoders' eigenspectra are dominated by the largest eigenvalue at epoch 0.

For the output eigenspectrum, decorrelation SSL methods are covered by Proposition 3.1 and Figure 3. However, we note that Proposition 3.3 and Corollary 3.4 are agnostic to the specific loss function $\mathcal{L}$ & can also apply to SimCLR with InfoNCE loss (Oord et al., 2018). In Figure 3, we empirically observe for a SimCLR-trained NN that the biggest normalised eigenvalues for projection features (right) are

larger compared to the encoder eigenspectrum (center right), although the effect is weaker for SimCLR compared to Barlow Twins, perhaps due to increased dimensional collapse in the projector (Hua et al., 2021; Jing et al., 2021).

Key takeaways: Whitening and collapse in SSL.

- Trained Barlow Twins NNs have whitened projections.
- Whitened projections don't imply whitened encoders, especially for non-whitened inputs & deeper encoders.
- Deeper projector MLPs may result in more collapsed encoder eigenspectra (empirically verified in Figure 4).

4. Whitened Features Affect Generalisation

In the previous section, we've seen theoretical & empirical evidence that interactions exist between SSL design choices (e.g. projection depth, or choice of method), and the speed of eigenspectrum decay of encoder features. We now demonstrate the rate of decay in the encoder eigenspectrum is important for the quality of learnt SSL representations, in terms of generalisation under linear evaluation. In particular, we show that the relationship between degree of feature whitening & generalisation is not monotonically increasing.

We first seek a quantitative metric to measure how whitened a feature representation is, beyond the power law exponent β of Def. 2.1, as we wish to handle representations that do not possess an obvious eigenvalue power law. Instead, we consider the *normalised eigenvalue sum* (NESum):

Definition 4.1. Given a feature representation $\boldsymbol{\phi}(\boldsymbol{X}) \in \mathbb{R}^{N \times d_\phi}$ with feature-wise covariance matrix $\Sigma \in \mathbb{R}^{d_\phi \times d_\phi}$ and eigenvalues $\{\lambda_i\}_{i=1}^{d_\phi}$ in decreasing order. Then, we define the *normalised eigenvalue sum* to be:

$$\text{NESum}(\{\lambda_i\}_i) \triangleq \sum_{i=1}^{d} \frac{\lambda_i}{\lambda_1}$$

with convention $\frac{0}{0}=0$. NESum takes values in $[0, d_\phi]$, with collapsed features having NESum=0, and NESum=d_ϕ corresponding to exactly whitened features.

Moreover, it is clear via a geometric series argument that for power law decaying feature eigenspectra with exponent β, NESum decreases as β increases, so we view larger values of NESum to denote whiter representations.

In Figure 4, we compare various ResNet-18 trained with Barlow Twins on CIFAR-10, in terms of encoder NESum against test accuracy under linear probe. For projection MLP depths from 1 to 5 & a range of regularisation strengths ρ (on a logarithmic scale from 0.001 to 0.05), we plot markers for NNs trained from three independent initialisations. The different coloured lines denote splines interpolating the three-seed averages across ρ for different depths. The size of the markers correspond to regularisation strength ρ.

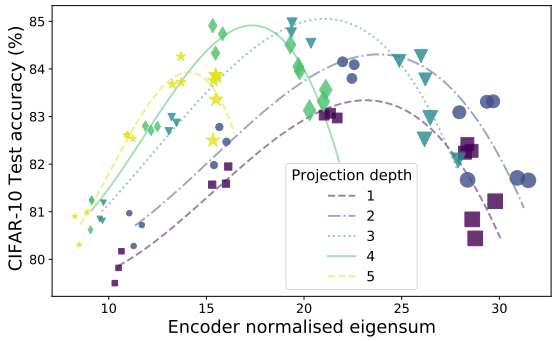


Figure 4. Different Barlow Twins networks on CIFAR-10 with ResNet18 encoder. Each marker corresponds to a trained NN, with marker size denoting regularisation strength ρ. Different coloured lines are spline fits split by projection MLP depth. We see that test accuracy across all depths does not monotonically increase as features become less collapsed.

As suggested by Corollary 3.4, we see that the deeper the projection MLP, the lower NESum across different values of ρ, indicating that encoder NNs trained with shallower projectors are more whitened. Moreover, for each depth, we see that too low NESum results in lower test accuracy, presumably because the features are too collapsed as expected from previous works studying collapse in SSL.

On the other hand, we also see that too high a value of NESum (which we see is due to larger ρ from the marker sizes, as supported by Proposition 3.1) results in lower test accuracy too. This might seem perplexing from the existing feature decorrelation SSL literature where whitening is simply used as a mechanism to avoid feature collapse, and so one might expect a monotonically increasing relationship between NESum & test accuracy.

However, we note that this is somewhat unsurprising given that eigenspectra in biological NNs such as a mouse's visual cortex are known to decay (Stringer et al., 2019). Thus, Figure 4 suggests that in SSL, we should not only seek to avoid too collapsed feature representations, but also too whitened representations. A corresponding figure with STL10 dataset can be found in Appendix B, with similar conclusions.

Key takeaways: Whitening and generalisation in SSL.

- The relationship between extent of feature whitening & generalisation performance in SSL is not monotonic.
- Lower Barlow Twins regulariser ρ yields more collapsed features.

4.1. Insights for Generalisation on Low Labelled Data

To examine the impact of feature whitening in SSL theoretically, we turn to the setting of low labelled data, when $S \ll N$ in the notation of Section 2. We focus on linear evaluation, as opposed to semi-supervised finetuning. Notwithstanding the fact that linear evaluation is one of the

main benchmarks for evaluating SSL methods, nonlinear finetuning in the setting of small labelled-data is vulnerable to tampering with useful features acquired from large-scale unlabelled data. Empirically, we demonstrate this in Section 6 on small labelled-data ImageNet evaluation, where we show that linear evaluation schemes can outperform semi-supervised non-linear finetuning.

Moreover, linear evaluation of SSL lends itself more kindly to theoretical analysis, particularly connections to kernel methods/Gaussian processes (GPs) (Sollich, 1999; Sollich & Halees, 2002; Sollich, 2002; Bordelon et al., 2020; Jin et al., 2021; Cui et al., 2021). These works study *learning curves* of kernel/GP regression, describing how generalisation error changes with the amount of labelled data S.

To this end, we define a kernel $k(\boldsymbol{x},\boldsymbol{x})=\langle h_{\boldsymbol{\theta}}(\boldsymbol{x}), h_{\boldsymbol{\theta}}(\boldsymbol{x}')\rangle$ using the inner product of encoder features. If inputs are assumed to have compact support e.g. normalised, then we can use Mercer's Theorem (Mercer, 1909) to decompose k:

$$k(\boldsymbol{x},\boldsymbol{x}') = \sum_{i=1}^{d_e} \lambda_i \psi_i(\boldsymbol{x})\psi_i(\boldsymbol{x}'), \tag{4}$$

with kernel eigenfunctions $\{\psi_i\}_i$ & eigenvalues $\{\lambda_i\}_i$ (equivalent to the covariance eigenvalues we consider in Def. 2.1) satisfying $\int k(\boldsymbol{x},\boldsymbol{x}')\psi_i(\boldsymbol{x}')p(\boldsymbol{x}')d\boldsymbol{x}' = \lambda_i\psi_i(\boldsymbol{x})$.

We consider a single output $C=1$ for simplicity, as the results we use extend straightforwardly for $C>1$ (Bordelon et al., 2020). We also consider squared error instead of cross-entropy, noting that solving classification tasks with squared error (by treating labels as one-hot regression targets) is often used due to the connection with kernels (Lee et al., 2019; Shankar et al., 2020; Lee et al., 2020; He et al., 2020a). In this case, we obtain trained predictor $\hat{f}_S(\boldsymbol{x})=k(\boldsymbol{x},\boldsymbol{X}_S)\left(K_{\boldsymbol{X}_S,\boldsymbol{X}_S}+\sigma^2 I_S\right)^{-1}\boldsymbol{Y}_S$ from Eq. (1), where $K_{\boldsymbol{X}_S,\boldsymbol{X}_S}\in\mathbb{R}^{S\times S}$ is the Gram matrix of $h(\boldsymbol{X}_S)$. Finally, let us assume we have noiseless observations $q(\boldsymbol{y}|\boldsymbol{x})=\delta_{f^*(\boldsymbol{x})}(\boldsymbol{y})$ & the true target function f^* satisfies:

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{d_e} \mu_i \psi_i(\boldsymbol{x}). \tag{5}$$

Then, existing works have derived the learning curves for generalisation error of $\hat{f}_S$, $\hat{\mathcal{E}}_S=\mathbb{E}_{p(\boldsymbol{x})}[\left(\hat{f}(\boldsymbol{x})-f^*(\boldsymbol{x})\right)^2]$, when k & f^* both observe power law behaviour:

Proposition 4.2 (Bordelon et al. (2020); Jin et al. (2021))**.** *If $\lambda_i=\Theta(i^{-\beta})$ & $\mu_i^2=\Theta(i^{-\alpha})$, $\forall i$, with $\alpha>1$ to ensure f^* square integrable, then as $S\to\infty$, we have $\hat{\mathcal{E}}_S=\Theta(S^{\frac{1-\alpha}{\beta}})$.*

We see in Proposition 4.2 that if $\{\lambda_i\}_i$ & $\{\mu_i\}_i$ observe power laws, then so does the generalisation error $\hat{\mathcal{E}}_S$. Moreover, for fixed f^* (& $\alpha > 1$), larger β results in slower decaying error as S increases. This allows us to conclude:

Corollary 4.3. *Let k' be defined like k in Eq. (4) but with new eigenvalues $\{\lambda_i'\}_i$, & corresponding generalisation error $\hat{\mathcal{E}}_S'$. If $\lambda_i' \sim \Theta(i^{-\beta'})\ \forall i$, then we have $\log\frac{\hat{\mathcal{E}}_S}{\hat{\mathcal{E}}_S'} \sim c + (1-\alpha)(\beta^{-1}-\beta'^{-1})\log(S)$, for some c.*

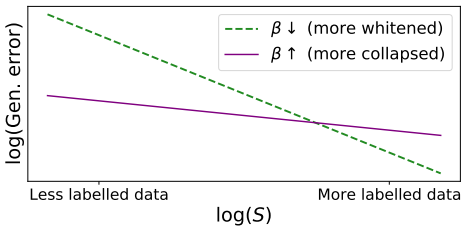


Figure 5. Power law in S for generalisation error $\hat{\mathcal{E}}_S$.

Figure 5 illustrates Corollary 4.3's implications: kernels with small β enjoy fast decaying error curves as $S\to\infty$, but this means for low S, the relative error is larger for small β. In Figure 5, we see that a more collapsed feature eigenspectrum can perform better at low labelled data S, despite performing worse at larger S. Though hypothetical, Figure 5 shows that one can alter generalisation by simply regulating eigenvalue decay, which we utilise in Section 5.

In Appendix A, we prove Theorem A.2, which shows that kernels with whitened eigenspectra may perform relatively worse at low S, compared to unwhitened eigenspectra, for f^* without power law assumptions.

Key takeaways: Generalisation and labelled data size.

- Decay rate in feature eigenspectra affects how generalisation changes with labelled data size.
- More collapsed features may perform relatively better than more whitened features on small labelled data.

5. PostMan-Pat: Post-hoc Manipulation of the Principal Axes & Trace

In the previous section, we have seen that the degree to which features are whitened has an impact on generalisation error, particularly when one has varying amounts of labelled data S. The motivation for our methodological contribution is now clear: one can affect (& potentially improve) the generalisation performance of SSL simply by explicitly controlling the eigenspectrum decay of trained SSL methods.

Our method, named Post-hoc Manipulation of the Principal Axes & Trace or PostMan-Pat (PMP), rescales the principal components of *any* encoder's covariance Σ_h *after pretraining* to enforce a power law decay in the eigenvalues $\{\lambda_i\}_i$, with exponent β acting as a hyperparameter. We do so efficiently by estimating the encoder covariance matrix $\Sigma_h \in \mathbb{R}^{d_e\times d_e}$ using our unlabelled data $\boldsymbol{X}$, from which

we derive an untrainable rescaling matrix $W_{\text{PMP}} \in \mathbb{R}^{d_e \times d_e}$, detailed in Alg. 1. Then, we redefine our pretrained encoder by appending W_{PMP}, $h_{\text{PMP}}(\boldsymbol{x}) \leftarrow h_{\boldsymbol{\theta}}(\boldsymbol{x}) W_{\text{PMP}}$, before training linear classifier W_C in Eq. (1) as before. Pseudocode for PMP is provided in Alg. 1.

Algorithm 1 PyTorch pseudocode for PostMan-Pat (PMP).

```
# h: Pretrained encoder (with standardised neurons).
# beta: Power law exponent.
# base_rank: Rank from which to start power law.
# B: Batch size.
# N: Unlabelled data size.
# D: Dimensionality of the encoder embeddings.
# harmonic: D-dim tensor with the i^th element 1/(i+1).
# mm: Matrix-matrix multiplication.
# eig: SVD operator (eigenvalues in decreasing order).

class PMPEncoder(nn.Module):
    def __init__(self, encoder, W_pmp):
        super().__init__()
        self.encoder = encoder
        W_pmp.requires_grad = False # Fixed
        self.W_pmp = W_pmp

    def forward(self, x):
        x = self.encoder(x)       # 1xD
        return mm(x, self.W_pmp) # 1xD

# Compute feature covariance matrix.
cov = torch.zeros(D, D)
for x in loader:
    z = h(x)                  # BxD
    cov += mm(z.T, z) / N # DxD

# Compute W_pmp.
eig_vals, eig_vecs = eig(cov)
eig_ratio = eig_vals[base_rank] / eig_vals[base_rank:]
eig_rescaled = torch.ones(D)
eig_rescaled[base_rank:] = eig_ratio * \
        (base_rank * harmonic[base_rank:]).pow(beta)

W_pmp_sqrt = eig_vecs * eig_rescaled.sqrt() # DxD
W_pmp = mm(W_pmp_sqrt, eig_vecs.T)          # DxD

# New PMP encoder for linear evaluation in Eq.(1).
h_pmp = PMPEncoder(h, W_pmp)
```

We next show that PMP does indeed result in a power law decaying eigenspectra with exponent β:

Proposition 5.1. *The PMP encoder h_{PMP}, as described in Alg. 1, has i) a β-power law eigenspectrum, & ii) the same left & right eigenvectors as h.*

Proof. Let $UD^{\frac{1}{2}}V^T$ be SVD of $h(\boldsymbol{X})$, so that $\Sigma_h = \frac{1}{N} V D V^T$. We defined $W_{\text{PMP}} \triangleq V R^{\frac{1}{2}} V^T$ in Alg. 1, for R diagonal & $R_{ii} = \frac{D_{rr}}{D_{ii}} (\frac{r}{i})^{\beta}$ if $i \geq r$ & 1 else, where r denotes the base rank.

So if $h_{\text{PMP}}(\boldsymbol{x}) = h(\boldsymbol{x}) W_{\text{PMP}}$, then $h_{\text{PMP}}(\boldsymbol{X}) = h(\boldsymbol{X}) W_{\text{PMP}}$ has SVD: $UD_{\text{PMP}}^{\frac{1}{2}} V^T$ where D_{PMP} satisfies $\left(D_{\text{PMP}}\right)_{ii} = D_{rr} (\frac{r}{i})^{\beta}$ if $i \geq r$ & D_{ii} else. □

As ℓ_2-regularisation/weight decay can be thought of as minimising an unregularised version of Eq. (1), $\sum_{s=1}^{S} l(f(\boldsymbol{x}_s), \boldsymbol{y}_s)$, subject to constraints on the ℓ_2-norm of W_C, we see that PMP can be viewed as an alternative to linear probes, but with eigendecomposition specific constraints. We speculate this specificity allows PMP to outperform linear probes on complex datasets, like ImageNet in Section 6.

PMP can also be viewed as using a 2-layer linear MLP classifier, instead of W_C in Eq. (1), but with fixed first layer. Training both layers in a 2-layer MLP adds non-convexity to the optimisation process: we compare PMP to a 2-layer linear MLP classifier with all layers trainable in Section 6.

Implementation details PMP requires a single eigendecomposition of Σ_h, which is 2048×2048 for a ResNet-50, so adds minimal cost to the standard ImageNet benchmark for SSL. To compute Σ_h, we need a single forward pass over unlabelled data $\boldsymbol{X}$, or we can estimate Σ_h with a moving average online. We standardise the d_e neurons in h to have zero mean and unit variance to avoid non-zero means resulting in a dominant largest eigenvalue. Though Corollary 4.3 and Figure 5 suggest larger values of β may be preferable with smaller S, we stress there are interactions with ℓ_2 regularisation not covered by our theory (c.f. Figure 10), & it is important to tune hyperparameters (as usual for complex machine learning tasks), in PMP's case β & W_C's weight decay, for best results.

6. Experiments

In interest of space, experimental details not covered in the main paper can be found in Appendix C. In all PMP experiments we start the power-law behaviour after the tenth largest eigenvalue, in line with Nassar et al. (2020), who studied importance of power-law decay in eigenspectra for adversarial robustness in NNs.

Analysis of PMP on CIFAR-10 In Figure 6, we examine the efficacy of PMP on CIFAR-10 for Barlow Twins, SimCLR and a pretrained supervised NN compared to standard linear probe evaluation. All methods use ResNet-18 encoder. On the leftmost column, we plot eigenspectra for different values of power law exponent β, compared to the original encoder. The linear trends (after the 10th largest eigenvalue) observed on log-log scale indicate that PMP successfully induces a power law, with larger β resulting in faster decaying eigenvalues. SSL methods on the left-most column have projector depth 2.

In all other columns, we plot the relative change in test accuracy when using PMP compared to standard linear probe (on the same pretrained encoder), as a function of the power law β. Error bars indicate 95% confidence over 20 data splits (if applicable) & 3 independent encoder initialisations.

We see that the performance of PMP is monotonically increasing in β for the lowest amounts of labelled data, achieving up to 4% higher test accuracy for Barlow Twins when 0.1% labelled data is available (i.e. 5 examples per CIFAR-10 class), and that performance drops off dramatically for

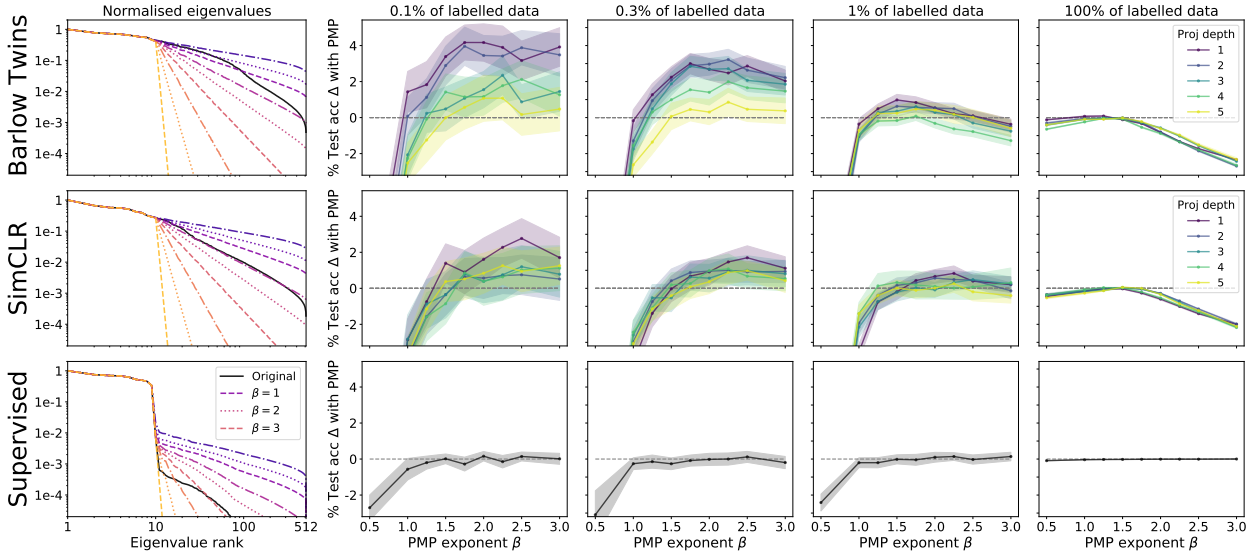


Figure 6. PMP eigenspectra (left) & CIFAR-10 test acc. relative to standard linear probe for varying labelled data sizes across methods.

low β when the features are more whitened. On the other hand, when more labelled data is available, we observe drop-offs in performance using PMP with larger β, although with a well-tuned β, PMP always at least matches the standard linear probe in test-accuracy across all settings. These observations are to be expected from Corollary 4.3: suppressing the useful tail eigenmodes is helpful when not enough labelled data is available to learn them, but can harm performance when there is sufficient labelled data.

We also observe that encoders trained with deeper projection layers benefit less from PMP in small S regimes. This is again suggested by Corollary 3.4, as deeper projection layers already have more collapsed encoder eigenspectra.

Finally, we note that PMP does not seem to be very effective for supervised encoders, which may be because we observe in Figure 6 (bottom left) that supervised training encourages the ResNet-18 encoder to have 10 dominant eigenvalues (corresponding to the 10 classes of CIFAR-10, c.f. Figure 8), and that increasing the size of the smaller eigenvalues via lower β hurts performance, suggesting that the smaller eigenmodes contain unuseful features.

ImageNet-1K For a given pretrained ResNet-50 encoder $h_{\boldsymbol{\theta}}$, we compare PMP to other evaluation schemes on differing amounts of ImageNet-1K labelled data. We compare PMP both to baseline schemes that are also linear in the pretrained encoder $h_{\boldsymbol{\theta}}$: i) standard linear probe (LP), & ii) replacing W_C with a 2-layer linear MLP (MLP) in Eq. (1), as well as non-linear finetuning (NFT), where encoder $h_{\boldsymbol{\theta}}$ is also trainable. We use the same data splits for 1% & 10% as provided by Chen et al. (2020a), and for 0.3%, we sample 3 independent subsets from the 1% split to provide standard deviations. We use a validation split from the accessible training labels to tune hyperparameters for all evaluation schemes, c.f. Appendix C.

In Table 1, we see that PMP improves considerably over other linear evaluations (LP & MLP) over a range of labelled-data sizes & SSL methods: Barlow Twins, SimCLR, & SwAV (Caron et al., 2020). Moreover, we see that PMP consistently outperforms NFT with Barlow Twins & SwAV encoders for smaller values of labelled data, despite keeping $h_{\boldsymbol{\theta}}$ fixed. For example, with 0.3% labelled data & Barlow Twins encoder, PMP obtains 42.3% top-1 accuracy compared to 40.7% for NFT. Likewise, for 1% labelled data & SwAV, PMP beats NFT by over 2.5% in top-1 accuracy. Our top-1 accuracy of 56.2% (obtained via PMP with Barlow Twins pretraining) is to our knowledge the best reported result for an SSL-pretrained ResNet-50 encoder linearly evaluated on 1% ImageNet-1K labels.

Interestingly, standard LP with SwAV can also outperform NFT under label scarcity, where encoder updates may be susceptible to intefering with the useful features acquired during SSL pretraining. However, we also find that NFT outperforms all linear evaluation schemes for SimCLR. Additional experiments are provided in Appendix B including: analysis of the effect of weight decay on PMP in low-labelled data (Figure 10) and evaluation on the ImageNetV2 (Recht et al., 2019) test sets (Table 4).

Transfer Learning We next investigate the ability of PMP to improve transferability of SSL features to new datasets. Using ResNet-50 encoders pretrained on ImageNet-1K, we compare linear probing against PMP on a variety of downstream image classification tasks: CIFAR-100 (Krizhevsky, 2009), Stanford Cars (Krause et al., 2013) and Oxford 102 Flowers (Nilsback & Zisserman, 2008)

Table 1. ImageNet-1K validation accuracy (%) of PMP against standard SSL evaluation schemes, across pretrained checkpoints, with low labelled-data (0.3%, 1%, or 10% labels). Supervised results are from Zhai et al. (2019). Top-1 accuracies with LP on 100% labelled-data are in brackets. 1st & 2nd best results per SSL method & labelled-data level are **bold** & underlined respectively.

Method		Top-1			Top-5		
Pretrain	Eval	0.3%	1%	10%	0.3%	1%	10%
SimCLR (69.3)	LP	$34.2_{\pm.2}$	48.1	61.0	$57.2_{\pm.3}$	73.8	84.3
	MLP	$31.8_{\pm.4}$	45.2	61.5	$54.1_{\pm.3}$	71.1	85.0
	PMP	$35.9_{\pm.2}$	50.9	62.5	$57.9_{\pm.2}$	76.6	85.2
	NFT	**39.8**$_{\pm.2}$	**52.5**	**67.5**	**65.5**$_{\pm.2}$	**78.9**	**88.7**
SwAV (74.7)	LP	$36.5_{\pm.3}$	53.8	68.2	$61.8_{\pm.1}$	78.8	88.8
	MLP	$34.6_{\pm.4}$	52.0	67.5	$59.3_{\pm.2}$	77.2	88.6
	PMP	**39.3**$_{\pm.3}$	**55.9**	68.5	**64.1**$_{\pm.2}$	**79.7**	89.1
	NFT	$32.4_{\pm.3}$	53.6	**70.8**	$57.8_{\pm.2}$	79.1	**90.5**
Barlow (73.5)	LP	$39.9_{\pm.1}$	55.0	63.2	$63.8_{\pm.2}$	79.0	83.4
	MLP	$37.6_{\pm.1}$	53.0	66.3	$61.5_{\pm.1}$	76.9	87.2
	PMP	**42.3**$_{\pm.1}$	**56.2**	67.3	**65.8**$_{\pm.1}$	**79.7**	88.6
	NFT	$40.7_{\pm.1}$	55.3	**70.0**	**65.8**$_{\pm.1}$	79.6	**89.9**
Supervised		–	25.4	56.4	–	48.4	80.4

in Table 2. Across transfer datasets and pretrained SSL encoders, we observe that our PostMan-Pat (PMP) outperforms linear probe (LP) evaluation. We use 100% labelled training data from the transfer dataset in Table 1, demonstrating that while one motivation for PMP was for low-labelled data (Section 4.1), PMP can still outperform LP on large-labelled data settings.

Table 2. **Transfer Learning**: Comparison of top-1 test accuracies (%) for PMP and LP across SSL methods and transfer datasets.

Method		Transfer Dataset		
Pretrain	Eval	C-100	Cars	Flowers
SimCLR	LP	65.26	46.88	84.65
	PMP	66.13	47.83	85.88
Barlow	LP	74.19	69.36	92.29
	PMP	75.10	69.67	92.54
SwAV	LP	75.24	63.39	90.47
	PMP	76.10	64.46	92.00

Different Architectures To assess whether PMP is able to improve label efficiency across different encoder architectures, Table 3 shows compares linear probing and PMP for a ViT-B/16 vision transformer (Dosovitskiy et al., 2020) pretrained using MoCo-v3 (Chen et al., 2021) on ImageNet-1K. We see that PMP consistently improves against LP for ViT-B/16 with lower labelled settings, e.g. 55.2% vs. 54.0% Top-1 on 0.3% labels, and 65.1% vs. 64.5% on 1% labels, although the improvement is more modest on 10% labels. Understanding better how the encoder architecture impacts SSL features (beyond Corollary 3.4), and hence PMP's effectiveness, is interesting future work.

Table 3. **Different Architecture:** PMP and LP ImageNet test accuracies on 1% and 10% labels using ViT-B/16 with MoCo-v3.

	Top-1			Top-5		
Eval	0.3%	1%	10%	0.3%	1%	10%
LP	$54.0_{\pm.3}$	64.5	72.3	$78.0_{\pm.3}$	86.6	91.2
PMP	$55.2_{\pm.2}$	65.1	72.4	$78.7_{\pm.2}$	86.8	91.2

7. Summary & Discussion

Inspired by work on feature whitening self-supervised learning (SSL) to avoid collapse, we explored the gap between whitening & collapse in SSL. We identified power law behaviour in feature eigenspectra decay as a possible way to bridge between these extremes, and studied the design choices in SSL that affect the rate at which feature eigenvalues decay. We demonstrated theoretically & empirically that weaker regularisation in Barlow Twins & deeper projector layers lead to more collapsed encoders. Moreover, we found empirically that generalisation performance in SSL is non-monotonic in the degree of feature whitening, & highlighted the significance of considering feature eigenspectrum decay in label scarce settings. Finally, we used our insights to motivate a novel post-processing method: PostMan-Pat (PMP) that efficiently enforces a power law in encoder eigenvalues, & demonstrated the ability of PMP to outperform other linear schemes (consistently) & non-linear finetuning (at times) across low labelled-data settings. We hope that the improved label efficiency of PMP can be applied to practical settings of label scarcity. More generally, we hope that our work leads to further progress in SSL by highlighting both the spectrum that exists between collapsed & whitened features, and that where one lies along this spectrum is significant for generalisation performance & label efficiency.

We point out that although we provide empirical evidence from practical settings to corroborate our theoretical results, our theory has some non-standard assumptions to ease analytical exposition, such as linear projector MLPs, much like related theoretical work in SSL (Tian et al., 2021; Wang et al., 2021; Jing et al., 2021). Moreover, we have not studied the actual features acquired during SSL pretraining, e.g. the impact of transformation choice, instead showing that one can improve generalisation accuracy simply by rescaling pre-defined features. Finally, compared to standard linear probes, PMP introduces a new hyperparameter β which needs to be tuned, although it is worth noting that PMP has far fewer significant hyperparameters than non-linear finetuning. For future work, it would be interesting to design an SSL method that directly factors in the rate of feature eigenvalue decay into the pretraining regime, & also to study tuning schemes for PMP hyperparameters.

References

Bardes, A., Ponce, J., and LeCun, Y. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

Bordelon, B., Canatar, A., and Pehlevan, C. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning*, pp. 1024–1034. PMLR, 2020.

Bordes, F., Balestriero, R., and Vincent, P. High fidelity visualization of what your self-supervised representation knows about. *arXiv preprint arXiv:2112.09164*, 2021.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020b.

Chen, X. and He, K. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

Chen, X., Fan, H., Girshick, R., and He, K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020c.

Chen, X., Xie, S., and He, K. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9640–9649, 2021.

Cui, H., Loureiro, B., Krzakala, F., and Zdeborová, L. Generalization error rates in kernel regression: The crossover from the noiseless to noisy regime. *arXiv preprint arXiv:2105.15004*, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Ermolov, A., Siarohin, A., Sangineto, E., and Sebe, N. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pp. 3015–3024. PMLR, 2021.

Field, D. J. Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A*, 4(12):2379–2394, Dec 1987.

Foster, A., Pukdee, R., and Rainforth, T. Improving transformation invariance in contrastive representation learning. In *International Conference on Learning Representations*, 2021.

Goyal, P., Duval, Q., Reizenstein, J., Leavitt, M., Xu, M., Lefaudeux, B., Singh, M., Reis, V., Caron, M., Bojanowski, P., Joulin, A., and Misra, I. Vissl, 2021.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Hayou, S., Doucet, A., and Rousseau, J. On the impact of the activation function on deep neural networks training. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2672–2680. PMLR, 09–15 Jun 2019.

Hayou, S., Clerico, E., He, B., Deligiannidis, G., Doucet, A., and Rousseau, J. Stable resnet. In *International Conference on Artificial Intelligence and Statistics*, pp. 1324–1332. PMLR, 2021.

He, B., Lakshminarayanan, B., and Teh, Y. W. Bayesian deep ensembles via the neural tangent kernel. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1010–1022. Curran Associates, Inc., 2020a.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020b.

Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., and Zhao, H. On feature decorrelation in self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9598–9608, 2021.

Izmailov, P., Nicholson, P., Lotfi, S., and Wilson, A. G. Dangers of bayesian model averaging under covariate shift. *arXiv preprint arXiv:2106.11905*, 2021.

Ji, Z. and Telgarsky, M. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations*, 2018.

Jin, H., Banerjee, P. K., and Montúfar, G. Learning curves for gaussian process regression with power-law priors and targets. *arXiv preprint arXiv:2110.12231*, 2021.

Jing, L., Vincent, P., LeCun, Y., and Tian, Y. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32: 8572–8583, 2019.

Lee, J., Schoenholz, S., Pennington, J., Adlam, B., Xiao, L., Novak, R., and Sohl-Dickstein, J. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33, 2020.

Li, Y., Pogodin, R., Sutherland, D. J., and Gretton, A. Self-supervised learning with kernel dependence maximization. *arXiv preprint arXiv:2106.08320*, 2021.

Maennel, H., Alabdulmohsin, I. M., Tolstikhin, I. O., Baldock, R., Bousquet, O., Gelly, S., and Keysers, D. What do neural networks learn when trained with random labels? In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19693–19704. Curran Associates, Inc., 2020.

Martens, J., Ballard, A., Desjardins, G., Swirszcz, G., Dalibard, V., Sohl-Dickstein, J., and Schoenholz, S. S. Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *arXiv preprint arXiv:2110.01765*, 2021.

Mercer, J. Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations. *Philosophical Transactions of the Royal Society of London Series A*, 209:415–446, January 1909. doi: 10.1098/rsta.1909.0016.

Nassar, J., Sokol, P., Chung, S., Harris, K. D., and Park, I. M. On 1/n neural representation and robustness. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6211–6222. Curran Associates, Inc., 2020.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, pp. 722–729, 2008. doi: 10.1109/ICVGIP.2008.47.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Papyan, V., Han, X. Y., and Donoho, D. L. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020. ISSN 0027-8424. doi: 10.1073/pnas.2015509117. URL https://www.pnas.org/content/117/40/24652.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.

Ruderman, D. and Bialek, W. Statistics of natural images: Scaling in the woods. In Cowan, J., Tesauro, G., and Alspector, J. (eds.), *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1994.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Shankar, V., Fang, A., Guo, W., Fridovich-Keil, S., Ragan-Kelley, J., Schmidt, L., and Recht, B. Neural kernels without tangents. In *International Conference on Machine Learning*, pp. 8614–8623. PMLR, 2020.

Sollich, P. Learning curves for gaussian processes. In Kearns, M., Solla, S., and Cohn, D. (eds.), *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1999.

Sollich, P. Gaussian process regression with mismatched models. In Dietterich, T., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

Sollich, P. and Halees, A. Learning curves for gaussian process regression: Approximations and bounds. *Neural computation*, 14(6):1393–1428, 2002.

Stringer, C., Pachitariu, M., Steinmetz, N., Carandini, M., and Harris, K. D. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765): 361–365, 2019.

Tian, Y., Chen, X., and Ganguli, S. Understanding self-supervised learning dynamics without contrastive pairs. *arXiv preprint arXiv:2102.06810*, 2021.

Wang, X., Chen, X., Du, S. S., and Tian, Y. Towards demystifying representation learning with non-contrastive self-supervision. *arXiv preprint arXiv:2110.04947*, 2021.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1476–1485, 2019.

Zhang, C., Zhang, K., Zhang, C., Pham, T. X., Yoo, C. D., and Kweon, I. S. How does simsiam avoid collapse without negative samples? a unified understanding with self-supervised contrastive learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=bwq6O4Cwdl.

A. Proofs & Additional Results

Throughout, we use o, O, Θ to denote standard mathematical notation for order size.

A.1. Proposition 3.1

We start by proving Proposition 3.1.

Proposition 3.1. *Suppose an NN encoder+projection trained via Barlow Twins (Zbontar et al., 2021) achieves (i) training loss $\mathcal{L}_{BT} = \epsilon$, & ii) $\exists a \leq b$ positive constants such that $a \leq \Sigma^z_{ii} \leq b$, $\forall i \in [d_p]$. Then, if $\epsilon \leq \frac{a^2\rho}{b^2}$, the projector has a whitened eigenspectrum.*

Proof. Because $\mathcal{L}_{\text{BT}} = \epsilon$, we see that the off-diagonal contribution to the loss satisfies:

$$\sum_{i=1}^{d_p} \sum_{j \neq i} C_{i,j}^2 \leq \frac{\epsilon}{\rho}.$$

From Eq. (3), we see that $C_{ij} = \frac{\Sigma^z_{i,j}}{\sqrt{\Sigma^z_{i,i}\Sigma^z_{j,j}}}$ up to transformations (see remark below), and from assumption ii), we deduce that:

$$\sum_{i=1}^{d_p} \sum_{j \neq i} (\Sigma^z_{i,j})^2 \leq \frac{b^2\epsilon}{\rho}.$$

Thus, if $\text{diag}(\Sigma^z)$ denotes the diagonal version of Σ^z , then we see that

$$\|\text{diag}(\Sigma^z) - \Sigma^z\|_F^2 \leq \frac{b^2\epsilon}{\rho}.$$

Recall that the squared Frobenius norm of a matrix is also the sum of its squared eigenvalues. So if $\frac{b^2\epsilon}{\rho} < a^2$, then we have a whitened projection eigenspectrum of Σ^z, with new constants $a' = \sqrt{a^2 - \frac{b^2\epsilon}{\rho}}$ and $b' = \sqrt{b^2 + \frac{b^2\epsilon}{\rho}}$ satisfying Def. 2.1.

□

Remark A.1. Though it is not strictly true that $C_{ij} = \frac{\Sigma^z_{i,j}}{\sqrt{\Sigma^z_{i,i}\Sigma^z_{j,j}}}$ due to the input transformations in C_{ij}, we note that test-time feature-averaging across transformations has been shown to be an effective method (at the expense of extra forward passes) to improve generalisation of contrastive SSL (Foster et al., 2021).

In this case, we can replace features $z(\boldsymbol{x}) \in \mathbb{R}^{d_p}$ with augmented features $Z(\boldsymbol{x}) \in \mathbb{R}^{d_p \times K}$, where $Z(\boldsymbol{x})_{i,k} = z_i(T_k(\boldsymbol{x}))$, corresponding to K independently sampled transformations $\{T_k\}_{k=1}^K \overset{\text{i.i.d.}}{\sim} \mathcal{T}$.

Then, the natural covariance definition $\Sigma^Z_{i,j} = \frac{1}{NK}\sum_{n=1}^N \left(Z(\boldsymbol{x}_n)Z(\boldsymbol{x}_n)^T\right)_{i,j}$ does satisfy $C_{ij} = \frac{\Sigma^Z_{i,j}}{\sqrt{\Sigma^Z_{i,i}\Sigma^Z_{j,j}}}$ in the $K \to \infty$ limit.

A.2. Proposition 3.3

We next prove Proposition 3.3, restated below:

Proposition 3.3. *Suppose we have an L-layer linear MLP, $f(\boldsymbol{x}) = \prod_{l=1}^L W_l \cdot \boldsymbol{x} \in \mathbb{R}^{d_L}$, trained to convergence using gradient flow & no bias terms on some loss $\mathcal{L}(f(\boldsymbol{X}))$ with weight decay $\eta > 0$. Assume further that the first layer matrix $W_1 \in \mathbb{R}^{d_1 \times d}$ & input covariance matrix $\Sigma^{\boldsymbol{x}} \in \mathbb{R}^{d \times d}$ are aligned as in Def. 3.2. Then:*

1. *Adjacent layers' matrices W_l & W_{l-1} become aligned during training for $1 < l \leq L$ so that principal components can be grouped together across layers. If $\lambda_{l,j}$ denotes the j^{th} eigenvalue of the empirical covariance of features at layer l, then:*

2. *For any uncollapsed output eigenvalue j with $\lambda_{L,j}>0$ & any two layers $0\leq k<l$ (where $k=0$ denotes the input layer), we have:*

$$\lambda_{l,j}=(\lambda_{k,j})^{\frac{L-l}{L-k}}(\lambda_{L,j})^{\frac{l-k}{L-k}},$$

i.e. $\lambda_{l,j}$ is a weighted geometric mean between $\lambda_{k,j}$ & $\lambda_{L,j}$, with weighting specified by closeness to k or L.

Proof. If our linear MLP is $f(\boldsymbol{x})=\prod_{l=1}^{L}W_l\cdot\boldsymbol{x}$ with weight matrices $W_l\in\mathbb{R}^{d_l\times d_{l-1}}$, then gradient flow on loss $\mathcal{L}$ with weight decay $\eta>0$ yields dynamics (c.f. (Ji & Telgarsky, 2018)):

$$\dot{W}_l=-W_{l+1}^T\cdots W_L^T\chi^T W_1^T\cdots W_{l-1}^T-\eta W_l, \tag{6}$$

$\forall l\geq 1$, where:

$$\chi=\boldsymbol{X}^T\frac{\partial\mathcal{L}(f)}{\partial f}\bigg|_{f=f(\boldsymbol{X})}\in\mathbb{R}^{d\times d_L}. \tag{7}$$

We see from Eq. (6) that:

$$W_l^T\dot{W}_l+\eta W_l^T W_l=\dot{W}_{l-1}W_{l-1}^T+\eta W_{l-1}W_{l-1}^T, \tag{8}$$

and taking the transpose of Eq. (8) yields:

$$\dot{W}_l^T W_l+\eta W_l^T W_l=W_{l-1}\dot{W}_{l-1}^T+\eta W_{l-1}W_{l-1}^T, \tag{9}$$

Taking the sum of Eqs. (8) and (9), and integrating both sides with respect to time gives us:

$$W_l^T W_l=W_{l-1}W_{l-1}^T+C_l e^{-2\eta t}, \tag{10}$$

for some constant matrix C_l. Thus, we see that $\lim_{t\to\infty}W_l^T W_l-W_{l-1}W_{l-1}^T=0$. If we let $U_l D_l V_l^T$ denote the Singular Value Decomposition (SVD) of W_l in this limit, then (noting that SVDs are unique only up to permutations of the principal components, so we are free to choose a permutation that ensures alignment) we see that:

- $V_l^T U_{l-1}=I_{d_{l-1}},\ \forall l\geq 2.$
- $D_L=\cdots=D_l=D_{l-1}=\cdots=D_1.$

This concludes the proof for 1).

For 2), recall we assumed that $V_1^T U_{\boldsymbol{x}}=I$, where the empirical input covariance has SVD $\Sigma^{\boldsymbol{x}}=U_{\boldsymbol{x}}D_{\boldsymbol{x}}^2U_{\boldsymbol{x}}^T$. For simplicity, we can assume that the input distribution is centred $\frac{1}{N}\sum_n\boldsymbol{x}_n=0$ without loss of generality due to linearity, which ensures that the empirical output distribution is too: $\frac{1}{N}\sum_n f(\boldsymbol{x}_n)=0$.

Thus by construction, the empirical input covariance is $\Sigma^{\boldsymbol{x}}=\frac{1}{N}\sum_n\boldsymbol{x}_n\boldsymbol{x}_n^T$, so we can put everything together and calculate the output covariance to be:

$$\begin{aligned}
\Sigma^L&=\frac{1}{N}\sum_n f(\boldsymbol{x}_n)f(\boldsymbol{x}_n)^T\in\mathbb{R}^{d_L\times d_L} &(11)\\
&=W_L\cdots W_1\cdot\Sigma^{\boldsymbol{x}}\cdot W_1^T\cdots W_L^T &(12)\\
&=U_L\Big(\prod_{l=1}^{L}D_l^2\Big)D_{\boldsymbol{x}}^2U_L^T &(13)\\
&=U_L D_1^{2L}D_{\boldsymbol{x}}^2U_L^T &(14)
\end{aligned}$$

Let us denote $\gamma_j=(D_1)_{j,j}^2$ and $\lambda_{0,j}=(D_{\boldsymbol{x}})_{j,j}^2$, then we see from Eq. (14) that $\lambda_{L,j}=\gamma_j^L\lambda_{0,j}$, such that $\gamma_j=(\frac{\lambda_{L,j}}{\lambda_{0,j}})^{\frac{1}{L}}$.

In a similar vein to Eqs. (11) to (14), it is simple to calculate the empirical covariance at layer $l \geq 0$ by

$$\Sigma^l = U_l D_1^{2l} D_{\boldsymbol{x}}^2 U_l^T. \tag{15}$$

We thus conclude that $\lambda_{l,j} = \lambda_{0,j}(\frac{\lambda_{L,j}}{\lambda_{0,j}})^{\frac{l}{L}} = (\lambda_{0,j})^{\frac{L-l}{L}}(\lambda_{L,j})^{\frac{l}{L}} \quad \forall l$, from which the result 2) follows easily, by noting that $\left(\frac{\lambda_{l,j}}{(\lambda_{L,j})^{\frac{l}{L}}}\right)^{\frac{1}{L-l}}$ is constant in l: i.e. one can set $\left(\frac{\lambda_{l,j}}{(\lambda_{L,j})^{\frac{l}{L}}}\right)^{\frac{1}{L-l}} = \left(\frac{\lambda_{k,j}}{(\lambda_{L,j})^{\frac{k}{L}}}\right)^{\frac{1}{L-k}}$ and simplify. □

A.3. Theorem A.2

Finally, we state and prove Theorem A.2. Suppose that we have two kernels k_1 and k_2, with which we want to learn two (potentially different) functions f_1^* and f_2^* respectively using kernel ridge regression. If one kernel has whitened eigenvalues but the other does not, then the whitened kernel has relatively worse generalisation error at a particular value of labelled data $S^* < \infty$, under certain assumptions. Note Theorem A.2 does not require power law assumptions on f_a^*, unlike Proposition 4.2.

The intuition is that all eigenmodes are learnt as $S \to \infty$, but at a large but intermediate value of S^* (depending on ridge parameter σ^2 and the level of imbalance in eigenvalue sizes), only the largest eigenvalues are learnt. As it is harder to learn the smaller eigenvalues (in the sense that we need more data to learn the same size eigenmode μ_i^2 if eigenvalue λ_i is smaller, c.f. Eq. (20)) compared to larger eigenvalues, at this intermediate value of S^* it is possible to have relatively lower error at the small eigenmodes by not learning the small eigenmodes (and only learning the dominant eigenmodes), compared to larger values of S:

Theorem A.2. *For $a \in [2]$, let kernel k_a have sorted (in decreasing order) eigenvalues $\{\lambda_{a,i}\}_{i=1}^{d_a}$ & eigenfunctions $\{\psi_{a,i}\}_{i=1}^{d_a}$.*

Let f_a^* be such that $\exists \{\mu_{a,i}\}_{i=1}^{d_a}$ satisfying Eq. (5) with $\{\psi_{a,i}\}_i$, i.e. $f_a^*(\boldsymbol{x}) = \sum_{i=1}^{d_a} \mu_{a,i}\psi_{a,i}(\boldsymbol{x})$.

Suppose $\frac{\mu_{a,i}^2}{\lambda_{a,i}^2} = \Theta(1), \quad \forall a \in [2], i \in [d_a]$, and also that $\sigma^2 = \Theta(1)$ is fixed.

Suppose also $\{\lambda_{2,i}\}_i$ are whitened, but $\{\lambda_{1,i}\}_i$ are not, in that $\exists i \in [d_1]$, and $M > 0$ large satisfying $\frac{\lambda_{1,1}}{\lambda_{1,i}} = M > 0$ and $\lambda_{1,1} = \Theta(1)$.

If we try to learn f_a^* with kernel k_a via ridge regression with ridge parameter σ^2, and let $\hat{\mathcal{E}}_S^a$ denote the associated generalisation error.

Then:

1. *$\frac{\hat{\mathcal{E}}_S^1}{\hat{\mathcal{E}}_S^2} \xrightarrow{S \to \infty} C$, where $C > 0$ satisfies $\sum_{i=1}^{d_1} \frac{\mu_{1,i}^2}{\lambda_{1,i}^2} = C \sum_{i=1}^{d_2} \frac{\mu_{2,i}^2}{\lambda_{2,i}^2}$.*
2. *Moreover, if M is large enough, then $\exists S^* < \infty$ depending on σ^2 and λ_i s.t. $\frac{\hat{\mathcal{E}}_{S^*}^1}{\hat{\mathcal{E}}_{S^*}^2} < C$.*

Proof. For $a = 1, 2$, let us define

$$w_{a,i} = \frac{\mu_{a,i}}{\sqrt{\lambda_{a,i}}},$$

such that $f_a^*(\boldsymbol{x}) = \sum_{i=1}^{d_a} w_{a,i}\sqrt{\lambda_{a,i}}\psi_{a,i}(\boldsymbol{x})$ and let $\hat{\mathcal{E}}_{i,S}^a$ denote the generalisation error associated to mode i for learning function f_a^*, so that by orthogonality, we have:

$$\hat{\mathcal{E}}_S^a = \sum_i \hat{\mathcal{E}}_{i,S}^a. \tag{16}$$

Then, from Bordelon et al. (2020) (Proposition 3), we have the following approximation:

$$\hat{\mathcal{E}}_{i,S}^a = \frac{w_{a,i}^2}{\lambda_{a,i}}\left(\frac{1}{\lambda_{a,i}} + \frac{S}{\sigma^2 + t_a(S)}\right)^{-2}\left(1 - \frac{S\gamma_a(S)}{(\sigma^2 + t_a(S))^2}\right)^{-1} \tag{17}$$

where $t_a(S)$ is the solution to the implicit equation

$$t_a(S) = \sum_{i=1}^{d_a} \left(\frac{1}{\lambda_{a,i}} + \frac{S}{\sigma^2 + t_a(S)} \right)^{-1}, \tag{18}$$

and $\gamma_a(S)$ is defined by

$$\gamma_a(S) \triangleq \sum_{i=1}^{d_a} \left(\frac{1}{\lambda_{a,i}} + \frac{S}{\sigma^2 + t_a(S)} \right)^{-2}. \tag{19}$$

Note that, $0 \leq t_a(S) \leq \sum_{i=1}^{d_a} \lambda_{a,i} < \infty$, and likewise $0 \leq \gamma_a(S) \leq \sum_{i=1}^{d_a} \lambda_{a,i}^2 < \infty$. Moreover, we have $\gamma_a(S) \leq \Theta(S^{-2})$. Also note that $t_a(S) = o(1)$ as $S \to \infty$.

Then, as $S \to \infty$, we have from Eq. (17):

$$\hat{\mathcal{E}}^a_{i,S} = \frac{\sigma^4 w_{a,i}^2}{S^2 \lambda_{a,i}} (1 + o(1)). \tag{20}$$

So from Eq. (16), we see that

$$\frac{\hat{\mathcal{E}}^1_S}{\hat{\mathcal{E}}^2_S} \xrightarrow{S \to \infty} C \tag{21}$$

where C satisfies

$$\sum_{i=1}^{d_1} \frac{w_{1,i}^2}{\lambda_{1,i}} = C \sum_{i=1}^{d_2} \frac{w_{2,i}^2}{\lambda_{2,i}}. \tag{22}$$

This concludes the proof for 1).

Now we are given that $M\lambda_{1,i} = \lambda_{1,1} = \Theta(1)$ for large M.

Suppose that we have S^* satisfying:

$$1 \ll \frac{S^*}{\sigma^2 + t_a(S^*)} \ll M, \quad \forall a \in [2]. \tag{23}$$

We know that such an S^* exists as $\sigma^2 \leq \sigma^2 + t_a(S) \leq \sigma^2 + \sum_{i=1}^{d_a} \lambda_{a,i} < \infty, \forall S, a \in [2]$.

Then, we have:

$$\frac{\hat{\mathcal{E}}^1_{i,S^*}}{\hat{\mathcal{E}}^1_{1,S^*}} = \frac{w_{1,i}^2 \lambda_{1,1}}{w_{1,1}^2 \lambda_{1,i}} \left(\frac{1}{\lambda_{1,i}} + \frac{S^*}{\sigma^2 + t_1(S^*)} \right)^{-2} \left(\frac{1}{\lambda_{1,1}} + \frac{S^*}{\sigma^2 + t_1(S^*)} \right)^{2} \tag{24}$$

$$= \frac{w_{1,i}^2 \lambda_{1,1}}{w_{1,1}^2 \lambda_{1,i}} \left(\frac{\lambda_{1,i} S^*}{\sigma^2 + t_1(S^*)} \right)^{2} \left(1 + O(\frac{1}{S^*}) + O(\frac{S^*}{M}) \right), \tag{25}$$

and by construction, we have

$$\frac{\lambda_{1,i} S^*}{\sigma^2 + t_1(S^*)} \ll 1,$$

so we see that:

$$\frac{\hat{\mathcal{E}}^1_{i,S^*}}{\hat{\mathcal{E}}^1_{1,S^*}} < \frac{w^2_{1,i}\lambda_{1,1}}{w^2_{1,1}\lambda_{1,i}} \tag{26}$$

$$= \lim_{S\to\infty} \frac{\hat{\mathcal{E}}^1_{i,S}}{\hat{\mathcal{E}}^1_{1,S}} \quad \text{from Eq. (20)} \tag{27}$$

up to $\left(1 + O(\frac{1}{S^*}) + O(\frac{S^*}{M})\right)$ multiplicative error, as $\frac{w^2_{1,i}}{\lambda_{1,i}} = \frac{\mu^2_{1,i}}{\lambda^2_{1,i}} = \Theta(1) > 0$ by assumption.

Likewise, for $j \notin \{i, 1\}$, we know $\lambda_{1,j} \leq \lambda_{1,1}$ (as eigenvalues are sorted so $\lambda_{1,1}$ is the largest eigenvalue), so that from Eq. (24) (replacing i with j):

$$\frac{\hat{\mathcal{E}}^1_{j,S^*}}{\hat{\mathcal{E}}^1_{1,S^*}} \leq \frac{w^2_{1,j}\lambda_{1,1}}{w^2_{1,1}\lambda_{1,j}} = \lim_{S\to\infty} \frac{\hat{\mathcal{E}}^1_{j,S}}{\hat{\mathcal{E}}^1_{1,S}}. \tag{28}$$

On the other hand, for k_2, as $(\lambda_{2,i})_i$ are whitened, i.e. $\lambda_{2,i} = \Theta(1), \forall i$, we have:

$$\frac{\hat{\mathcal{E}}^2_{i,S^*}}{\hat{\mathcal{E}}^2_{1,S^*}} = \lim_{S\to\infty} \frac{\hat{\mathcal{E}}^2_{i,S}}{\hat{\mathcal{E}}^2_{1,S}} \tag{29}$$

up to $\left(1 + O(\frac{1}{S^*})\right)$ multiplicative error.

Finally, we have the following ratio for the dominant eigenmode errors between k_1 and k_2:

$$\frac{\hat{\mathcal{E}}^1_{1,S^*}}{\hat{\mathcal{E}}^2_{1,S^*}} = \frac{w^2_{1,1}\lambda_{2,1}}{w^2_{2,1}\lambda_{1,1}} \left(\frac{1}{\lambda_{1,1}} + \frac{S^*}{\sigma^2 + t_1(S^*)}\right)^{-2} \left(\frac{1}{\lambda_{2,1}} + \frac{S^*}{\sigma^2 + t_2(S^*)}\right)^{2} \times \tag{30}$$

$$\left(1 - \frac{S^*\gamma_1(S^*)}{(\sigma^2 + t_1(S^*))^2}\right)^{-1} \left(1 - \frac{S^*\gamma_2(S^*)}{(\sigma^2 + t_2(S^*))^2}\right) \tag{31}$$

$$\tag{32}$$

but note that by construction in Eq. (23), S^* satisfies $t_1(S^*), t_2(S^*) = o(\sigma^2)$ as we recall $\sigma^2 = \Theta(1)$. Moreover, as both $\lambda_{1,1}$ and $\lambda_{2,1}$ are $\Theta(1)$, we have $\big($up to $(1 + O(\frac{1}{S^*}) + O(\frac{t_1(S^*)+t_2(S^*)}{\sigma^2}))$ multiplicative error$\big)$:

$$\frac{\hat{\mathcal{E}}^1_{1,S^*}}{\hat{\mathcal{E}}^2_{1,S^*}} = \frac{w^2_{1,1}\lambda_{2,1}}{w^2_{2,1}\lambda_{1,1}} \tag{33}$$

$$= \lim_{S\to\infty} \frac{\hat{\mathcal{E}}^1_{1,S}}{\hat{\mathcal{E}}^2_{1,S}}. \tag{34}$$

Putting this all together, for large enough M & S^* satisfying Eq. (23) (such that all $(1 + o(1))$ multiplicative errors may be

ignored):

$$\frac{\hat{\mathcal{E}}^1_{S^*}}{\hat{\mathcal{E}}^2_{S^*}} = \frac{\sum_{m=1}^{d_1} \hat{\mathcal{E}}^1_{m,S^*}}{\sum_{j=1}^{d_2} \hat{\mathcal{E}}^2_{j,S^*}} \tag{35}$$

$$= \frac{\hat{\mathcal{E}}^1_{1,S^*}}{\hat{\mathcal{E}}^2_{1,S^*}} \frac{\sum_{m=1}^{d_1} \frac{\hat{\mathcal{E}}^1_{m,S^*}}{\hat{\mathcal{E}}^1_{1,S^*}}}{\sum_{j=1}^{d_2} \frac{\hat{\mathcal{E}}^2_{j,S^*}}{\hat{\mathcal{E}}^2_{1,S^*}}} \tag{36}$$

$$< \lim_{S\to\infty} \frac{\hat{\mathcal{E}}^1_{1,S}}{\hat{\mathcal{E}}^2_{1,S}} \frac{\sum_{m=1}^{d_1} \frac{\hat{\mathcal{E}}^1_{m,S}}{\hat{\mathcal{E}}^1_{1,S}}}{\sum_{j=1}^{d_2} \frac{\hat{\mathcal{E}}^2_{j,S}}{\hat{\mathcal{E}}^2_{1,S}}} \quad \text{by Eqs. (26), (28), (29) and (34)} \tag{37}$$

$$= \lim_{S\to\infty} \frac{\hat{\mathcal{E}}^1_{S}}{\hat{\mathcal{E}}^2_{S}} \tag{38}$$

$$= C \tag{39}$$

as required. □

B. Additional Experiments

STL-10 analysis Figure 7 is akin to Figure 4, but trained with Barlow Twins on STL-10 dataset. Training hyperparameters matched exactly the values in Figure 7, except slightly different data-augmentations (Color Jitter & Gaussian Blur) were used for SSL pretraining, matching the default values of the codebase in Footnote 2. We observe similar trends in Figure 7 to Figure 4 where deeper projections have more collapsed encoder representations, and also test accuracy is not monotonic in the degree of whitening.

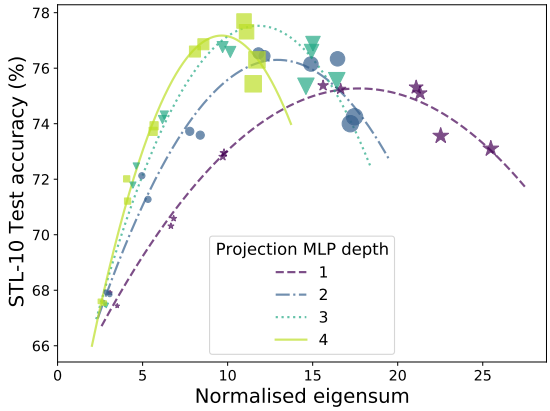


Figure 7. Akin to Figure 4 but using STL-10 as unlabelled dataset for Barlow Twins training as opposed to CIFAR-10.

Gram matrices across methods In Figure 8 we plot feature Gram correlation matrices, with $(i,j)^{\text{th}}$ entry

$$\frac{\langle h_{\boldsymbol{\theta}}(\boldsymbol{x}_i), h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\rangle}{\|h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2 \|h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\|_2}$$

over 1000 CIFAR-10 test points, for the 3 pretrained ResNet-18 (either with SimCLR, Barlow Twins, or Supervised) displayed in Figure 6. We see that the NN trained with supervision has feature Gram matrix that is much closer qualitatively to the classwise Gram matrix in $\mathbb{R}^{1000\times 1000}$ (which takes $(i,j)^{\text{th}}$ value 1 if input i and input j are from the same class, and 0

else). As the classwise Gram matrix has exactly C non-zero eigenvalues for C classes, this provides evidence that the feature collapse in Figure 6 (bottom left) is due to the fact that the 10 dominant eigenvalues correspond to the 10 different classes in CIFAR-10. This is consistent with the recently observed Neural Collapse phenomenon (Papyan et al., 2020). On the other hand, without training labels, SSL methods do not have as obvious a correspondence between dominant eigenvalues and classes.

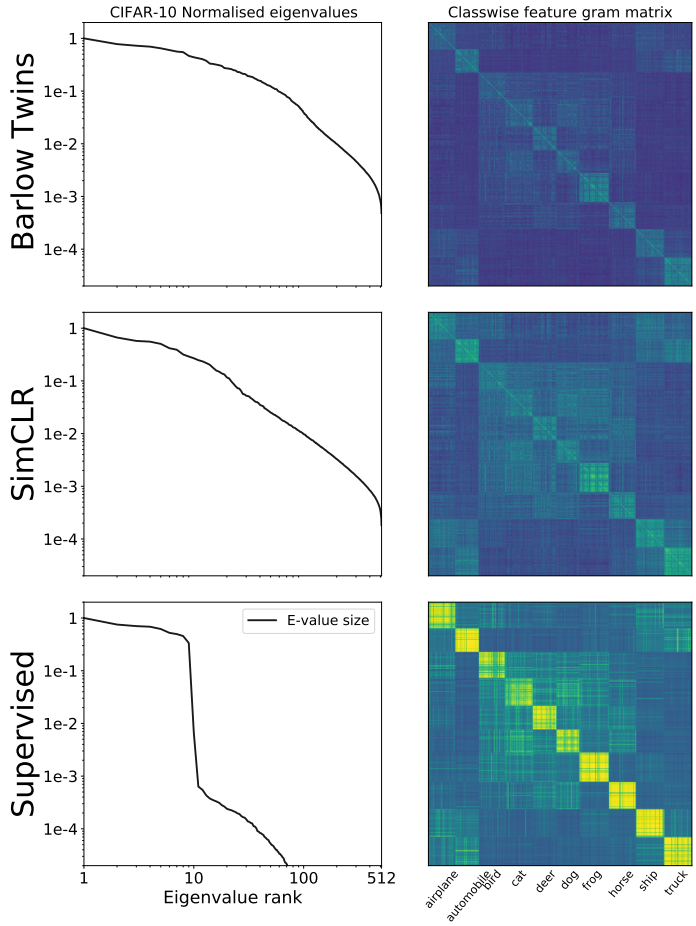


Figure 8. Feature eigenspectra and Gram matrices corresponding to Figure 6, over 1000 CIFAR-10 test examples.

ImageNet-1K with large labelled-data. In Figure 9 (right), we plot the performance of PMP with Barlow Twins pretrained ResNet-50 encoder when given access to all 1.2 million training labels for evaluation. We see that PMP is able to match the performance of standard linear probe of 73.5% at values of $\beta = 0.8$. This is unsurprising given Figure 9 (left), which shows that the encoder eigenspectra (after rank 10) already approximately decays with exponent 0.9. Weight decay 0.0001 is used. It would be interesting to see if one can improve SSL in large labelled data regimes with more optimal tuning of e.g. weight decay, but for this it would also be desirable to first design more efficient methods of PMP hyperparameter tuning.

Interactions between weight decay and β for low labelled data. In Figure 10, we plot the accuracy of PMP with SimCLR, SwAV and Barlow Twins pretrained ResNet-50 encoders with 1% training labels, for different values of weight decay. For smaller values of weight decay, we see that larger β, corresponding to more collapsed features, yield higher top-1 accuracy. This is consistent with the findings of Corollary 4.3, Theorem A.2, and Figure 6, where suppressing the smaller eigenvalues is useful in low labelled data regimes. However, we also find that for larger values of weight decay, this trend is reversed, in that the best values of β are small, hence more whitened eigenspectra perform better. Indeed, the

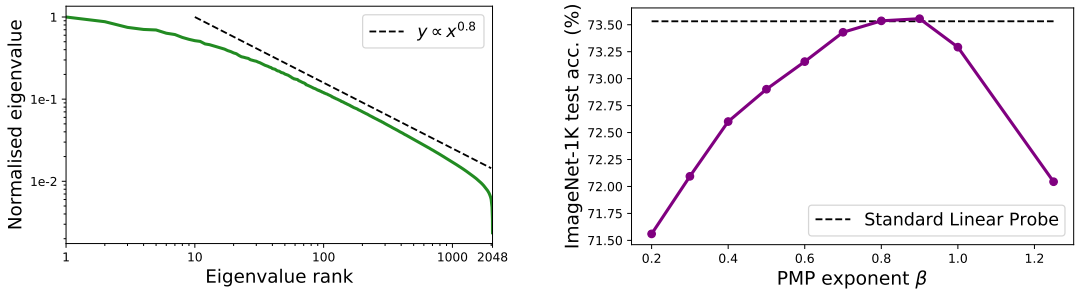


Figure 9. Barlow Twins eigenspectra plot (left) & ImageNet-1k validation accuracy using *all* 1.2 million training labels for PMP as a function of β (right). We see that the best values of β match standard linear probe at the value of $\beta = 0.8$ which matches the rate of decay of the original encoder. The encoder was pretrained with Barlow Twins and taken from the official implementation of Zbontar et al. (2021). Weight decay 0.0001 was used at evaluation time.

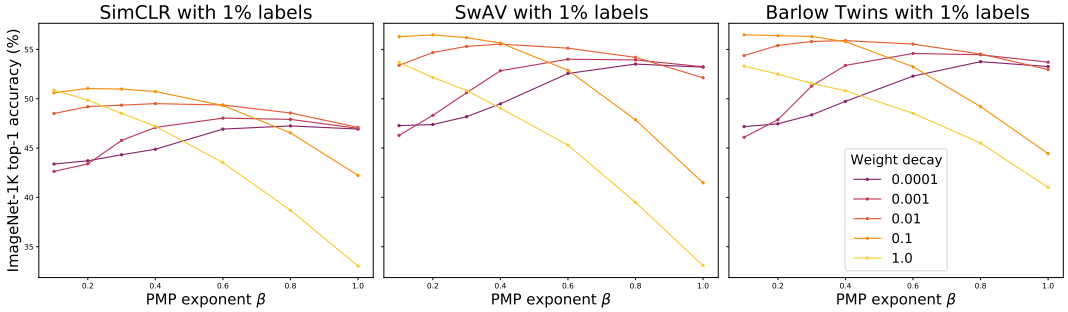


Figure 10. Accuracy in low labelled data setting as a function of PMP power law exponent β for different weight decay settings.

best performing hyperparameter setting chose a relatively large weight decay combined with small β (although we see that too large weight decay also hurts accuracy, particularly for better performing encoders: SwAV and Barlow Twins). This surprising observation is not covered by our theoretical results (which concern fixed weight decay), but does emphasise the importance of where an encoder lies along the gap between collapsed & whitened features, and its decay rate of eigenspectra, in determining its generalisation performance, particularly in low-labelled data settings.

B.1. ImageNetV2 results for PMP

To verify that our results in Table 1 are not overfit to the ImageNet-1K validation set, in Table 4 we provide corresponding results for evaluating PMP in low-labelled settings (from the ImageNet-1K training set) on the ImageNetV2 test datasets. We observe the same trends in Table 4 as in Table 1: PMP always outperforms LP; PMP often outperforms NFT; and LP sometimes outperforms NFT on SwAV.

C. Experimental Details

C.1. Figure 3

Our SimCLR implementation was taken from an open-source codebase[2] & we used default hyperparameters provided like 0.5 temperature for InfoNCE; our Barlow Twins implementation used Alg. 1 of Zbontar et al. (2021).

Both Barlow Twins & SimCLR ResNet-18 encoders used projectors of depth 2, with ReLU & BatchNorm. Barlow Twins used wide projector width 1024 & batch size 256 whereas SimCLR used smaller width 256 & larger batch size 512, which are all standard hyperparameter choices. All networks were trained with SGD loss for 100 epochs with weight decay 0.0004, momentum 0.9 & a cosine annealed learning rate.

[2]https://github.com/facebookresearch/luckmatters

Table 4. **ImageNetV2 test evaluation**: PMP top-1 % accuracy vs. LP & non-linear finetuning (NFT) on the 3 ImageNetv2 test sets: Matched Frequency (MF), Threshold0.7 (T-0.7) & TopImages (Top-I), across low-label settings & SSL methods.

METHOD	LABELS	EVAL	MF	T-0.7	TOP-I
BARLOW	1%	LP	44.18	52.65	58.85
		PMP	**45.60**	**54.11**	**60.03**
		NFT	45.00	53.39	59.06
	10%	LP	51.62	60.40	66.32
		PMP	55.36	64.49	70.65
		NFT	**58.57**	**67.46**	**73.15**
SIMCLR	1%	LP	37.78	45.73	51.58
		PMP	40.89	49.06	54.62
		NFT	**41.76**	**49.80**	**55.63**
	10%	LP	48.62	57.77	64.15
		PMP	49.77	59.26	65.72
		NFT	**54.80**	**64.00**	**69.70**
SWAV	1%	LP	42.46	50.68	57.15
		PMP	**44.24**	**52.44**	**58.63**
		NFT	42.74	50.21	56.70
	10%	LP	55.68	64.76	70.88
		PMP	56.44	65.29	71.54
		NFT	**58.76**	**68.13**	**73.92**

Factors such as learning rate and regularisation strength were chosen to ensure all networks achieved similar test accuracy under linear probe (≈85%). Learning rate was 0.32 for SimCLR & 0.25 for Barlow Twins, with $\rho = 0.01$. All methods used the default data-augmentations on CIFAR-10 as described in Chen et al. (2020a).

C.2. Figure 4

All training details follow Figure 3 above, though we add that the values for ρ used were $\{0.001, 0.003, 0.01, 0.03, 0.05\}$.

C.3. Figure 6

For all PMP experiments, we start eigenvalue decay power law after the tenth largest eigenvalue in PMP, Alg. 1. This is consistent with Nassar et al. (2020), who studied importance of power-law decay in eigenspectra for adversarial robustness, & the findings of Stringer et al. (2019). For the subsets of labelled data (including for 0.3% ImageNet-1K too), we sample uniformly at random from the CIFAR-10 train set, ensuring that all classes have an equal number of examples (so that 0.1% labelled-data corresponds to 5 examples per class). In all cases at evaluation time, for linear probe or PMP, we trained W_C using SGD for 50 epochs with batch size 128, momentum 0.9, weight decay 0.001, learning rate 0.1 and cross-entropy loss.

The supervised NN was trained for 160 epochs using SGD+momentum with learning rate 0.05 and batch size 128. Weight decay for supervised training was set to 0.0003 to ensure the NN also achieved similar CIFAR-10 test accuracy (85%). To that end, standard data augmentation (random crops & flips) was not used to train the supervised NN, and the only preprocessing of images was normalising before training.

C.4. ImageNet-1K evaluation: Table 1

Our ImageNet-1K implementation was based off the official Barlow Twins (Zbontar et al., 2021) implemtation[3], which is also where we obtained the ResNet-50 Barlow Twin checkpoint pretrained on ImageNet-1K. The SimCLR (Chen et al., 2020a) & SwAV (Caron et al., 2020) ResNet-50 checkpoints were obtained from the VISSL library's (Goyal et al., 2021) model zoo. In particular, we selected the SimCLR checkpoint that was trained for 800 epochs, and the SwAV checkpoint that was trained for 800 epochs with multi-crop setting: $2 \times 224 + 6 \times 96$. These selections were based on the best top-1 accuracy performing checkpoints (under linear probe).

[3]https://github.com/facebookresearch/barlowtwins

The top-1 accuracies under linear probe in Table 1 have slight discrepancies to those reported in VISSL ($< 0.4\%$ difference), possibly reflecting slight differences in linear evaluation training schemes e.g. we use weight decay 0.0001 & normalise each of the 2048 features to have zero mean and unit variance across the unlabelled ImageNet-1K dataset. Indeed, we normalised neurons across all linear evaluation schemes & all labelled-data settings, in order to avoid the setting where non-zero means result in a single eigenvalue that is orders of magnitude larger than others.

In all linear evaluation schemes: standard linear probe (LP); 2-layer linear MLP classifier (MLP); & our PostMan-Pat (PMP), we train the classifier for 100 epochs using SGD & momentum 0.9, with a cosine annealed learning rate starting at 0.1, with weight decay tuned in all cases (along with power law exponent β for PMP). For the linear MLP classifier we used single hidden layer of width 4096.

For non-linear finetuning (NFT), we tuned followed the same training procedure, apart from additionally tuning the number of training epochs (between 20 & 40, which is consistent with Zbontar et al. (2021)), as well as separate encoder & classifier learning rates. In NFT, we did not use weight decay for the encoder, as this would remove the useful features learning in the encoder during pretraining. However, we did tune weight decay for the linear classifier W_C.

Hyperparameter tuning data splits For any given set of labelled data, we split the data into 4:1 splits for the 1% or 10% labelled-data setting, or 2:1 splits for the 0.3% labelled-data setting (as in the 0.3% setting we have only 3 labels per class). Splits were chosen uniformly at random so that each class had an equal number of examples in the larger split, which was then used for training. Top-1 accuracy on the smaller split was used for hyperparameter tuning.

C.5. Dataset Transfer: Table 2

We found it important to recalculate the PMP rescaling matrix W_{PMP} on unlabeled data from the new dataset, and all hyperparameters were tuned on a 4:1 split of the training data. For Oxford Flowers, as there are only 1020 training images, which would result in a low-rank approximation to $W_{\text{PMP}} \in \mathbb{R}^{2048\times 2048}$, for each training image we generate 10 data augmented versions (using standard random crop and horizontal flips) to estimate the empirical covariance used in W_{PMP}. All datasets were obtained from the Torchvision PyTorch library (Paszke et al., 2019)

C.6. Different Architecture evaluation: Table 3

All experimental details follow those in Appendix C.4, and the pretrained ViT-B/16 checkpoint was again obtained from VISSL (Goyal et al., 2021).

D. Postman Pat

The Postman Pat abbreviation (which we further shorten to PMP) for our method, Post-hoc Manipulation of the Principal Axes & Trace, was inspired by the now retired British children's TV character: Postman Pat, pictured in Figure 11 with his cat Jess (though in the Danish version Jess is renamed to Emil).

Figure 11. Postman Pat and his cat Jess.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Exploring the Gap between Collapsed & Whitened Features in Self-Supervised Learning
Publication Status	Published
Publication Details	Bobby He and Mete Ozay. Exploring the Gap between Collapsed & Whitened Features in Self-Supervised Learning. In Proceedings of the 39th International Conference on Machine Learning, 2022.

Student Confirmation

Student Name:	Bobby He		
Contribution to the Paper	- Lead author with one advisor. - Identified problem tackled in paper. - Derived all theoretical results and proofs, which were checked by advisor. - Performed all experiments. - Wrote the paper, with helpful feedback from advisor.		
Signature 		Date	8/1/23

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof Yee Whye Teh		
Supervisor comments Bobby is lead author and did all the research work in this paper.		
Signature 	Date	8 Jan 2023

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 6

Conclusion and Discussion

This thesis treats the dichotomy and similarities between kernel learning and feature learning as a lens through which to view neural networks (NNs). In this lens, we place an increased focus on studying the *inner products of feature representations*, $\langle h(\boldsymbol{x}), h(\boldsymbol{x}') \rangle$, formed by an NN, which is motivated through theoretical insights from the training dynamics of wide NNs. In particular, we posit that we can understand the behaviour of NNs through understanding such inner products, both at initialisation and throughout training. This forms an alternative to more traditional ways to reason about NNs, which include understanding the finite width and depth expressivity of different architectures through circuit theory (e.g. Le Roux and Bengio (2008); Martens et al. (2013), or studying networks on an individual neuron level (Olah et al., 2017). This "kernel vs feature learning" perspective has both strengths and weaknesses: it eschews permutation symmetric redundancies that exist in parameter and feature spaces (Entezari et al., 2021), but, unlike parameter spaces, also requires some particular choice of inputs $\boldsymbol{x}$ to produce features in the first place.

From this new lens, it is possible to view a trained NN as a (potentially data-dependent) kernel k, which determines a notion of "similarity" between different inputs.[1] Whether or not this kernel is data-dependent is determined by whether feature or kernel learning occured during training; valid training regimes exist for both. As discussed, feature learning is strongly believed to be crucial to explaining the success of DNNs, but kernel learning is perhaps better studied and has certain tractability advantages, in existing literature.

The papers posing as chapters in this thesis can be thought of as attempts to demonstrate why this "kernel vs feature learning" lens is an interesting and useful way to understand NNs, as we now discuss:

- In Chapter 3, we utilise the tractability of the kernel regime to precisely characterise the relationship between deep ensembles and Bayesian inference, and identify missing variance in the NN function at initialisation as the reason why there isn't a Bayesian interpretation for wide NNs trained in the kernel regime. Fundamentally, this missing variance arises due to the difference between the NNGP kernel, which describes a wide NN at initialisation, and the NTK kernel, which describes

[1] For a trained NN (of any width), we can always simply retrain its last layer, and for common losses like cross-entropy or least squares, the ensuing predictions will be determined entirely by its feature kernel, c.f. Appendix A in Chapter 4.

its training dynamics. We propose to correct for this missing variance with an additive random GP function $\delta(\cdot)$ to our NN function that remains untrainable throughout training, and provide a random features approximation to $\delta(\cdot)$ that enables computation with finite-width NNs. In doing so, our methods offer a practical way to build and train large-scale NNs with a Bayesian interpretation, and when ensembled together, our modified NNs can be viewed as approximating a Bayesian posterior, which we call the NTKGP posterior.

- In Chapter 4, we view the "knowledge" of a trained NN as being contained in its *feature kernel*, $k(\boldsymbol{x}, \boldsymbol{x}') = \langle h_L(\boldsymbol{x}), h_L(\boldsymbol{x}') \rangle$, induced from last layer representations h_L, and treat the feature kernel as the primary object for knowledge distillation. One key benefit in doing so is the fact that the feature kernel is an intrinsic property of a model, that exists independent of dataset or architecture. We demonstrate that an NN's trained feature kernel is highly dependent on its parameter initialisation, in the sense that different initialisations bias a trained NN to learn different features. This allows us to theoretically show that targeting the teacher's feature kernel is a principled method for self-distillation and (ensemble) distillation, which we do by extending the multi-view data setup of Allen-Zhu and Li (2020). Our theory further allows us to derive practical considerations, like using the correlation kernel and feature regularisation, for improving the generalisation performance of the student using feature kernels, which we demonstrate empirically across a range of distillation settings. We note that Chapter 4 is perhaps the most closely aligned chapter with the overall theme of this thesis in terms of combining kernel and feature learning in NNs.

- Finally, in Chapter 5, we explore the gap between collapsed and whitened features that are learnt in self-supervised learning (SSL). We view collapsed and whitened features as two extremes of a spectrum of possible learnt features, and identify power law behaviour in the feature kernel's eigenvalues as a spectrum, parameterised by power law exponent $\beta \geq 0$, that bridges between these two extremes. By considering the rate of decay of feature eigenvalues, our work stands in contrast to existing work in the SSL literature, which tends to view each eigenmode of the feature kernel as independent binary outcomes: collapsed or uncollapsed. Our insights highlight different hyperparameters, like projection depth and regularisation strength, that affect the degree of feature whitening, and moreover the importance of the decay rate of feature eigenvalues in determining downstream evaluation performance, particularly in the context of scarce labelled data.

Through the different chapters in this thesis, we see that there is scope for our perspective to be potentially relevant across a wide range of areas related to NNs. Given this generality, we believe reasoning about NNs in terms of kernel and feature learning is a fundamental and insightful way to think about DL systems.

6.1 Limitations

Having said that, we stress that the ideas we have covered are not without limitation. First and foremost, the analytic nature of tracking NN training dynamics using inner products like the feature kernel are only approximate outside of the infinite-width limit. Moreover, tracking feature learning lim-

its for general architectures quickly becomes computationally infeasible (Yang and Hu, 2020; Bordelon and Pehlevan, 2022), whilst the kernel regime has cubic complexity in dataset size, which is already unfavourable relative to gradient descent with finite-width NNs. As a result, for all the analytic tractability that kernel and feature learning regimes provide in infinite-width NNs, there is still a gap in our understanding to state-of-the-art finite-width DNN architectures.

In addition, Chapter 3 is restricted to the kernel regime, and is thus incompatible with feature learning during training. In fact, a Bayesian interpretation for wide NNs that exhibit feature learning is somewhat complicated, because a necessary implication of such feature learning regimes is that the output function is identically zero at initialisation (Mei et al., 2018; Chizat and Bach, 2018; Sirignano and Spiliopoulos, 2018; Rotskoff and Vanden-Eijnden, 2018; Yang and Hu, 2020; Bordelon and Pehlevan, 2022), which cannot be viewed as encoding some prior belief. This stands in contrast to the NNGP output function limit (Corollary 2.6), which has often been interpreted as a functional prior for wide Bayesian NNs (Neal, 1996; Matthews et al., 2018; Lee et al., 2018a); this discrepancy arises precisely due to the output layer downscaling we described in Section 2.3.2 that is required for feature learning. Schut et al. (2021) highlight the degradation in uncertainty quantification in wide deep ensembles that exhibit feature learning as a result, and propose a modification to gradient descent to correct for this degradation.

Outside of the kernel regime, we resort to different simplifications in order to gain tractability. In Chapter 4 we are restricted to a single hidden-layer nonlinear CNN in our theory for feature kernel distillation, and in Chapter 5 we consider the feature kernel of deep linear networks that exhibit alignment across layers when trained with SSL. Though these assumptions are commonplace in other works in the literature, they reflect an unsatisfying gap between settings where we can interpret training behaviours mathematically, and settings that we use in practice.

Moreover, in Chapters 4 and 5, by focusing on the feature kernel (i.e. inner product of last layer representations), we are essentially abstracting away the architecture of the backbone encoder h_L. This has the obvious benefit that our findings extend across architectures, but at the same time is also a limitation as it restricts what we can say about different choices in architecture, which is clearly undesirable. The exception is our consideration of the impact of the depth hyperparameter L in Chapter 5, which is possible due to the fact that we consider deep linear MLPs.

Chapter 4 highlights another consideration that is often overlooked in the literature: properties of the data we use in deep learning. Indeed, the multi-view data construction (Allen-Zhu and Li, 2020) is crucial to prove that the different ensemble members learn different features, which is reflected in the model's trained feature kernel in our case. Such a setting is motivated through intuition about image data, but is likely a crude simplification of true behaviour in real-world data. At the same time, empirical results in Allen-Zhu and Li (2020) suggest that training on datasets with different properties, e.g. Gaussian-like inputs, will lead to different NN behaviours e.g. when ensembling. Further investigation into properties of the data used, and how they interact with the training algorithms and models in DL, is an important direction for future work, which mirrors the sentiment of Xiao and Pennington (2022).

This links nicely into a further limitation, this time for our work in Chapter 5, where we focus

only on the relative weighting of eigenmodes in learnt features and its relevance for downstream tasks. However, it is arguably more important (and difficult) to study *what* those learnt eigenmodes correspond to, which holds signficance not only for the downstream generalisation of our models, but also related questions like interpretability and robustness. Again, answering this question will likely depend on both the dataset and models involved, but also in the case of Chapter 5, the data augmentations used in the joint-embedding framework for SSL.

6.2 Outlook

We began this thesis with a discussion on the theory of DNNs, and its importance for our understanding of the DL systems that are used in practice. Such has been the empirical success of deep learning, that we note the purpose of theory in DL is actually something that is debated in the community. Detractors argue that deep learning has already come so far without a grounded mathematical theory, and instead practical intuitions are sufficient for progress. Moreover, they contest that much of the theoretical work in DL is irrelevant in practice, relying on unrealistic assumptions that either plainly do not hold, or are contrived in order to explain some specific observed phenomenon in a specific scenario, thereby restricting the generality of new insights.

While we are generally in favour of theory in deep learning, we sympathise with some of these views. Ultimately, we believe that useful theory should be intertwined with practice, with both aiding the development of the other. At its core, the fundamental question that we are interested in is improving our *understanding* of the general behaviours of real-world NNs, and using this understanding to improve, in some sense, the NNs we train in practice. We believe that both theory and empirical practice have a key part to play in answering this question, and ideally should interact and evolve together in order to do so. Oftentimes in DL, we see empirical observations driving theoretical research, but the value of theory is arguably stronger in the opposite, less prevalent, direction.[2]

The theory of wide NNs, and of kernel and feature learning, is among the most successful and practically impactful mathematical frameworks for DL. Indeed, perhaps the area of NN training we understand best mathematically is initialisation, where, as discussed in this thesis, connections to Gaussian Processes have been known dating back to the 1990s (Neal, 1996). More recently, initialisation time infinite-width signal propagation considerations have lead to significant advances in our understanding of different NN architectures tools, like skip connections and normalisation layers, and our ability to train deep NNs with or without such tools (Xiao et al., 2018; Hayou et al., 2021a; Martens et al., 2021; Zhang et al., 2022). Despite this, we note that recent work has suggested even our understanding of initialisation is incomplete, and more nuanced behaviour is possible depending on the choice of wide-and-depth limit (Li et al., 2022; Hayou, 2022).

[2]As an aside, we note that one pragmatic alternative to this is so-called "empirical theory" (Nakkiran, 2021), which starts by observing real world practice, and aims to simplify aspects of it into forms that we can quantitatively describe (albeit not necessarily using fully rigorous mathematical justification). A nice example for DNNs is showing the linear relationship between batch size and optimal training speed (Shallue et al., 2019; Zhang et al., 2019b). One potential issue with empirical theory is that eventually its predictions may break down, and without a mathematical framework it becomes unclear when this may be.

Beyond initialisation, there are still many holes in our understanding surrounding the optimisation of finite-width NNs in practice, not to mention generalisation. As discussed, both kernel and feature learning infinite-width regimes have their own limitations in describing NN training dynamics: kernel regimes are incompatible with feature learning, and feature learning regimes are significantly more computationally expensive and complex to analyse. Moreover, a multitude of open questions exist, exposing our incomplete understanding of NN optimisation. These include the advantages of skip connections for training speed (Martens et al., 2021; Zhang et al., 2022) or the reasons why different optimisers are preferred for different architectures (e.g. SGD with momentum for ResNets and adaptive optimisers for transformers). Other exciting, but not yet fully understood, observations include the edge of stability (Cohen et al., 2020) and neural collapse (Papyan et al., 2020).

In this thesis, we have taken a kernel and feature learning view of neural networks in order to study a few different areas of DNNs, including ensembling, knowledge distillation, and self-supervised learning. While we cannot guarantee that such an approach will always be fruitful in the future, one thing we can be sure of is that there won't be any shortage of interesting questions to investigate.

Bibliography

Taiga Abe, E. Kelly Buchanan, Geoff Pleiss, Richard Zemel, and John Patrick Cunningham. Deep ensembles work, but are they necessary? In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=Wl1ZIgMqLlq`.

Ben Adlam and Jeffrey Pennington. Understanding double descent requires a fine-grained bias-variance decomposition. *Advances in neural information processing systems*, 33:11022–11032, 2020.

Ben Adlam, Jaehoon Lee, Lechao Xiao, Jeffrey Pennington, and Jasper Snoek. Exploring the uncertainty properties of neural networks' implicit priors in the infinite-width limit. In *International Conference on Learning Representations*, 2020.

Laurence Aitchison. Why bigger is not always better: on finite and infinite neural networks. In *International Conference on Machine Learning*, pages 156–164. PMLR, 2020.

Laurence Aitchison, Adam Yang, and Sebastian W Ober. Deep kernel processes. In *International Conference on Machine Learning*, pages 130–140. PMLR, 2021.

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

Sotiris Anagnostidis, Luca Biggio, Lorenzo Noci, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=FxVH7iToXS`.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

Dyego Araújo, Roberto I Oliveira, and Daniel Yukimura. A mean-field limit for certain deep neural networks. *arXiv preprint arXiv:1906.00193*, 2019.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019a.

Sanjeev Arora, Simon S Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2019b.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. In *36th International Conference on Machine Learning, ICML 2019*, pages 9904–9923. International Machine Learning Society (IMLS), 2019c.

Devansh Arpit, Víctor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Pranjal Awasthi, Nishanth Dikkala, and Pritish Kamath. Do more negative samples necessarily hurt in contrastive learning? In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1101–1116. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/awasthi22b.html`.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pages 1352–1361. PMLR, 2021.

David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *International Conference on Machine Learning*, pages 342–350. PMLR, 2017.

Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *arXiv preprint arXiv:2205.11508*, 2022.

Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2269–2277. PMLR, 13–15 Apr 2021. URL `https://proceedings.mlr.press/v130/baratin21a.html`.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18: 1–43, 2018.

Suzanna Becker and Geoffrey E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10925–10934, 2022.

Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *Advances in Neural Information Processing Systems*, 32, 2019.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. *arXiv preprint arXiv:2205.09653*, 2022.

Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, page 737–744, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sebastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=z71OSKqTFh7`.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. *arXiv preprint arXiv:1912.01198*, 2019.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020a. URL `https://proceedings.mlr.press/v119/chen20s.html`.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper/2009/file/5751ec3e9a4feab575962e78e006250d-Paper.pdf`.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005. doi: 10.1109/CVPR.2005.202.

Kamil Ciosek, Vincent Fortuin, Ryota Tomioka, Katja Hofmann, and Richard Turner. Conservative Uncertainty Estimation By Fitting Prior Networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJlahxHYDS`.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2020.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(76): 2493–2537, 2011. URL `http://jmlr.org/papers/v12/collobert11a.html`.

Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

Amit Daniely. Sgd learns the conjugate kernel class of the network. *Advances in Neural Information Processing Systems*, 30, 2017.

Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

Soham De and Sam Smith. Batch normalization biases residual blocks towards the identity function in deep networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19964–19975. Curran Associates, Inc., 2020.

Valentin De Bortoli, Alain Durmus, Xavier Fontaine, and Umut Simsekli. Quantitative propagation of chaos for sgd in wide neural networks. *Advances in Neural Information Processing Systems*, 33: 278–288, 2020.

Bruno de Finetti. Sul significato soggettivo della probabilità. *Fundamenta Mathematicae*, 17(1):298–329, 1931. URL `http://eudml.org/doc/212523`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Francesco D' Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3451–3465. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/1c63926ebcabda26b5cdb31b5cc91efb-Paper.pdf`.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

Yann Dubois, Stefano Ermon, Tatsunori Hashimoto, and Percy Liang. Improving self-supervised learning by characterizing idealized representations. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=agQGDz6gPOo`.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL `http://jmlr.org/papers/v12/duchi11a.html`.

Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. In *International Conference on Learning Representations*, 2019.

Gintare Karolina Dziugaite, Kyle Hsu, Waseem Gharbieh, Gabriel Arpino, and Daniel M Roy. On the role of data in PAC-Bayes bounds. In *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2021.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2021.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pages 3015–3024. PMLR, 2021.

Cong Fang, Jason Lee, Pengkun Yang, and Tong Zhang. Modeling from features: a mean-field framework for over-parameterized deep neural networks. In *Conference on learning theory*, pages 1887–1936. PMLR, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2018.

Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bklfsi0cKm`.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *Advances in Neural Information Processing Systems*, 33:14820–14830, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `https://proceedings.mlr.press/v9/glorot10a.html`.

Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004. URL `https://proceedings.neurips.cc/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf`.

Eugene Golikov and Greg Yang. Non-gaussian tensor programs. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=AcHUIG2wA8-`.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=Fp7__phQszn`.

Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 318–319, 2020.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006. doi: 10.1109/CVPR.2006.100.

Manu Srinath Halvagal, Axel Laborieux, and Friedemann Zenke. Predictor networks and stop-grads provide implicit variance regularization in byol/simsiam. *arXiv preprint arXiv:2212.04858*, 2022.

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. In *International Conference on Learning Representations*, 2019.

Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 569–579, Red Hook, NY, USA, 2018. Curran Associates Inc.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

Soufiane Hayou. On the infinite-depth limit of finite-width neural networks. *arXiv preprint arXiv:2210.00688*, 2022.

Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. In *International conference on machine learning*, pages 2672–2680. PMLR, 2019.

Soufiane Hayou, Eugenio Clerico, Bobby He, George Deligiannidis, Arnaud Doucet, and Judith Rousseau. Stable resnet. In *International Conference on Artificial Intelligence and Statistics*, pages 1324–1332. PMLR, 2021a.

Soufiane Hayou, Bobby He, and Gintare Karolina Dziugaite. Probabilistic fine-tuning of pruning masks and pac-bayes self-bounded learning. *arXiv preprint arXiv:2110.11804*, 2021b.

Bobby He and Mete Ozay. Feature kernel distillation. In *International Conference on Learning Representations*, 2021.

Bobby He and Mete Ozay. Exploring the gap between collapsed & whitened features in self-supervised learning. In *International Conference on Machine Learning*, pages 8613–8634. PMLR, 2022.

Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian deep ensembles via the neural tangent kernel. *Advances in neural information processing systems*, 33:1010–1022, 2020a.

Bobby He, Sheheryar Zaidi, Bryn Elesedy, Michael Hutchinson, Andrei Paleyes, Guy Harling, Anne Johnson, and Yee Whye Teh. Technical Document 3: Effectiveness and Resource Requirements of Test, Trace and Isolate Strategies. 2020b.

Bobby He, James Martens, Guodong Zhang, Aleksandar Botev, Andrew Brock, Samuel L. Smith, and Yee Whye Teh. Deep transformers without shortcuts: Modifying self-attention for faithful signal propagation. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=NPrsUQgMjKK`. under review.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016b.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020c.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Yehuda Hoffman and Erez Ribak. Constrained Realizations of Gaussian Fields: A Simple Algorithm. *The Astrophysical Journal*, 380:L5–L8, 1991.

Jiri Hron, Yasaman Bahri, Roman Novak, Jeffrey Pennington, and Jascha Sohl-Dickstein. Exact posterior distributions of wide bayesian neural networks. *arXiv preprint arXiv:2006.10541*, 2020a.

Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning*, pages 4376–4386. PMLR, 2020b.

Tianyu Hua, Wenxiao Wang, Zihui Xue, Sucheng Ren, Yue Wang, and Hang Zhao. On feature decorrelation in self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9598–9608, 2021.

Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. In *International conference on machine learning*, pages 4542–4551. PMLR, 2020.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4475–4483. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/huang20f.html`.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Guangda Ji and Zhanxing Zhu. Knowledge distillation in wide neural networks: Risk bound, data efficiency and imperfect teacher. *Advances in Neural Information Processing Systems*, 33:20823–20833, 2020.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations*, 2021.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: Mean field approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1032–1041. PMLR, 2019.

George S Kimeldorf and Grace Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL `https://proceedings.neurips.cc/paper/1994/file/b8c37e33defde51cf91e1e03e51657da-Paper.pdf`.

Pei Ling Lai and Colin Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(05):365–377, 2000.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural computation*, 20(6):1631–1649, 2008.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep Neural Networks as Gaussian Processes. In *International Conference on Learning Representations*, 2018a.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. SNIP: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018b.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Mufan Li, Mihai Nica, and Dan Roy. The future is log-gaussian: Resnets and their infinite-depth-and-width limit at initialization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7852–7864. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/412758d043dd247bddea07c7ec558c31-Paper.pdf`.

Mufan Bill Li, Mihai Nica, and Daniel M. Roy. The neural covariance SDE: Shaped infinite depth-and-width networks at initialization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=WG3vmsteqR_`.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in neural information processing systems*, 31, 2018.

Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. Nvidia tesla: A unified graphics and computing architecture. *IEEE micro*, 28(2):39–55, 2008.

Ekaterina Lobacheva, Nadezhda Chirkova, Maxim Kodryan, and Dmitry P Vetrov. On power laws in deep ensembles. *Advances In Neural Information Processing Systems*, 33:2375–2385, 2020.

Yizhang Lou, Chris E Mingard, and Soufiane Hayou. Feature learning and signal propagation in deep neural networks. In *International Conference on Machine Learning*, pages 14248–14282. PMLR, 2022.

David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

DJC MacKay. Introduction to gaussian process. *Neural Networks and Machine Learning*, 1998.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

James Martens, Arkadev Chattopadhya, Toni Pitassi, and Richard Zemel. On the representational efficiency of restricted boltzmann machines. *Advances in Neural Information Processing Systems*, 26, 2013.

James Martens, Andy Ballard, Guillaume Desjardins, Grzegorz Swirszcz, Valentin Dalibard, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *arXiv preprint arXiv:2110.01765*, 2021.

Alexander G de G Matthews, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Sample-then-optimize posterior sampling for Bayesian linear models. In *NeurIPS Workshop on Advances in Approximate Bayesian Inference*, 2017.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian Process Behaviour in Wide Deep Neural Networks. In *International Conference on Learning Representations*, volume 4, 2018.

Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre GR Day, Clint Richardson, Charles K Fisher, and David J Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics reports*, 810:1–124, 2019.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Hossein Mobahi, Mehrdad Farajtabar, and Peter Bartlett. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 33:3351–3361, 2020.

Preetum Nakkiran. *Towards an Empirical Theory of Deep Learning*. PhD thesis, Harvard University, 2021.

Radford M Neal. Priors for Infinite Networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination Press, 2015.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1g30j0qF7`.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL `https://github.com/google/neural-tangents`.

Kento Nozawa and Issei Sato. Understanding negative samples in instance discriminative self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34:5784–5797, 2021.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

Luis A Ortega, Rafael Cabañas, and Andres Masegosa. Diversity and generalization in neural network ensembles. In *International Conference on Artificial Intelligence and Statistics*, pages 11720–11743. PMLR, 2022.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized Prior Functions for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

John D Owens, Mike Houston, David Luebke, Simon Green, John E Stone, and James C Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015. doi: 10.1109/ICASSP.2015.7178964.

Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

Daniel Park, Jascha Sohl-Dickstein, Quoc Le, and Samuel Smith. The effect of network width on stochastic gradient descent and generalization: an empirical study. In *International Conference on Machine Learning*, pages 5042–5051. PMLR, 2019a.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019b.

Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 268–284, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Tim Pearce, Mohamed Zaki, Alexandra Brintrup, Nicolas Anastassacos, and Andy Neely. Uncertainty in Neural Networks: Bayesian Ensembling. *arXiv preprint arXiv:1810.05546*, 2018.

Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, Brown Univ Providence Ri Inst for Brain and Neural Systems, 1992.

Huy Tuan Pham and Phan-Minh Nguyen. Global convergence of three-layer neural networks in the mean field regime. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=KvyxFqZS_D`.

Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learning*, pages 5142–5151. PMLR, 2019.

R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3): 21–45, 2006. doi: 10.1109/MCAS.2006.1688199.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Frank P Ramsey. Truth and probability. Technical report, McMaster University Archive for the History of Economic Thought, 1926.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005.

Daniel A Roberts, Sho Yaida, and Boris Hanin. The principles of deep learning theory. *arXiv preprint arXiv:2106.10165*, 2021.

Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33(1):1–39, 2010.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32, 2019.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Grant M Rotskoff and Eric Vanden-Eijnden. Trainability and accuracy of neural networks: An interacting particle system approach. *arXiv preprint arXiv:1805.00915*, 2018.

Walter Rudin. Fourier analysis on groups. *Interscience*, 1962.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. 1998.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *International Conference on Learning Representations*, 2017.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In David Helmbold and Bob Williamson, editors, *Computational Learning Theory*, pages 416–426, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44581-4.

Lisa Schut, Edward Hu, Greg Yang, and Yarin Gal. Deep ensemble uncertainty fails as network width increases: Why, and how to fix it. In *Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20:1–49, 2019.

Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Jonathan Ragan-Kelley, Ludwig Schmidt, and Benjamin Recht. Neural kernels without tangents. In *International Conference on Machine Learning*, pages 8614–8623. PMLR, 2020.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A law of large numbers. *arXiv preprint arXiv:1805.01053*, 2018.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020.

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. Does knowledge distillation really work? *Advances in Neural Information Processing Systems*, 34:6906–6919, 2021.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

Yee Whye Teh, Avishkar Bhoopchand, Peter Diggle, Bryn Elesedy, Bobby He, Michael Hutchinson, Ulrich Paquet, Jonathan Read, Nenad Tomasev, and Sheheryar Zaidi. Efficient bayesian inference of instantaneous re-production numbers at fine spatial scales, with an application to mapping and nowcasting the covid-19 epidemic in british local authorities. 2021.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pages 10268–10278. PMLR, 2021.

Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Christopher Tosh, Akshay Krishnamurthy, and Daniel Hsu. Contrastive learning, multi-view redundancy, and linear models. In *Algorithmic Learning Theory*, pages 1179–1206. PMLR, 2021.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1365–1374, 2019.

Francisca Vasconcelos, Bobby He, Nalini Singh, and Yee Whye Teh. Uncertainr: Uncertainty quantification of end-to-end implicit neural representations for computed tomography. *arXiv preprint arXiv:2202.10847*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Zixin Wen and Yuanzhi Li. Toward understanding the feature learning process of self-supervised contrastive learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11112–11122. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/wen21c.html.

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33: 6514–6527, 2020.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4697–4708. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/322f62469c5e3c7dc3e58f5a4d1ea399-Paper.pdf.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, 2020.

Lechao Xiao and Jeffrey Pennington. Synergy and symmetry in deep learning: Interactions between the data, model, and inference algorithm. In *International Conference on Machine Learning*, pages 24347–24369. PMLR, 2022.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pages 5393–5402. PMLR, 2018.

Lechao Xiao, Jeffrey Pennington, and Samuel Schoenholz. Disentangling trainability and generalization in deep neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10462–10472. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/xiao20b.html`.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

Hongfei Xu, Qiuhui Liu, Josef van Genabith, Deyi Xiong, and Jingyi Zhang. Lipschitz constrained parameter initialization for deep transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 397–402, 2020.

Ge Yang and Samuel Schoenholz. Mean Field Residual Networks: On the Edge of Chaos. In *Advances in neural information processing systems*, pages 7103–7114, 2017.

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019a.

Greg Yang. Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *arXiv preprint arXiv:1910.12478*, 2019b.

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020a.

Greg Yang. Tensor programs iii: Neural matrix laws. *arXiv preprint arXiv:2009.10685*, 2020b.

Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. In *International Conference on Machine Learning*, pages 11762–11772. PMLR, 2021.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, David Farhi, Jakub Pachocki, Xiaodong Liu, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. In *NeurIPS 2021*, March 2022a. URL `https://www.microsoft.com/en-us/research/publication/tuning-large-neural-networks-via-zero-shot-hyperparameter-transfer/`.

Greg Yang, Michael Santacroce, and Edward J Hu. Efficient computation of deep nonlinear infinite-width neural networks that learn features. In *International Conference on Learning Representations*, 2022b. URL `https://openreview.net/forum?id=tUMr0Iox8XW`.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016a.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016b.

Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris C Holmes, Frank Hutter, and Yee Whye Teh. Neural ensemble search for uncertainty estimation and dataset shift. *Advances in Neural Information Processing Systems*, 34:7898–7911, 2021.

Sheheryar Zaidi, Michael Schaarschmidt, James Martens, Hyunjik Kim, Yee Whye Teh, Alvaro Sanchez-Gonzalez, Peter Battaglia, Razvan Pascanu, and Jonathan Godwin. Pre-training via denoising for molecular property prediction. *arXiv preprint arXiv:2206.00133*, 2022.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Biao Zhang, Ivan Titov, and Rico Sennrich. Improving deep transformer with depth-scaled initialization and merged attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 898–909, 2019a.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019b.

Guodong Zhang, Aleksandar Botev, and James Martens. Deep learning without shortcuts: Shaping the kernel with tailored rectifiers. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=U0k7XNTiFEq`.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2018.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019c.

Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1):239–263, 2002. ISSN 0004-3702. doi: https://doi.org/

10.1016/S0004-3702(02)00190-X. URL https://www.sciencedirect.com/science/article/pii/S000437020200190X.