Efficient Processing and Representations for 3D Tasks

Theo W. Costain
St John's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity Term 2023

In loving memory of Max.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Theo W. Costain, St John's College

Acknowledgements

First, I want to thank my supervisor Prof. Victor Prisacariu, without whom this thesis would never have been possible. Thank you for encouraging me to undertake the DPhil and for your patience, support, guidance, and advice over the years. I would also like to thank Prof. David Murray, for tolerating me for several years in my undergrad, without whose kindness, support, encouragement, and advice I would could never have got this far.

Thank you to my examiners Prof. Andrea Vedaldi and Dr. Vassileios Balntas. I am grateful for the time, attention and feedback they have given to my work.

I am very thankful to my friends and colleagues in the AVL for all of the lunches, coffees, discussions and assistance over the years. Particularly to Henry, Michael, Marcelo, Ryan, Wesley, and Kejie. It was a long slog, but I couldn't have gotten through it without you. I'll miss our time in the lab.

My friends, both in London and Oxford, deserve my thanks for keeping me sane: Mahmoud, Roshni, Wil, Laurence, James, Tom, Carlos, Phoebe, Lizzie, Louis, Jaimie, Pippa, Marlene, Emily, Ed, Chris, and my friends from SJCBC.

Thank you to my family, my parents Jan and David, my uncles Michael & David and aunts Pam & Elaine, and grandparents John, Jean, Winston and Madge for their love and support. Finally, and most importantly, thank you to Beth, the best thing to ever happen to me. Without the love and support you've given me over the last seven years, I wouldn't have been able to finish this.

Theo W. Costain
St John's College

Doctor of Philosophy
Trinity Term 2023

Efficient Processing and Representations for 3D Tasks

Abstract

The challenge this thesis aims to address is that of improving the efficiency of deep learning on 3D tasks. We consider this problem from two angles. First how we can improve the efficiency of operations on dense voxel grids. Then second how recent improvements in representational efficiency can be leveraged in computer vision pipelines.

The cubic growth in memory requirements for dense voxel grids means consideration of computational complexity for 3D tasks is of much greater importance than in 2D tasks. Whilst a variety of approaches to addressing this issue are available, in our work we consider two related approaches: First we consider quantisation, which presents a powerful approach to reducing both the energy, computational, and memory requirements of deep networks. Our work investigates how varying the level of quantisation in each of the layers of a deep network can provide for greater memory savings than using the same level throughout, whilst maintaining the same accuracy. Second, where quantisation capable hardware is not available, particularly in power constrained environments, simply reducing the overall size of a model can have greater improvements to overall energy consumption than an equivalent reduction in the amount of computation performed. We approach the problem of reducing model sizes by compressing the weights of a network rather than altering its structure. We examine this task through the lens of approximation, using functional approximations like cosine and Chebyshev series to learn a compact representation of the weights of a deep network.

Recent developments in the form of neural implicit representations (NIRs) present an exciting new approach to improving the representational efficiency of 3D pipelines, considering what tasks are possible using a given representation, rather than just the compute required. Our work examines how early NIR works focused on the reconstruction task alone, hampering their use in later tasks. We show how when training for reconstruction tasks alone, NIRs learn representations that are unsuitable for semantic tasks. To address this, we demonstrate that simple joint training for semantic tasks like classification is sufficient to generate representations that are meaningful even for unseen semantic tasks. Expanding on this work, we propose an approach to contextualise NIRs that are trained for reconstruction alone avoiding any need to retrain the encoders. This contextualising approach then permits training on larger unlabelled datasets to provide improved reconstruction performance, before fine tuning on a labelled dataset to achieve the required semantic capability.

Contents

Acronyms & Abbreviations

ML machine learning

DL deep learning

CV computer vision

FLOP floating point operation

CNN convolutional neural network

KD knowledge distillation

NAS neural architecture search

RNN recurrent neural network

RL reinforcement learning

GPU graphical processing unit

EA evolutionary algorithms

BO bayesian optimization

GP Gaussian process

FPGA field programmable gate array

AD automatic differentiation

TDC tensor decomosition and compression

BTD block term decomosition

IR implicit representations

NIR neural implicit representation

MLP multi-layer perceptron

SOTA state-of-the-art

SDF signed distance function

UDF unsigned distance function

ReLU rectified linear units

AR augmented reality

VR virtual reality

FP32 32-bit floating point number

NeRF neural radiance field

STE straight through estimator

1 Introduction

Contents

Although the foundations of modern deep learning (DL) were laid down over fifty years ago[1; 2], and the underlying mathematics even earlier[3], trends in hardware development have only relatively recently enabled the development and deployment of "super-human" [4] models. Modern hardware, in particular graphical processing units (GPUs), possess millions of times more raw computational power than the machines available when the early implementations were developed[5]. The exponential decrease in the cost of floating point operations (FLOPs) have permitted the development and deployment of models with massive computational[6; 7; 8] and memory [9; 10] requirements. Despite this, the growth in computational capacity provided by modern hardware is being out-stripped by the requirements of the latest and most compute intensive [7; 10] methods. As a result, deploying these new

approaches in data-centers is expensive, and deploying these latest advancements at the "edge"[1] is rarely feasible. Accordingly, there is significant interest in approaches and methods to improve the efficiency of deep learning methods, both in large-scale applications like data-centers and in smaller scale "edge" computing. Further, recent developments in the availability and quality of consumer level augmented reality (AR)/virtual reality (VR) hardware is gradually, but substantially, increasing the demand for efficient approaches to applying DL to 3D data.

In this thesis, we aim to improve the efficiency of deep learning approaches for 3D computer vision (CV) tasks. Approaching the problem from two different angles, we consider first how the efficiency of existing approaches to processing 3D data in dense voxel grid form can be improved, and next, how the introduction of neural implicit representations (NIRs) into deep learning pipelines can improve their efficiency.

Dense voxel grids, whilst the natural extension of conventional pixel images, have exorbitant *cubic* memory (and potentially computational) growth, meaning that simple elevation of 2D methods into 3D, whilst often effective[11; 12], is (relative to the 2D version) extremely expensive. Although one solution to this problem is to use a representation of the 3D data that is *not* dense, *i.e.* point-clouds or triangular meshes (hereafter simply referred to as just "meshes"), for many 3D tasks and pipelines, there is either strong desire, or outright requirement (*e.g.* medical imaging), that dense voxel grids are used to represent 3D data. In these cases, improving the efficiency of 3D data processing is of significant value. Thus, in the first part of this thesis, we consider how we can efficiently process 3D data, using methods like quantisation and weight compression, which improve efficiency through reductions in the energy and memory needed to process data through a given pipeline.

Where it is possible to use different representations of 3D data than dense voxel grids, more avenues to improving efficiency become available. One such example is the recent introduction of neural implicit representations (NIRs), which have opened a new avenue to representing 3D data in an efficient manner. Providing a dense

[1]Edge computing refers to computing that happens at the *edge* of a network, closest to users, thereby providing the lowest latency, but with an assumption that these devices will be more compute constrained than larger devices at or in the network core.

representation of shape and geometry, whilst maintaining memory efficiency comparable to sparse representations like triangular meshes, NIRs store explicit geometry in two parts: a compact encoding and a small decoding network. Information about the stored structure can be extracted at any location in space by conditioning the decoder network with the compact encoding and using the location as input. In the second part of this thesis, we discuss and tackle the challenges of integrating these new representations into existing common CV pipelines. Rather than considering efficiency only in terms of bits and FLOPs, we consider how the *representational* efficiency of NIRs can be improved, which is to say: what tasks can we perform with the representation in its current form, and how can we improve this capacity?

We outline the problems that arise when integrating the current approaches into existing pipelines, and present our approaches to resolving these problems, without compromising the efficiency benefits provided over using classical representations.

1.1 Efficient Processing

Whilst dense voxel grids are far from the most efficient representation of common 3D data, in cases where their use is unavoidable there are a number of avenues to improve the efficiency of their use. A plethora of approaches exist that propose methods to increase efficiency, however their focus is mainly on 2D tasks. Fortunately, lifting 2D techniques into 3D is usually a viable strategy, where memory limits permit, and one which our work utilises. Our efforts in improving processing efficiency focus on two areas: quantisation (section 2.1.4) and tensor compression (section 2.1.4).

1.1.1 Quantisation

In common implementations of deep learning methods, numerical values are represented using a 32-bit floating point number (FP32) (typically IEEE754 [13]). Commonly used in scientific computations, FP32 numbers express a specific subset of the real numbers, but the level of precision they provide is often more than is necessary [14]. As a result, half precision training (*i.e.* 16-bit floats) has become increasingly common, but where even more efficiency is desirable, quantisation

is able to further significantly reduce the number of bits used. Reducing the number of bits needed to represent a value to 8, 4, or even 1 bit, quantisation drastically reduces the memory required to store the values of the weights in a deep network. Further, if the activations (*i.e.* the intermediate states passed between layers in a deep network) are quantised as well, then with the use of specialised hardware that can perform operations on these quantised values directly, it becomes possible to achieve significant reductions in energy use per operation (compared to the same operation in FP32).

Our work in Chapter 3 presents an approach to quantisation that provides for increased efficiency over conventional quantisation methods, by varying the level of quantisation (*i.e.* how many bits the values are quantised down to, also referred to as bit-width) at each layer in the network. Automatically determining optimal bit-widths for each layer in the network is in essence a neural architecture search (NAS) problem (section 2.1.1), and by using a low dimensional parametric function that describes the distribution of bit-widths in each layer of the network, we are able to use bayesian optimization (BO) in conjunction with a multi-task Gaussian process (GP) to convert the NAS problem into a hyperparameter search. With this system, we are able to predict optimal quantisation schemes without exhaustively searching the space.

Further, we demonstrate how this approach can be utilised in 3D pipelines, including video classification, 3D segmentation, and NIRs. To the best of our knowledge, no other work has investigated the application of quantisation to these tasks.

1.1.2 Weight Compression

Whilst quantisation can best provide its benefits when custom hardware is available, tensor compression finds its value in edge (*i.e.* power constrained) environments. In modern hardware, the energy demands of loading data from DRAM vastly outweigh the costs of many arithmetic operations, in some cases by as much as $\times 1000$ [15], whilst cache reads require substantially less. Therefore in energy constrained environments, it is desirable for as much of the model as possible to

fit in cache to minimise DRAM loads. Further, it is often desirable that devices at the edge are as cheap as possible, which means that the custom hardware usually needed to gain the most benefit from quantisation is not available. In these settings, it becomes desirable to reduce the memory (and if possible computational) requirements of deep learning pipelines. This is a large area of work, with a variety of approaches, which we covered in more detail later (section 2.1), but an group of approaches that can be applied regardless of a given pipeline's architecture are tensor decomposition and compression (section 2.1.5).

In Chapter 4, we present our work on compressing the weights of layers in deep networks. By re-framing conventional discrete convolution on regular grids as discrete convolution over continuous functions of space, we are able to describe the weights of a convolutional neural network (CNN) layer's filter as parametric functions. Replacing the original parametric functions with functional approximations, such as Cosine and Chebyshev series, we are able to reduce the number of values needed to describe each layer of the network, with minimal reduction to the overall performance of the network. Importantly, unlike other similar approaches to compressing kernel weights, we aim to approximate the weights of a trained network. This means our approach does not require training from scratch, and can instead be applied as a post processing step requiring only a modicum of fine tuning. A lack of viable comparisons mean the experiments presented address only 2D tasks, however, we discuss the extension to 3D tasks and how trivially it can be conducted.

1.2 Efficient Representations

In 3D tasks where there is no need to use dense voxel grids, efficiency gains can be achieved through the use of representations that store information in a sparse manner. If we consider the environments we live in, often much of the total volume of a given space is empty, therefore representing this information in a sparse manner can lead to a substantial reduction in memory compared to dense representations. As a result, much of the literature that tackles common computer vision tasks on

3D data, uses representations other than dense voxel grids, such as point-clouds or meshes, however, these sparse representations can have significant limitations.

Although one of the most commonly captured forms of 3D data, point-clouds provide no surface information beyond what can be inferred from local context, and meshes whilst providing the missing surface information, are difficult to capture directly, and are usually constructed from other modalities.

One potential solution to these problems can be found in the recently proposed approach of neural implicit representations (NIRs). Trading some computational complexity for reduced spatial (memory) complexity, NIRs can represent 3D shapes and scenes at lower memory costs [16] than sparse representations, whilst providing continuous dense shape information. To achieve this, these approaches learn continuous functions over space that describe the geometry of the shape or scene, with the "neural" aspect of their naming arising from their implementation in the form deep networks, typically encoding conditioned multi-layer perceptrons (MLPs).

Although NIRs are able to store 3D shapes and scenes more compactly than comparable classical representations, this comes at an increased computational cost when rendering/extracting an explicit structure. As these methods are relatively new, little attention has been paid in the existing literature to common downstream tasks like semantic segmentation (as is discussed in more detail both below and in section 2.3.2), using NIRs for these tasks would require the extraction of an explicit shape eliminating any efficiency benefit from their use.

Our work in the second part of this thesis considers if and how implicit representations can be used in common tasks downstream of reconstruction, such as classification and segmentation, without requiring explicit rendering/extraction. Our work outlines approaches that can be used to improve the efficiency of common 3D pipelines by integrating implicit representations to provide memory savings, whilst allowing more tasks to be performed with the same networks.

1.2.1 Generalising NIRs

Although early NIR works[16; 17; 18] were quick to tout the admittedly impressive capabilities of their approaches, the vast majority of the works considered only how their approaches could be used for the task of geometric reconstruction (*i.e.* generating a high fidelity point-cloud or mesh from the shape encoding). The reconstruction abilities of the original approaches, and the improvements produced by later works, present a compact representation of complex geometry, which is is achieved in part by trading spatial (memory) complexity for computational complexity. Importantly, the original works provided no insights into how or whether their shape encodings might be directly integrated into semantic tasks common in computer vision. If this integration is not possible, then performing semantic tasks using NIRs would require extraction of explicit geometry that could be used in existing pipelines, entirely defeating the benefits achieved from using NIRs in the first place. Alternatively, new pipelines would need to be developed that could operate directly on these shape encodings, but this would likely represent a substantial duplication of work not to mention the uncertainty inherent in the development of any new pipeline.

Our work in Chapter 5 begins with experiments into the semantic interpretability of the shape encodings themselves. We found that the shape encodings that were learnt for the reconstruction task displayed poor separability when used in semantic tasks. This observation suggests that if NIRs are trained for reconstruction alone, using them in existing pipelines would introduce the previously mentioned issues, and eliminate any improvements to overall efficiency gained through their. The remainder of our work then explores how a simple joint training scheme can entirely alleviate this problem, paving a route for the use NIRs in common task pipelines with a minimum of modification.

1.2.2 Contextualising NIRs

Whilst joint-training is a viable solution to permit the use of NIRs in common task pipelines, such as classification or semantic segmentation, this approach can be

extremely costly in terms of both training and data acquisition. Whereas cameras that capture 2D images are ubiquitous and affordable, the same cannot currently be said for 3D data. As a result, the availability and quantity of densely labelled image data vastly outstrips that of 3D data (*e.g.* the COCO [19] dataset contains over 200k labelled images where as ScanNet [20], the largest fully annotated indoor 3D dataset, contains only 1.5k labelled scenes).

Whilst the increasing availability of 3D sensors and systems that capture 3D information[2] will likely improve the availability of unlabelled 3D data that can be used to train NIRs for the reconstruction task, labelling the 3D data will still remain expensive. Therefore, in an ideal case, it would be possible to maintain the separation between training for reconstruction tasks and semantic tasks, like with most existing NIRs works, without compromising performance.

Chapter 6 details our work on enriching the shape encodings of NIRs for use in semantic tasks. We show that, whilst shape encodings learnt from a reconstruction task alone are insufficient for use in semantic tasks, the necessary information is still preserved in some form within the encodings. Using a lightweight contextualising network, we reveal the hidden semantic information forming a small contextualising encoding. When combined with the shape encoding, this new contextualised encoding is able to recover full performance in semantic tasks. We show that this approach allows for models to be trained on large unlabelled datasets, and then fine-tuned on smaller labelled datasets providing not only full semantic accuracy, but increased reconstruction performance, when compared to an NIR trained only on the smaller labelled dataset.

[2]Recent AR systems such as the Meta Quest Pro and Apple Vision Pro, whilst not containing any explicitly 3D sensors, use a variety of techniques such as multi-view stereo [21] to infer 3D data from 2D images in real time.

1.3 Thesis Structure

This thesis is presented in an integrated[3] format, with the bulk of most chapters presented in the format in which they were published or submitted for publication. In these cases, to aid the reader and present the thesis as a cohesive undertaking, we have included an additional introduction and conclusion, as well as a highlighting of the contributions from the paper.

Chapter 2 places the work we present in the wider context of the existing research. In the previous section we explored the two angles from which we explore efficiency improvements in deep networks: improving the efficiency with which data is processed and stored, and exploring how the representational efficiency of NIRs can be improved. As the remaining chapters of this thesis can be divided into these two groupings, we separate the thesis into two parts.

Part I covers our work examining the efficient processing of 3D data, starting with our work on non-uniform quantisation schemes in chapter 3, followed by our work on compressing the weight tensors of deep networks in chapter 4.

Part II covers our work improving the representatonal efficiency of NIRs and integrating them into deep learning pipelines and methods, first in chapter 5 with our work exploring how the encodings of NIRs can be generalised to have meaning in semantic tasks, and second in chapter 6 which explores our efforts to reveal information hidden in the encodings, permitting their use in semantic tasks without requiring expensive retraining.

Finally, we summarise and discuss the work presented in this thesis, gathering the key insights and discussing potential avenues to explore based on the work presented.

1.4 Publications

- Marcelo Gennari do Nascimento, Theo W Costain, and Victor Adrian Prisacariu. Finding non-uniform quantization schemes using multi-task gaussian processes. In *European Conference on Computer Vision*, 2020

[3] `https://www.mpls.ox.ac.uk/graduate-school/information-for-postgraduate-research-students/submitting-your-thesis`

- Theo W Costain and Victor Adrian Prisacariu. Approximating continuous convolutions for deep network compression. In *British Machine Vision Conference*, 2022

- Theo W Costain and Victor Adrian Prisacariu. Towards generalising neural implicit representations. In *European Conference on Computer Vision Workshop*, 2022

- Theo W Costain, Kejie Li, and Victor Adrian Prisacariu. Contextualising implicit representations for semantic tasks. *Submitted For Publication*, 2023

2
Literature Review

Contents

This chapter aims to provide context to the work presented in the remaining chapters. In exploring the work that preceded ours, we hope to show both how the ideas behind our work arose, and what our work adds to the existing literature. Reflecting the division of the technical chapters of this thesis, the first section covers works relevant to the first part of the thesis, and the second and third sections cover the second part.

In the first section, we consider the efficiency of operations and models used in deep learning. We explore how the interplay between the availability of software and hardware drove varying approaches to improving the efficiency of deep learning.

Starting with architectural design, we discuss how the size and complexity of models was reduced. Then we turn to operational efficiency, starting with methods such as pruning and knowledge distillation, before moving onto methods such as quantisation (which has strong interplay with hardware considerations) and compression, which are the respective focuses of chapter 3 and chapter 4.

In the second section, we turn to how better choices in representation of 3D data can drive meaningful improvements in efficiency. We start with a brief consideration on how moving from 2D data to 3D presents new and complex challenges in its processing, as well as different popular approaches to processing 3D data in voxel grid form. Continuing with a consideration of *classical* representations, we discuss different approaches to the representation of 3D data and how these different approaches manage the trade-offs required.

Finally, in the third section, we turn to new implicit representations, such as neural radiance fields (NeRFs) and neural implicit representations (NIRs), and discuss the improvements and shortcomings they present.

2.1 Improving the efficiency of operations and models

There has been consistent interest in improving the efficiency of convolutional neural networks (CNNs) since the *relatively* recent explosion of work spurred by the publication of AlexNet [4]. Efforts to improve the efficiency of deep networks have targeted every level of the "stack" from high level architectural design [26; 27], to low level hardware design [28].

The rest of this section will cover efforts to improve efficiency through: the architectural design of networks themselves (Section 2.1.1), the training of smaller networks using the outputs of larger networks (knowledge distillation (KD), Section 2.1.2), the removal of unnecessary parameters (pruning, Section 2.1.3), the quantisation of weights and activations in the network (Section 2.1.4), and finally mathematical manipulations of the weights and kernels of network layers (Section 2.1.5).

2.1.1 Network Architecture

Whilst a substantial body of research has increased performance of deep networks through careful handcrafted network design, there also exists an array of work which has increased network efficiency in an automated fashion.

AlexNet [4] started the modern explosion in CNN work, but its structure was not particularly efficient. Subsequent works, like ResNets [29] and VGG[30], began to rapidly improve both the performance and efficiency of state of the art networks. As the popularity of CNN based methods gained ground, more explicit focus was given to the efficiency of these approaches. Early works like SqueezeNet [27], which popularised the use of 1×1 convolutions, and MobileNets [26], which introduced the idea of depth-wise convolution, achieved comparable performance to state-of-the-art (SOTA) methods of the time but with a fraction of the total floating point operations (FLOPs). Various other works [31; 32; 33; 34] employed a variety of structures and techniques to further improve performance, whilst attempting to minimise any increases in computational cost. However, all of these methods were discovered through manual trial and error.

In contrast, the field of *neural architecture search* (NAS) seeks to discover optimal[1] architectures automatically. One of the earliest works [35] in this regard used a recurrent neural network (RNN) trained using reinforcement learning (RL) to predict network architectures for image classification and language modelling. Although effective, this approach required monumental amounts of compute; Training the image classification network used 800 graphical processing units (GPUs) for over 3 weeks[2]. Following this initial work, many approaches [35; 36; 37; 38] favoured the same RL training, where a "controller" network manipulates some underlying network structure.

Different to the RL approaches, methods have made use of gradient descent [39; 40], evolutionary algorithms (EA) [41], and bayesian optimization (BO) [42; 43]. Gradient descent based approaches use clever parametrisation to make the choice

[1] for a given goal, accuracy/performance or efficiency or a combination of the two

[2] `https://openreview.net/forum?id=r1Ue8Hcxg`, accessed April 2023

of network structure differentiable, allowing the structure to be optimised with standard gradient descent.

Like RL, both EA and BO are common methods used for search in spaces with large cardinality, and where the computation of gradients that would allow direct optimisation are either infeasible, impractical, or impossible to compute.

EA are a class of search methods [44; 45; 46; 47] that effectively mimic Darwinian natural selection[48]. Drawing from an initial pool of samples, rounds of crossover and mutation followed by evaluation against a fitness criterion are performed, following which only the "fittest" samples are preserved as input for the next round. These approaches have a history of use in a variety of areas [49] before the recent uses in NAS. A notable benefit of EA is that because a pool of samples is maintained at all times, these method will often generate solutions with significantly different structures [50], but similar performance. The main downside of EA methods is that the fitness of each sample must be evaluated, which in the case of NAS likely means training the network, a very costly procedure. Whilst approaches to limit this cost have been proposed [51; 52], ultimately there is *no free lunch*[53], and the reductions in cost induce error in the calculation of fitness.

BO methods present a classical and probabilistic approach to search. Assuming a Gaussian process (GP) prior over the search space, they provide a rigorous balance between exploration and exploitation of the search space. One of the key benefits of BO over RL and EA is that by imposing the GP prior over the search space[3], it is able to be much more efficient per sample. Whilst their use is more popular in the closely related problem of hyperparameter search [54; 55; 56; 57], the need to invert a covariance matrix with $\mathcal{O}(n^2)$ complexity given n samples can render their use in high dimensional spaces extremely challenging. Because the dimensionality of the search space for our work in chapter 3 is relatively small, we are able to make use of BO in our method. A few other works [43; 58] have also made use

[3]Again though, there is *no free lunch* [53], so this prior assumption may reduce performance in certain cases.

of BO in NAS, both making significant strides by developing a distance measure that can be applied to the structure of a neural network.

2.1.2 Knowledge Distillation

Although it is commonly observed that larger models exhibit better performance on a given task [29; 59; 60], it is not readily apparent whether this is because the larger network is more capable or because it is easier to train. Hinton et al. [61], in their seminal work, developed the idea that it might be possible to use a large trained model to aid in the teaching of a smaller model.

The core idea in the work was a re-imagining of what is learned by a deep network: instead of viewing learning as discovering a set of weights and biases for a particular architecture, they view learning as discovering a mapping from input vectors to output vectors. Thus in the proposed student-teacher training paradigm[4], supervision is not just in the form of minimising the cross entropy between the target label and the learned probits, but also minimising the cross entropy with the soft targets produced by the teacher network.

This line of effort has yielded a number of methods for improving the efficiency of deep networks. For example, Mirzadeh et al. [62] showed that there is a lower bound to the size of student network a teacher can supervise, but that a progressive teaching scheme involving intermediate “teacher-assistant” networks can eliminate this problem. Romero et al. [63], showed that adding teacher supervision to intermediate layers of the networks allowed for the training of deeper thin networks, and Zhang et al. [64] showed that self-distillation can be used to improve the performance of deep networks at *essentially* no cost.

2.1.3 Pruning

That knowledge distillation works, necessarily implies that there is substantial redundancy, or "dead-weight", in many large deep networks. Rather than training smaller networks like KD, which can be quite expensive, pruning attempts to

[4]For the case of classification tasks

directly reduce or remove this dead-weight. How this pruning is achieved necessarily produces a convenient dichotomy[65]: either you only remove specific parameters of the network *inducing sparsity*, or you remove layers or filters from the network wholesale *not inducing sparsity*.

Amongst the early works in the sparsity inducing vein is the seminal work "Optimal Brain Damage" [66], which automatically removed individual parameters in the network. Later works [14; 65] improved on this, particularly the latter work which popularised the train, prune, re-train loop that is now common. Importantly, Frankle and Carbin [67] and its follow-up work [68] demonstrate that it is possible to prune a network before training, *i.e.* training only the pruned subnetwork, realising the benefits to size reduction in training as well as inference.

The main drawback to methods that induce sparsity is that modern hardware and software is not best suited to handling sparse operations (we will encounter this problem again in section 2.2). Methods have been proposed to avoid this problem, maintaining the density of a network or architecture by either removing redundant channels [69; 70; 71], or entire layers[65].

Whilst the above methods tackle redundancy throughout the network, other works have presented methods that achieve the same ends albeit focusing only on the final layers of the networks, and in particular the classifier. These works[72; 73; 74; 75] observe that the final classifier layer of many deep networks is extraordinarily parameter intensive and demonstrate that it's possible to used fixed classifiers significantly reducing the number of parameters in the network. The earliest work to propose this approach was Hoffer et al. [72] who demonstrated that a network using a fixed orthogonal projection matrix is able to achieve the same validation error as a learned classifier. Further, they show that an easily generated Hadamard matrix is sufficient, requiring less memory than the randomly generated projection matrix. Subsequently, Pernici et al. [73] improve on the fixed classifier approach using regular polytopes that make maximal use of the representation space.

Whilst the methods discussed in both this and the previous two section (knowledge distillation, section 2.1.2) are not directly related to the work in this thesis,

the underlying ideas about redundancy and excess complexity are an essential and important part of both the inspiration for and the landscape in which our work in chapter 4 exists.

2.1.4 Quantisation

If, as we have discussed so far, there is redundancy in the structure of networks and in the parameters of the networks, then it stands to reason that there might be redundancy in the *numerical* values of the parameters themselves. Quantisation presents one approach, section 2.1.5 presents another, to exploiting this potential redundancy.

In the context of this thesis we use the term quantisation to refer to the process of representing a number using fewer bits than the standard IEEE754 single precision floating point [13]. Reducing the number of bits used to represent a value, although reducing precision, also decreases the energy and space[5]required to perform the calculation, allowing both efficiency and throughput to be improved.

Arguably, the simplest form of quantisation involves quantising only the weights of a network [76; 77; 78; 79; 80; 81]. Whilst only useful for reducing the size of the network in memory, the extent of compression possible without any loss in performance is staggering. Early works such as Deep Compression [76] achieved size reductions up to 35× in the case of AlexNet[4]. Later works, such as Rastegari et al. [77], produced more aggressive quantisation, going as low as 1-bit with no loss in accuracy.

Ideally, we would like to realise the benefits of quantisation in more areas than simply storage, however the obvious next step, quantising inputs and activations, creates a challenge. In order for automatic differentiation to be applied to a network, every node in the computational graph must have a defined gradient [1]. The quantisation operation, has gradients equal either to 0 or ∞ (at the transitions between different values), and as such does not allow gradient information to flow backwards through the network. Most quantisation methods take the same

[5]here we mean the die area of integrated silicon devices or field programmable gate arrays (FPGAs)

approach to resolving this problem, and use the *relatively* simple but effective straight through estimator (STE) [82]. Simply put, the STE *pretends* that the forward pass is an identity function, and therefore has a gradient of 1 everywhere. Although quantisation can cause a reduction in accuracy, like Han et al. [14] from section 2.1.3, fine-tuning the network after quantisation can allow much of the performance to be recovered. In practice, limitations imposed by commonly available automatic differentiation (AD) systems [83; 84] mean that training quantised networks is somewhat more involved. Rather than maintaining a quantised network, the network is stored in FP32 and quantised before each forward pass, and the STE is applied in the backwards pass. A number of popular works have been trained in this way [77; 80; 85; 86; 87].

Ultimately, the greatest energy savings can be realised by the application of quantisation during training, however this introduces yet more complexity. As training converges, the error that is back-propagated through the network reduces and accordingly so do the gradients, which ultimately causes the gradients to be rounded to zero. A variety of approaches have been successful in mitigating this problem, including DoReFa-Net [85], which proposed stochastic rounding (where values are rounded up or down), and SWALP [88], which averages to suppress the quantisation errors.

2.1.5 Tensor Decomposition & Compression

Unlike quantisation, tensor decomosition and compression (TDC) exploits redundancy not in individual values, but across groups of values. The simplest example of this is separable convolutions, where provided specific conditions are met[6], the complexity of a convolution can be substantially reduced by breaking the application of a single large kernel into the application of several kernels of lower dimensionality.

Whilst this technique has a long history of use in computer vision as a method for increasing the efficiency of convolution operations, Rigamonti et al. [89] were

[6]In the case of 2D convolution, the condition is that the kernel matrix $A \in \mathbb{R}^{n \times m}$ can be expressed as $A = BC$ where $B \in \mathbb{R}^{n \times 1}$, $C \in \mathbb{R}^{1 \times m}$

one of the first works to demonstrate that these separable filters might be learned directly. Another related approach that was touched on in section 2.1.1 are the depth-wise and point-wise convolutions introduced by Howard et al. [26], which operate on a similar idea to that of conventional separable filters.

Subsequent works [90; 91; 92; 93; 94; 95; 96] made use of tensor decomposition techniques from linear algebra, such as CP [97] and block term decomosition (BTD) [98], to decompose the entire kernel tensor. These approaches, like separable convolutions, allow not only for improvements to the speed at witch the operations can be performed, but also reductions to the amount of memory required to store the networks themselves.

Whilst much of the focus in the above methods is concerned with their improvements to network inference speed, an important aspect of these methods is the reduction in parameter count, and therefore memory requirements, that they provide. This aspect is what our work in chapter 4 focuses on, however, works that solely focus on compression of kernel weights are less common.

Both Saldanha et al. [99] and Zamora et al. [100], propose a method to learn CNN filters constrained by structural priors. Specifically, they learn a parametrisation of Gaussian and Gaussian derivative filters using fractional calculus, learning filters that require substantially fewer parameters than standard filters whilst maintaining similar performance. Whilst effective, these two approaches share a major drawback in that, as the space of kernels they can learn is substantially constrained, the require training from scratch, a problem not present in our work.

2.2 The challenges of 3D data

Likely millions of person-years, if not more, have been expended in the storage, capture, and representation of 2D information as pixel grids. Whilst there are formats (*e.g.* SVG) that don't use pixel grids, these are rarely if ever used to represent real world 2D data. The existing tools and operations designed for pixel grids combined with the availability of incredibly efficient compression techniques,

mean there is very little need or desire to find more efficient representations of/for pixel grids.

On the other hand, naïvely extending pixel grids into 3D (*i.e.* voxel grids) results in a cubic, rather than quadratic, growth in both memory and computational requirements. Consider that an image taken on the highest resolution widely available camera sensor (200MP[7]) requires only 200 MB to be represented, but a 3D grid with slices of the same resolution requires 9.86 TB. On top of the significant memory requirements of dense voxel grids, even capturing 3D data in the first place is a devilishly complex problem, and is an active topic of research beyond the scope of this thesis. As a result, with the exception of medical scans, the use of dense voxel grids in the literature is rare, and especially so for real world data.

To avoid this issue, the vast majority of methods used for 3D tasks opt instead to use a different representation than dense 3D grids, of which we give an overview in the next part of this section (Section 2.2.1). However, recent developments in the literature have given rise to a new approach to representing 3D shape and structure, namely implicit representations (IR). Alongside the closely related NeRFs (Section 2.3.1), NIRs store 3D shape and structure as a learnt function representing a signed or unsigned distance function[8].

Relevant to the second part of this thesis, NIR are able to store high fidelity representations with only a small amount of memory. These approaches represent a promising alternative representation of 3D shape and structure which we cover in the last part of this chapter (Section 2.3.2).

2.2.1 Classical Representations and Operations

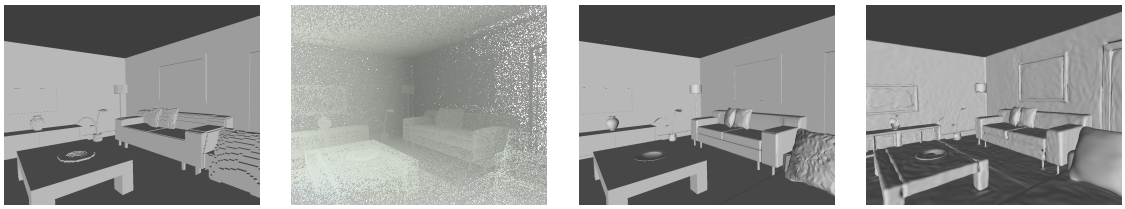
Although extremely expensive in terms of memory/spatial complexity, dense voxel grids represent the simplest form on which to perform operations. As the data is heavily structured, and implicitly ordered, processing data represented this way is very simple. In many cases, pipelines developed in 2D settings can be directly

[7] https://semiconductor.samsung.com/image-sensor/mobile-image-sensor/isocell-hp3/, accessed April 2023

[8] Whilst some IR methods learn occupancy probabilities, there is virtually no difference to a distance function followed by a sign operation.

applied, by simply expanding the number of spatial dimensions to be operated on[11]. For dense 3D data such as video, this approach is very effective, however, unlike with 3D data representing shapes, objects, and scenes, there is little to no sparsity present in the data[9].

Over time a number of different alternative approaches to dense grid representations have been developed. Where there is significant sparsity in the data, it becomes possible to exploit this sparsity to drastically reduce the memory requirements to store a given shape or scene. Most closely related to dense voxel grid representations is the OcTree. Where the voxel grid has a fixed number of elements along each axis, the octree stores 3D data as a hierarchical data structure, where each voxel (node) can be subdivided into 8 smaller voxels. This approach allows for increased fidelity only where signal/data is present. However, this approach eliminates much of the ease that regular grids provide during processing, whilst still imposing structural constraints. The tree structure of the data maintains some of the similarity between data and memory locality, but loses many of the advantages that voxel grids inherit from decades of optimisation in processing 2D grids. Accordingly, few methods have attempted to make use of this representation of deep learning tasks[101; 102].



(a) Voxel Grid (134 MB) **(b)** Point Cloud (12 MB) **(c)** Mesh (1.76 MB) **(d)** Implicit representation (790 kB)

Figure 2.1: A comparison of the memory required to represent the same scene using different representations. From left to right: a dense voxel grid at 512^3 resolution, a point-cloud with 1 million points, the original mesh from which the other representations are derived, and an implicit representation using the method from [103].

[9]It would be reasonable to argue that for some cases, *e.g.* action recognition on video data, only *foreground* elements may be relevant and could be segmented out producing sparse data. However, unlike with 3D shapes and scenes, the background data may contain meaningful cues or information, whereas sparsity in 3D shapes and scenes comes from truly empty space.

One of the earliest alternative approaches, arising initially from the field of computer graphics, is the polygon (most commonly triangular) mesh. Composed of a series of vertices and edges, typically forming triangular faces, meshes represent a compact representation of 3D shapes and scenes. Varying the size of the triangular faces (larger in flatter areas and smaller in areas with higher curvature) meshes can represent shapes with arbitrary and locally varying fidelity. However, these efficiencies come at a cost during processing, as the spatial relationships are not preserved in the structure of the data in memory to the same extent as they are for dense voxel grids. Fortunately, as meshes are (in their simplest form) a conventional mathematical graph, a variety of approaches can be utilised to process them, from very classical graph based methods[104; 105; 106; 107; 108; 109; 110; 111; 112], to more modern approaches [113; 114; 115; 116; 117; 118; 119; 120]. On top of this, in the case of explicitly triangular meshes, methods have been proposed to exploit the triangular structures to redefine both the convolution and pooling operations, familiar from 2D settings, for meshes[121; 122]. One, if not perhaps the main drawback of meshes is that it is not possible to directly/explicitly capture them from real world sensors. Instead they must be generated from other forms of captured data[123; 124].

Point-clouds represent the data format that is most commonly captured from commodity 3D sensors. Simply a collection of points in 3D space, sometimes with associated colour and or surface normal information, point-clouds trade all structural information for an extremely compact representation. However, whilst humans are often able to infer structural information from visualisations of point-clouds, this task can be much more challenging for machine systems . Despite this, the extreme ease with with point-clouds can be captured, means there exists a plethora of approaches to deep learning on point-clouds. Two of the earliest and arguably best known approaches are PointNet [125] & PointNet++ [126], which treat the point-clouds as an order invariant set of feature vectors and process them using multi-layer perceptrons (MLPs) and pooling operations (as well as distance based grouping operations in the case of PointNet++). Various other approaches have

been proposed, including a number of methods that attempt to explicitly mimic 2D convolution in 3D[112; 114; 115; 120] as well as expanding and extending the ideas of grouping and interpolation underpinning PointNet++[112; 115; 127]. Recently, other works have explored the use of new transformer [128] architectures, applying them to point clouds [129; 130; 131; 132].

Finally, one other common similar representation is *sparse* voxel grids Graham et al. [133]; Choy et al. [134], however these representations arguably have more in common with meshes than they do with voxel grids in terms of performing computations with them, sharing many of the same downsides and disadvantages, except that they posses fixed density and preserve *some* structural/surface information[10].

2.3 Implicit Representations

All the previously discussed representations encode 3D information in an explicit manner (*i.e.* ignoring decompression and parsing, no computation is needed to recover the information), however the alternative approach presented by implicit representationss (IRs) have recently seen an explosion in interest. Opposed to explicit representations, IRs store the 3D information in the form of a function over space (and viewing direction in the case of NeRFs).

We suggest that there are two distinct families of IR methods, that whilst sharing structural similarities, differ greatly in many areas such as training, aims, and method. These families are neural radiance fields (NeRFs), and neural implicit representations (NIRs). The first family tackle the problem of multi-view reconstruction and novel-view synthesis, typically overfitting an MLP to a single scene or object and using ray-tracing & rasterisation like approaches to extract information from the representation. The other family, which the second part of this thesis is concerned with, instead focuses purely on surface reconstruction, and aims to learn a generalised function that can be conditioned to represent multiple objects or scenes.

[10]Not withstanding resolution limitations associated with Nyquist sampling limits

2.3.1 Neural Radiance Fields

The recent introduction of NeRFs [135] has lead to an explosion of research. This groundbreaking work presented a step change in the task of novel-view synthesis. Dependent on fourier features [136] for its ability to reconstruct high-frequency signals, NeRF used a MLP to learn functions of colour and density over space, which are then integrated along a view/ray to give the value of a particular pixel.

A broad range of papers[137; 138; 139; 140; 141; 142; 143; 144] then arose based on this original foray, tackling an array of concerns from the original paper. Early follow-on works extended or improved the capabilities of NeRFs: Martin-Brualla et al. [137] provided extensions that allowed for training over sets of images with varying illumination and occlusions, allowing for the reconstruction of famous global landmarks from disparate collections of photos. Barron et al. [138] reduced the effect of aliasing that occurred when sampling from a single ray (without the use of more rays), by sampling from anti-aliased conically projected frustums. Whilst the original NeRF was very slow to train, both PlenOctrees [139] and InstantNGP [140] proposed methods to drastically speed up training, using OcTrees and multi-resolution hash grids respectively. Where the original NeRF is over-fitted to a single image, both IBR-Net [141] and CodeNeRF [142] respectively attempt to allow generalisation to novel scenes using a "ray-transformer" that draws on source images at rendering time, and latent coding to allow for generalisation.

Others have proposed the use of NeRFs in other tasks, such as pose estimation [145; 146], semantic segmentation [147; 148; 149; 150; 151], or, more closely related to the second part of the thesis as well as the next section, surface reconstruction[152; 153; 154].

NeuS [152] and VolSDF [153] both suggest different methods that effectively replace the volume density term from the original NeRF paper with some metric related to the signed distance function of the shape at a given location in space, allowing their method to reconstruct detailed shapes and surfaces from 2D images.

2.3.2 Neural Implicit Representations

Neural implicit representations (NIRs) first rose to popularity from the simultaneous work of Mescheder et al. [16], Park et al. [17], and Chen and Zhang [18]. All three works propose the same general structure: an MLP that takes a (x, y, z) coordinate in space and returns either the probability that the location is inside the shape [16; 18] or the value of a signed distance function (SDF) at that location [17]. The key advantage that these approaches provide, over conventional representations, is that their memory growth requirements are not directly tied to the resolution of the data they are storing. Rather, the resolution they are able to store is instead a function of both their memory as well as their training and inference-time computational budget. Therefore in using these methods, it becomes possible to trade off between model size, training time, inference time, and output resolution, to achieve a desired balance depending on the application.

Following the three initial methods, various methods were suggested to improve on different aspects of NIRs. One group of methods sought to improve the fidelity of NIRs and/or to broaden the types of training data NIRs could be trained with. SAL [155] proposed a method that removes the need for *signed* ground truth information when learning a signed distance function. NDF [156] instead learns an unsigned distance function (UDF) allowing their method to learn non-watertight shapes, mitigating wall thickness constraints in other methods. As well, they proposed a gradient based rendering scheme for extracting the surface; required because marching cubes cannot be applied directly to unsigned distance functions. More recently, UDF based approaches have been increasingly popular in the recent literature [157; 158; 159; 160; 161; 162]. Gropp et al. [163] introduced the eikonal loss, which encourages the network to learn a unit norm gradient, like a metric SDF, acting as geometric regularisation, improving both the smoothness as well as the accuracy of the reconstructions. SIREN [103], similar to Tancik et al. [136], presented a method that allows NIRs to better represent high frequency signals, by replacing rectified linear units (ReLU) activations with sinusoidal activations (along side a change to initialisation schemes required for the method to work).

Separately, a collection of works observed that the size and scale of object that could be reliably represented was limited when using a single vector as an encoding (as in [16; 17; 18]). To alleviate this limitation, a range of methods that altered the structure of the encodings were proposed. Arguably most well known is Convolutional Occupancy Networks [164], which proposed to use a grid (2D or 3D) of features that are interpolated between at a given location in space to provide a specific encoding for that location in space. Other works proposed similar approaches: Chibane et al. [165] proposed multi-resolution feature grids to better capture local and fine grained details. Rather than interpolating the encodings, Deep local shapes [166] and Jiang et al. [167] divide space into separate regions learning a separate encoding for each region, as well as providing methods to ensure consistency of the representations at the boundaries.

Whilst there has been much attention paid to the performance of NIRs for reconstruction tasks, few works have considered tasks other than reconstruction [160; 168] let alone the more abstract idea of "downstream" tasks [169], which the second part of this thesis considers.

One potential issue with NIRs not discussed in the literature is whether it is possible to perform semantic tasks on the learnt encodings without extracting an explicit representation. Whilst, both RangeUDF [160] and Kohli et al. [168] consider semantic segmentation tasks alongside reconstruction tasks, the former segmenting entire scenes and the latter considering only part segmentation of single objects, our work in Chapter 5 argues that when NIRs are *not* trained for semantic tasks alongside reconstruction, it is not necessarily possible to achieve satisfactory semantic task performance without first extracting explicit geometry. Luigi et al. [169] propose a method to extract semantic information from the weights of a NIR to produce a single vector that is meaningful for later semantic tasks, however their method operates on MLPs over-fitted to single objects rather than the more common encoding conditioned decoder that is prevalent in the literature.

Part I

Efficient Processing

3

Non-Uniform Quantization Schemes

Contents

3.1 Introduction

This chapter concerns our work on non-uniform quantisation schemes. The content consists of the paper "Finding Non-Uniform Quantization Schemes using Multi-Task Gaussian Processes"[22] (ECCV 2020), as well as further experiments and discussion that expand the work into 3D settings.

The core idea arises from the observation[170; 171] that different layers in deep networks may be responsible for different levels of abstraction, and accordingly, require varying levels of fidelity for the network's overall performance. However, deciding the fidelity for each layer in the network, admits a hyperparameter

search problem with exponential growth. Given these hyperparameters ultimately determine the structure of the network, searching this space is in essence neural architecture search (NAS).

In our work, we propose the use of bayesian optimization (BO) in combination with a multi-task Gaussian process (GP) to search the space. To minimise the dimensionality of the search space, we express the quantisation levels of each layer in the network in terms of low dimension parametric functions (Bézier[172] and Chebyshev[173] polynomials). In the paper, we focus on generic 2D networks, as is common in the existing literature, where our experiments show the efficacy of this approach across a range of networks and models. However, the methods we propose apply equally well to 3D settings and tasks. As such, we include our extension experiments that were not included in the original paper, exploring the application of the method to 3D settings and tasks. Finally, we discuss the different trade-offs that occur in 3D settings and accordingly the relevance of our method in these settings.

3.1.1 Contributions

- We re-cast the problem of NAS as hyperparameter search, by defining the structure of the network in terms of a low dimensional parametric function.
- We propose the application of BO to this hyperparameter search using a GP prior, to efficiently search the space.
- Through our experiments, we demonstrate the effectiveness of our approach, including on large datasets and models such as ImageNet/ResNet50.
- Our additional experiments (not included in the original paper), demonstrate that our approach performs equally well in higher dimensions.

Finding Non-uniform Quantization Schemes Using Multi-task Gaussian Processes

Marcelo Gennari do Nascimento, Theo W. Costain( ), and Victor Adrian Prisacariu

Active Vision Lab, University of Oxford, Oxford, UK
{marcelo,costain,victor}@robots.ox.ac.uk
https://code.active.vision

Abstract. We propose a novel method for neural network quantization that casts the neural architecture search problem as one of hyperparameter search to find non-uniform bit distributions throughout the layers of a CNN. We perform the search assuming a Multi-Task Gaussian Processes prior, which splits the problem to multiple tasks, each corresponding to different number of training epochs, and explore the space by sampling those configurations that yield maximum information. We then show that with significantly lower precision in the last layers we achieve a minimal loss of accuracy with appreciable memory savings. We test our findings on the CIFAR10 and ImageNet datasets using the VGG, ResNet and GoogLeNet architectures.

Keywords: Quantization · Bayesian Optimization · Gaussian Process

1 Introduction

The strategy of quantizing neural networks to achieve fast inference has been a popular method of deploying neural networks in compute constrained environments. Its benefits include significant memory savings, improved computational speed, and a decreased cost in the energy needed per inference. Many methods have used this family of strategies, quantizing down to anywhere between 8-bits and 2-bits, with little loss in accuracy [10,30]. It also bears noting that in most of these methods, after quantizing to very low precisions (1 to 5 bits), retraining is necessary to recover accuracy.

Recently, even though the quantization algorithms have significantly improved, they have almost exclusively *implicitly* assumed that the best strategy is to quantize all the layers uniformly with the same precision. However, there are two main reasons to believe otherwise: i) we argue that as it has been interpreted [17] that different layers extract different levels of features, it follows that different layers might require different levels of precision; ii) the idea of quantization as an approximation to the floating point (FP) version of the

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12362, pp. 383–398, 2020.
https://doi.org/10.1007/978-3-030-58520-4_23

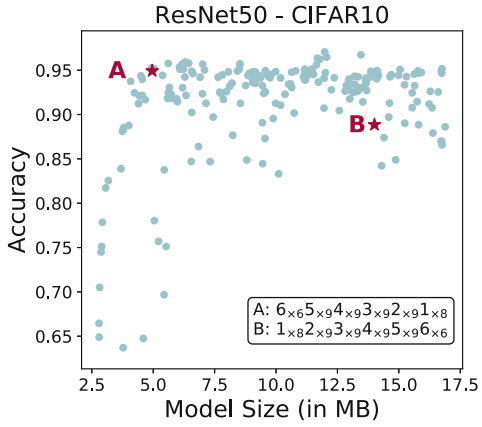


Fig. 1. Gaussian Process prediction for bit distribution in memory vs accuracy plot

network suggests that lower error in the early layers reduces the propagation of errors down the whole network, minimizing any drop in accuracy. We believe that as important as having a good quantization strategy, is to also have a good strategy for the distribution of bits through the network, thereby eliminating any redundant bits. The goal is then to find a configuration in a search space that uses the least amount of bits and achieves the highest accuracy per bit used.

We cast this Neural Architecture Search (NAS) problem into the framework of hyperparameter search, since the bit-width of each layer should ideally be found automatically. As with many NAS approaches, measuring the accuracy of a single configuration can take a considerable amount of time. To mitigate this issue, we propose a two stage approach. First, we map the full search space into a lower dimensional counterpart through a parameterised constraint function, and second, we use a Multi-task Gaussian Process to predict the accuracy at a higher epoch number from lower epoch numbers. This approach allows us to reduce both the complexity of the search space as well as the time required to determine the accuracy of a given configuration. Finally, as our Gaussian Process based approach is suitable for probabalistic inference, we use Bayesian Optimisation (BO) to explore and search the hyperparameter space of variable bit-size configurations.

For the quantization of the network, we use the DSConv method [16]. It achieves high accuracy without significant retraining, meaning the number of epochs needed for full training, and implicitly, the requirement for prediction power, is minimised.

To summarise, our main contributions are as follows:

1. we cast NAS as hyperparameter search, which we apply to the problem of variable bit-size quantization;
2. we reduce the time needed to measure the accuracy of a proposed bit configuration considerably by using multi-task GPs to infer future accuracy from current estimates;
3. we demonstrate performance across a broad range of configurations, described by Bezier curves and Chebyshev series.

The next sections are as follows: Sect. 2 shows previous work on quantization and hyperparameter search. Section 3 elaborates on the methodology used for search, including the constraint, exploration, and sampling procedures. Section 4 shows the results achieved on the CIFAR10 and ImageNet datasets using the networks listed above. Section 5 draws a conclusion and considers insights from the paper.

2 Related Work

Neural Architecture/Hyperparameter Search. One can consider finding bit distributions as a form of model selection [18], given its complexity and the limit on the parameters that it accepts as a solution. Previous methods have predominantly used Reinforcement Learning (RL) and Evolutionary Algorithms (EA) to model search, which is referred to in the literature as Neural Architecture Search. Examples include NASNet [34], MNasNet [26], ReLeq-Net [6], HAQ [29], among others [1,31] for RL and [13,15,24,33] for EA. Our work overlaps with these papers only on the goal of finding an optimal strategy given a search space.

ReLeQ-Net and HAQ, to the best of our knowledge, are the only methods whose aims are to find the optimal bit distribution through different layers of a network, and are therefore the papers that overlap the most with our work. It is notable that both of them use an RL based approach to search for optimal bit distributions. However, HAQ is more focused on hardware specific optimization, whereas both ours and ReLeQ-Net's methods attempt to be Hardware-Agnostic. Recently some work involving Bayesian Optimization (BO) for model architecture selection has been carried out, with systems such as NASBOT [11]. One of the reasons why BO has not been used for model selection has to do with how unclear it is to find a measure of "distance" between two models, which is the main problem that was addressed by NASBOT.

Alternatively, one can see determining bit distribution as finding hyperparameters to be tuned given a model, *i.e.* not different from finding the optimal learning rates or weight decays. Historically, this has been tackled by BO techniques. In neural networks specifically, this was popularized after the work of [21], and followed by others [2,7,22,27]. As a result BO can be considered a natural method for searching for optimum bit distribution configurations.

Quantization. Quantization strategies can be either trained from scratch or derived from a pretrained network. The methods of [4,10,30,32] initialize their networks from scratch. This ensures that there is no initial bias on the values of the parameters, and they can achieve the minimum difference in accuracy when extremely low bit values are used (1-bit/2-bits) - a notable exception being DoReFa-Net [32], which reportedly had slightly better results when quantizing the network starting from a pretrained network. The methods of [8,14,16,28] quantize the network starting from a pretrained network. These methods start with a bias on the values of the parameters, which can limit how much they recover from the lost accuracy. A benefit of these methods though is that they

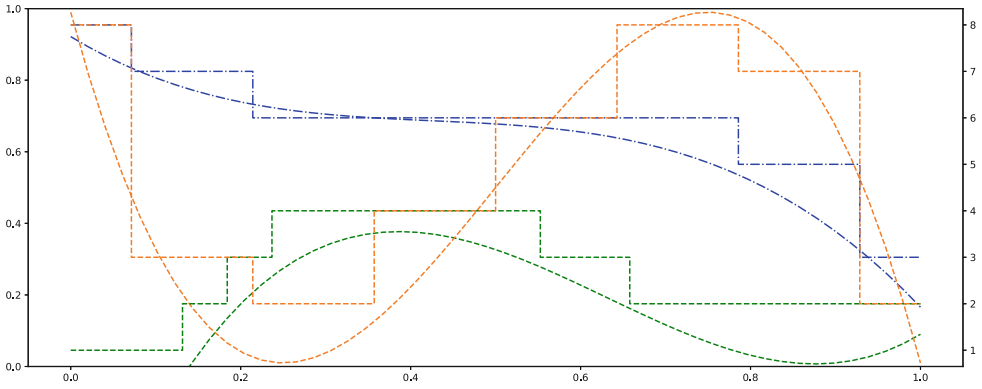


Fig. 2. Three examples of modified Chebyshev functions and their clamped versions. The continuous lines represent the values of the modified Chebyshev functions as function of the layer, which is then converted into bitwidths whose values are represented on the right hand axis. This is for two 8 layer VGG11s and a 20 layer ResNet corresponding to configurations 1) blue: 8,7,6,6,6,6,5,3 2) orange: 8,3,2,4,6,8,7,2 and 3) green: 1,1,1,2,3,4,4,4,4,4,4,3,3,2,2,2,2,2,2,2 (Color figure online)

can be quickly fine-tuned over a few epochs re-achieving state-of-the-art results. These methods are more interesting to us because of their quick deployment cycle. It is worth noting that all of these methods use a uniform distribution of precision, meaning that all layers are quantized to the same number of bits.

3 Method

Our method consists of three parts: constraining, exploring, and sampling the search space. We first constrain the search space by assuming dependence between adjacent bit numbers. We do this by drawing bit distributions from a low-degree Polynomial (in the experiments we use a 2^{nd} degree Bezier curve and a 4^{th} order Chebyshev series). Given a these distributions, we quantize the network using the DSConv [16] method. We explore the space by placing a Gaussian prior over the polynomial parameters, and sampling/retraining a set of hyperparameters that gives the most information about the final payoff function. After exploring, we rank the configurations based on sampling the GP for accuracy, and choose the ones that are the most appropriate for our end-use. Each of these phases will be explained further in this section.

3.1 Constraining the Space

When trying to find the bits, from 1–8, for each layer, the search space will have size of 8^n, where n is the number of layers of the network. For a CNN of 50 layers, the search space will be $2^{150} \approx 10^{45}$, which is a similar size to a game of chess ($\approx 10^{50}$). The size of this space makes an exhaustive search prohibitive.

Our method for constraining the search space relies on the use of parameterised functions. We model a function of degree n with a few parameters, which

describe the search space. We then discretise the function, such that a bit configuration of any layered size network can be sampled from a few parameteres alone.

We use two parameterised functions to illustrate our solution:

- We define the Bezier function $\mathbf{B}(x;\mathbf{w}) = \mathbf{w}^T\phi(x)$ for $x \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d,\ 0 \leq x \leq 1,\ 0 \leq w_i \leq 1\ \forall i \in \{i : i \in \mathbb{N}^+, i <= d\}$, where d is the degree of the polynomial. The vector $\phi(x)$ is the feature vector of the Bezier curve *i.e.* for Linear Bezier $\phi(x) = [1-x,\ x]^T$, for Quadratic Bezier $\phi(x) = [(1-x)^2,\ 2(1-x)x,\ x^2]^T$, *etc.*
- We define the modified Chebysehv function $\mathbf{T}(x;\mathbf{w}) = \frac{(\mathbf{w}-0.5)^T\phi_d(x)+1}{2}$ for $x \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d,\ -1 \leq x \leq 1,\ 0 \leq w_i \leq 1\ \forall i \in \{i : i \in \mathbb{N}^+, i <= d\}$, where d is the degree of the polynomial. The vector $\phi_d(x)$ is defined as $[T_0(x), T_1(x), T_2(x), \ldots, T_{d-1}(x)]^T$ where $T_0(x) = 1, T_1(x) = x$, and $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$

The constraint function, $g(t)$, is then a clamped and rounded version of the chosen polynomial, $p(t)$, such that the bits, b, for each layer generated are between 1 and 8, and $b_i \in \mathbb{N}$. We can define then $g(t) = \lfloor \text{CLAMP}(8p(t)+1), 1, 8) \rceil$, where $\lfloor \cdot \rceil$ is the rounding function, and $\text{CLAMP}(f, a, b) = \min(\max(f, a), b)$.

Figure 2 shows an example of a Chebyshev function and its clamped version. The y-axis in the left indicate the value of the Chebyshev function for different values of x. This is then clamped, rounded, and scaled such that it transforms into a discontinuous line that represents the bit chosen for each layer of a CNN. The bit value is indicated in the y-axis in the right.

By constraining the search space in this way, the minimization problem then shifts as follows:

$$\begin{array}{ll} \textbf{Naïve Approach} & \\ \min\limits_{\mathbf{b}} & \mathcal{L}(t;\mathbf{b}), \mathbf{b} \in \mathbb{N}^n \\ \text{s.t.} & 1 \leq b_i \leq 8,\ \forall i \in \{1, n\} \end{array} \qquad \begin{array}{ll} \textbf{Our Approach} & \\ \min\limits_{\mathbf{w}} & \mathcal{L}(t;\mathbf{w}), \mathbf{w} \in \mathbb{R}^d \\ \text{s.t.} & b_i = g(\frac{i}{n}),\ \forall i \in \{1, n\} \\ & 0 \leq w_j \leq 1,\ \forall j \in \{1, d\} \end{array} \tag{1}$$

where $\mathcal{L}$ is the loss function (to be introduced in Sect. 3.3).

The search then reduces to finding the parameters of the polynomial basis $\mathbf{w}$, which consequently define the bit distributions throughout the layers. The search space is then continuous and compatible with GPs, and significantly reduced to only d dimensions. Using this parameterisation, we are able to easily define a distance metric between configurations to be used when calculating the kernel function and predictive distribution from our Gaussian Process. So, with this setup, the search space can be sufficiently explored in a timely manner.

Quantization Strategy. The method used for quantizing the CNN is DSConv. This is because our aim is to minimize time taken during training, and DSConv has consistently shown good accuracy properties in models, even before retrained.

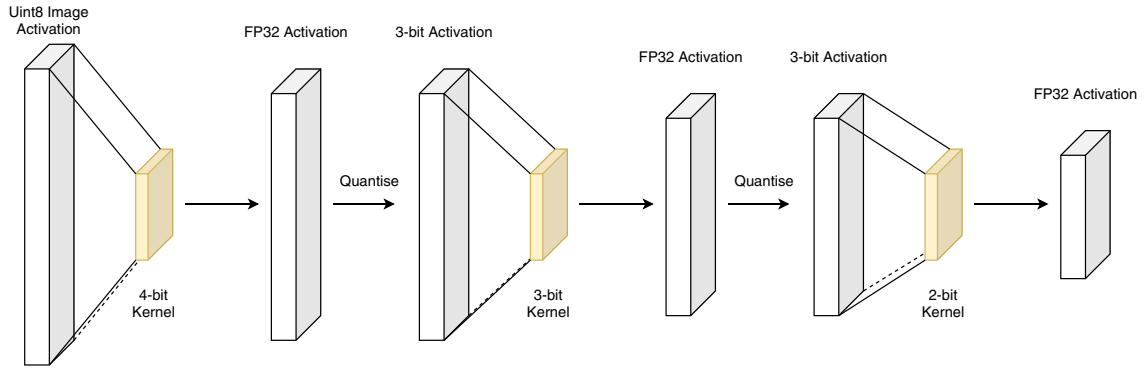


Fig. 3. Quantization given variable bit-widths. Notice that the input is the image, which is a `uint8` tensor (normalization can be dumped into a KDS tensor [16]), so it is not quantized. The quantization of activations is done before the convolution such that the convolution can be done using the same precision.

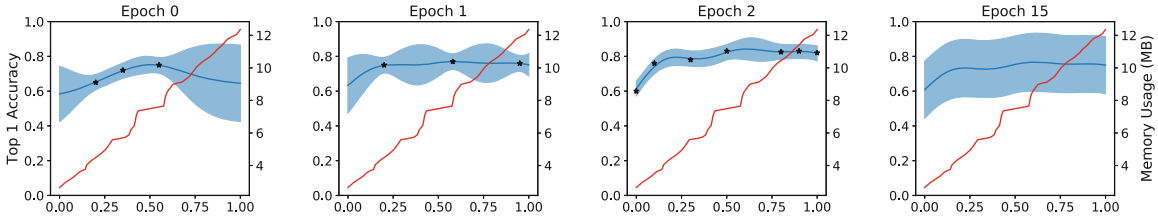


Fig. 4. Multi-task Gaussian Process for inferring accuracy of quantized network. The quantization function is a Bezier Linear with the first parameter set to 0.5 *i.e.* $g(t; w_1) = 0.5 + t(w_1 - 0.5)$. For all figures, the x-axis is the value of w_0, the left y-axis is the accuracy on CIFAR10 of a toy CNN with 10 layers. The right y-axis (red line) shows the model size for a given value of w_0. The epoch correspondences for each task is [0, 1, 2, 15] respectively. After this exploration phase, the decision procedure is run on the predictive distribution of Task 4. (Color figure online)

In this method, both the activations and the weights are quantized, such that fast inference is possible. Each of the weight tensors are divided into blocks of size $B = 32$ depthwise. Each block holds B integers and one FP32 multiplier value. The integers are found by simply scaling each of the block from the original FP32 weight tensor by $\frac{2^{b-1}}{\text{MAX}(w)}$, and then flooring and cropping to range. The FP32 value is calculated by simply minimizing the L2 norm of the block with respect to the original corresponding block: $\xi = \frac{\sum_{i=0}^{B-1} w_i w_{qi}}{\sum_{i=0}^{B-1} w_{qi}^2}$. The activation tensor is quantized similarly, but using Block Floating Point (BFP) format in each of the blocks instead.

In order to take advantage of the low bit multiplication speed, the activation tensor and the weight tensor need to have the same precision. Figure 3 shows how this is done. The activation tensor prior to a convolution layer is set to be quantized to the same bit precision as that layer. The first convolution is not quantized since the input image is already in `uint8` format. Also note that we quantize only the convolutional layers. The Fully Connected layers are all left in the original FP32 precision for training.

3.2 Exploring the Space

Next, we need a way of exploring the space in order to learn the accuracy of the network given a limited set of $\mathbf{w}$ points. We propose a Multi-Task Gaussian Process prior in the neural network, such that each task corresponds to the estimation of the accuracy of the quantized network given $\mathbf{w}$ after a certain number of epochs, *e.g.* task 1 corresponds to 0 epochs, task 2 to 1 epoch, task 3 to 2 epochs, task 4 to 15 epochs. Let there be m tasks, and a prior on f_l, $l \in \{0, m\}$, such that $f_l \sim \mathcal{GP}(\mu(t), k(t, t'))$. We also place a probability distribution $\mathcal{P}(f_0, f_1, ...f_m)$ over different tasks. Let $y_{(t,l)} = f_l(t) + \epsilon(t)$ be the observation at hyperparameter value t for task l, and let $\epsilon(t) \sim \mathcal{N}(0, \sigma^2; t)$ be the observation noise, which is normally distributed. This defines independent Gaussian Likelihoods $y_{(t,l)} \sim \mathcal{N}(f_l, \sigma^2; t)$. From this model, observations $\mathbf{y}$ are drawn, such that $\mathbf{y} = (y_{11}, ..., y_{s1}..., y_{12}, ..., y_{s2}, ..., y_{1m}, ..., y_{sm})$, where y_{il} is the i^{th} observation of the l^{th} task [23].

We used the Intrinsic Correlation Model (ICM) of [5] and [3] for kernel calculation (in our experiments we made use of the squared exponential kernel). We can then define the mean and the correlation between tasks as:

$$\begin{aligned} \langle f_l(x) \rangle &= \mu_l(x) \\ \mathbb{C}(f_l(x), f_{l'}(x')) &= k^f(l, l')k^x(x, x') \end{aligned} \tag{2}$$

where k^f and k^x are positive semi definite functions, corresponding to the correlation between functions and the correlation between inputs respectively. From this it follows that the covariance is $K = K^f \otimes K^x$, where $\otimes$ is the Kronecker product, K^f is the matrix of correlations between the functions and K^x is the matrix of correlations between the inputs. For a new set of data points x_*, the mean prediction can then be calculated using the normal formula for the predictive distribution:

$$\begin{aligned} \overline{\mathbf{f}}(x_*) &= \mu_l(x_*) + (K^{x_*})^T \Sigma^{-1} \mathbf{y} \\ \Sigma &= K + D \otimes I \end{aligned} \tag{3}$$

where D is an $m \times m$ diagonal matrix where the $(l, l)^{th}$ term is σ_l^2.

Figure 4 shows an example of the Multi-Task setting with a 1D Bezier Curve for ease of visualization. Each plot shows the predictive mean and variance for each epoch after 14 data points have been collected, using the exploration algorithm explained in Sect. 3.2. The idea is to predict what is the distribution of the last task given inputs in earlier tasks.

Exploration Phase. In order to make decisions on what parameters to choose, we need to explore the space to predict the accuracy of the last task. The exploration phase for the multitask Gaussian Process follows the Low-Fidelity Search from [23]. The idea is to find the values of x, l such that it gives us maximal information $\mathbb{I}(y_{(x,l)}; f_m |\ \mathbf{y}) = \mathbb{H}(y_{(x,l)} \mid \mathbf{y}) - \mathbb{H}(y_{(x,l)} \mid \mathbf{y}, f_m)$, where $\mathbf{y}$ is the observation history, and (x, l) is the action to be performed. It is important to weight the information by a measure of the cost that it takes to perform that operation.

So the exploration procedure chooses x, l that maximizes $\mathbb{I}(y_{(x,l)}; f_m | \mathbf{y})$ per unit cost. This means that the parameter that has the most information about the payoff function will be picked.

Depending on the dataset and model chosen, the user can favour exploration on one fidelity over the other by decreasing the cost λ of running that particular task. Additionally, we set up a budget on the amount of time in unit cost or number of architectures that we are willing to explore. The Exploration Phase finishes when the Budget has been fully used. After this is finished, the user can run their preferred method of ranking configurations using the posterior of the trained GP.

3.3 Sampling the Space

The naïve goal is to find the highest accuracy per bit possible, which corresponds to finding the minimum of the loss function $\mathcal{L}(t; \mathbf{w}) = -\frac{y_{(t,m)}}{\sum_{i=1}^{n} b_i}$. However, there is a trade-off that must be considered. A model, *e.g.* ResNet20, using a total of 40 bits and achieving 80% accuracy (ratio of 2%/bit) is arguably worse than a model that uses 43 bits and achieves 85% accuracy (ratio of 1.97%/bit). The goal is instead to find a decision procedure that takes into account the regret of not using more bits based on a set of constraints. This relationship should be linear instead of inversely proportional. A better strategy is to assume that using 4-bits for all layers is the lowest uniform quantization scheme without loss of accuracy. Each bit used less than this should be a reward, and each bit used more than this should be a penalty, this is added (or subtracted) to the accuracy to get an "effective accuracy". We then define the effective accuracy as $\mathcal{E}(a, \mathbf{b}, n) = a - \frac{\sum_{i=1}^{n} b_i - 4n}{k}$, where a is the accuracy of the original network, and k is a constant of penalty per bit. Therefore, for $k = 100$ each bit used in addition to the average of 4-bits incurs a penalty of 1% in the effective accuracy. The reverse incurs a reward of 1% in the effective accuracy. The decision procedure becomes then to minimize the negative effective accuracy, $\mathcal{L} = -\mathcal{E}(a, \mathbf{b}, n)$. Once we have enough information about the GPs, we can rank configurations based on their loss in order to pick the most relevant for us.

4 Experiments and Results

We tested our method in a variety of configurations, using versions of the original VGG, ResNet, and GoogLeNet models, altered in order to take CIFAR10, and ImageNet32 as input. For training CIFAR10 and ImageNet32, we used data augmentation by cropping 32×32 image of the 4-pixel padded original. We used a SGD optimiser with momentum of 0.9, and weight decay of 5×10^{-5}. The learning rate started as 10^{-1}, and was divided by 10 after 150 and 250 epochs.

We ran the exploration procedure on ∼65 configurations for each network using the multi-task algorithm outlined above. From these configurations, we could then use the mean of the gaussian to draw estimates of the accuracy

of many different quantization schemes. Using the decision outlined above, we sorted the results by either accuracy, memory, or computational complexity, and selected the points of interest for better visualization and intuition of what the general trend of the found configurations are.

Results on Accuracy Using the CIFAR10 Dataset. Results on CIFAR10 and ablation tests are displayed in Table 1. The configurations are color coded for clarity, with red representing higher bit counts and green representing lower bit counts. These configurations were selected based on the decision procedure outlined above, using the Bezier Linear polynomials.

For comparison, we show 6 configurations of each network: the first and third configurations were picked by our decision procedure; the second and fourth are simply the inverse order of the first and third configurations; the fourth and fifth rows use the traditional uniform distribution of bits for a fair comparison.

It is important to note that the decision to pick these configurations are based on the estimate of the GP rather than on the actual Top-1 results. In order to compare fairly, we also included the Top-1 score and standard deviation from 10 runs after properly training each of them for an additional 30 epochs using the same hyperparameters and optimiser that were used to train their FP32 version. We have also included a delta column which shows the difference between the Top-1 estimate from the GP and the Top-1 after fine-tuning the network. It is remarkable that most of the error in estimation is within 1%, which shows how the GP was able to generalize and interpolate properly as expected.

It can be seen that in general, using more bits in earlier layers yields more accurate, and lighter configurations. The higher accuracy can be explained numerically, since higher bits are used in earlier layers, the error propagation through the network is smaller. The lower memory usage is due to the fact that later layers have a higher number of channels, and therefore using lower precision in those layers yield a massive difference in memory need. For VGG16, the first configuration is both lighter, faster, and more accurate than using 3-bits for all layers. This pattern is repeated for the deeper VGG19 too, where the first configuration yielded superior results to the constant 3-bits for all layers, and also for ResNet18 as well. This "rule of thumb" is somewhat weaker in the GoogLeNet architecture though, even though there is still a clear correlation.

Results on Accuracy Using Chebyshev Series. In order to test robustness of the method in relation to the choice of prior functions, we chose to use a Chebyshev Series of fourth degree, which has a larger search space than the Bezier Linear model. We have tested the model using the CIFAR10 dataset as well, and the results are shown in Table 2.

As it can be seen, the 4^{th} degree introduced more flexibility as to what bit configurations the method is capable of finding. We found that with higher degree of polynomials, the number of architectures to search should also increase. In our experiments, we have searched for $\sim$150 configurations before finding good results. The table shows the expected result that more bits at the beginning compensate for the fewer bits at the end of the network. The ResNet-18 result

Table 1. Results for many configurations on CIFAR10. VGG16 and VGG19 correspond to the architectures introduced in [20]. ResNet18 is the architecture from [9], and the GoogLeNet architecture is from [25]. The Configuration refers to the bit value for each layer of a given model, from earlier layers in the left to later layers in the right. They are color coded for clarity: red for higher bits and green for lower bits. It is important to note that we quantize only the convolutional layers, which means that VGG16 has 13 values, VGG19 has 16 values, ResNet18 has 20 values, and GoogLeNet has 64. Because of its size, the GoogLeNet values were represented by a subscript indicating the number of times that a given bit-width is used. The column "Delta" refers to the difference between the GP estimation of the Top1 accuracy and the actual mean Top1 accuracy ($n = 10$) after properly retraining that particular configuration.

CNN	Configuration (bits per layer)	Top 1 Estimate from GP	Mean Top1	Std	Delta	# Bits	Memory (in MB)
VGG16	32-bit Floating Point	–	93.7%	–	–	–	58.8
	6 555 44 333 22 11	(95.5%)	**93.7%**	0.2%	−1.8%	50	**4.84**
	11 22 333 44 555 6	(91.3%)	**87.7%**	0.2%	−3.6%	50	**9.50**
	7 66 55 44 33 222 1	(92.1%)	93.7%	0.1%	1.6%	50	5.26
	1 222 33 44 55 66 7	(90.1%)	91.5%	0.4%	1.4%	50	10.75
	4444444444444	(93.3%)	93.8%	0.1%	0.5%	52	8.28
	3333333333333	(92.9%)	93.5%	0.2%	0.6%	39	6.44
VGG19	32-bit Floating Point	–	93.9%	–	–	–	80.1
	6 555 4444 333 222 11	(94.4%)	**93.7%**	0.1%	−0.7%	54	**6.95**
	11 222 333 4444 555 6	(91.6%)	**89.6%**	0.4%	−2.0%	54	**12.04**
	5 4444 33333 2222 11	(93.9%)	93.5%	0.1%	−0.4%	46	6.14
	11 2222 33333 4444 5	(90.3%)	88.4%	1.2%	−0.9%	46	10.05
	3333333333333333	(92.9%)	93.4%	0.2%	0.5%	48	8.76
	2222222222222222	(92.1%)	92.2%	0.2%	0.1%	32	6.25
ResNet18	32-bit Floating Point	–	95.4%	–	–	–	44.6
	666 5555 4444 3333 2222 1	(96.3%)	**95.4%**	0.1%	−0.9%	75	**3.72**
	1 2222 3333 4444 5555 666	(95.9%)	**92.9%**	0.3%	−3.0%	75	**8.00**
	44444444 3333333333 22	(95.3%)	95.3%	0.1%	0.0%	60	4.34
	22 3333333333 44444444	(94.5%)	94.2%	0.2%	−0.3%	66	6.08
	33333333333333333333	(94.4%)	95.0%	0.1%	0.6%	60	4.90
	22222222222222222222	(93.1%)	93.3%	0.5%	0.2%	40	3.49
GoogLeNet	32-bit Floating Point	–	95.5%	–	–	–	24.32
	$4_{\times 21}$ $3_{\times 27}$ $2_{\times 16}$	(94.7%)	**95.3%**	0.1%	0.6%	207	**2.35**
	$2_{\times 16}$ $3_{\times 27}$ $4_{\times 21}$	(94.6%)	**94.2%**	0.1%	−0.4%	207	**2.98**
	$6_{\times 8}$ $5_{\times 13}$ $4_{\times 12}$ $3_{\times 13}$ $2_{\times 13}$ $1_{\times 5}$	(95.8%)	95.3%	0.1%	−0.5%	231	2.65
	$1_{\times 5}$ $2_{\times 13}$ $3_{\times 13}$ $4_{\times 12}$ $5_{\times 14}$ $6_{\times 8}$	(94.7%)	90.5%	0.2%	−4.2%	231	3.73
	$3_{\times 64}$	(94.7%)	95.1%	0.1%	0.4%	192	2.68
	$2_{\times 64}$	(93.4%)	93.5%	0.2%	0.1%	127	1.92

resembles the configuration found in Table 4, even though it found a configuration that has more usage of 3-bits, but performs slightly worse. As also expected, when the bit distribution is inverted in the network, it results in both higher memory and lower accuracy.

The same behaviour is found with the VGGs, with the slight difference that as VGG11 is too shallow, it requires more bits to recover the accuracy. VGG16 is considerably deeper, and therefore our algorithm was able to compress it more significantly.

This results shows that our method can be used with a variety of basis. It is worth bearing in mind that the GP processing capability requires the inversion of a matrix, which is proportional to the degree of the polynomial chosen. Therefore our method will only work in a timely manner when using fewer hyperparameters to describe the function.

Table 2. Results of method when using Chebyshev Polynomials of 4^{th} degree.

Method	Network	Bitwidths	Accuracy loss	Memory (as a % of original)
Ours	ResNet18	6 4 33 222 33333333 22	−0.6%	8.0%
		22 33333333 222 33 4 6	−1.2%	11.5%
Ours	VGG11	7 666 7 6 5 4	−0.1%	16.8%
		4 5 6 7 666 7	−0.5%	19.7%
Ours	VGG16	4 3 222 3333 22 11	−0.8%	6.3%
		11 22 3333 222 3 4	−2.4%	8.3%

Results on Network Size. Figure 5 shows the result of the GP-estimated accuracy of different configurations by their model size. The solid purple line links the uniform configurations, starting with all 1s and finishing with all 6s. Therefore, any point that lies above that line is an interesting point, since it gives better accuracy by using the same amount of memory of its uniform counterpart. We have highlighted a number of different interesting configurations with red stars and labelled them from A-M in order to better visualise what each point represent.

As it can be seen in the figure, the choice of bit-usage throughout the network plays an important role in both the accuracy and the memory usage. Even though there is a clear trend that links model size and accuracy, there are a handful of configurations which can perform well on both fronts. It can be seen that, in general, points that are above the purple line are linearly decreasing with bit-usage whereas the ones that are below the purple line are linearly increasing with bit-usage.

The surprising result is that, in the CIFAR10 experiments, even though using uniformly 1-bit for all layers achieves bad results, by just introducing a couple of bits in the first three quarters of the network (such as in points A, C and F), the memory increase is almost negligible, but the accuracy recovery is significant. Adding bits at the end of the network however, achieves the opposite effect. It can also be noticed that points A, C, E, and F, achieve better accuracy than the uniformly 2s configuration whilst using 50% less memory. This is even more evident in point E, in which we used up to 6 bits in the first layers, but still achieved less memory usage due to the usage of 1-bit in the bigger kernels at the end of the network.

In the ImageNet32 experiments, we also see some improvement, albeit less dramatic than the CIFAR10 experiments. The overall massage is still the same, as it can be seen in points H, J and L, for which adding bits in the first layers has achieved good accuracy with small memory increases. It is still noteworthy that even with a dataset as challenging as ImageNet32, due to its substantial decrease of information when compared to the default ImageNet, the GP could find good configurations without needing more datapoints. This shows that this method can be robust to changes in dataset.

Brief Comparison with ReLeQ. One of the other papers that touched in this subject was ReLeQ [6]. As explained in the literature review section, they use a reinforcement learning approach to find optimum bit-distributions over the network. Whilst their quantization methodology varies greatly from that used in this paper, it is worth comparing their results to ours. Their results for the CIFAR10 dataset in two of the networks are shown in Table 3. It can be seen that we achieve similar results for ResNet, though with different mean bits. Since ReLeQ's method does not use the same constraint as our method, it could find more varied solutions. This is a limitation to our method which allows it to find solutions to the network faster whilst using less computational power, but reduces the freedom of choice.

Table 3. Comparison of our accuracy results with ReLeQ's method [6] on CIFAR10. The authors in [6] did not provide their models size in memory, so we estimated using both our and the original author's quantisation scheme to make a fair comparison.

Method	Network	Bitwidths	Accuracy loss	Model size (MB)	
				DSConv	WRPNx1
ReLeQ [6]	ResNet-20	8 22 3 222 3 2 333 222 3 2222 8	0.12%	3.88	3.25
Ours	ResNet-20	666 5555 4444 3333 2222 1	0.1%	3.54	2.91
ReLeQ [6]	VGG-11	8 5 8 5 6666 8	0.17%	6.86	6.61
Ours	VGG-11	777 666 55	0.14%	6.35	5.42
ReLeQ [6]	VGG-16	888 6 8 6 8 6 8 6 8 6 8 6 88	0.1%	13.32	12.54
Ours	VGG16	6 555 44 333 22 11	0.1%	4.62	3.74

Results on ImageNet Using ResNet. For completeness, we have included some of the results found by our algorithm on the more challenging ImageNet dataset [19]. This was trained using an Adam Optimizer [12], with learning rate of 10^{-5}.

Table 4 shows the results. As expected, the same pattern of decreasing precision downstream holds across datasets. Comparing these results with the results from DSConv [16], we can see that a decreasing bit-width throughout the architecture, decreasing from 6 bits to 2, is superior to the "all 4s" and "all 3s".

As with ReLeQ's method, HAQ's method has a weaker constraint on bit distribution, which means it would be able to find configurations that our method would not; However, even with our very strong constraint, we were still able to find configurations that are competitive in memory requirements to those found by HAQ. This shows the strength of the conclusion that later layers require lower precision than earlier layers to maintain the same accuracy.

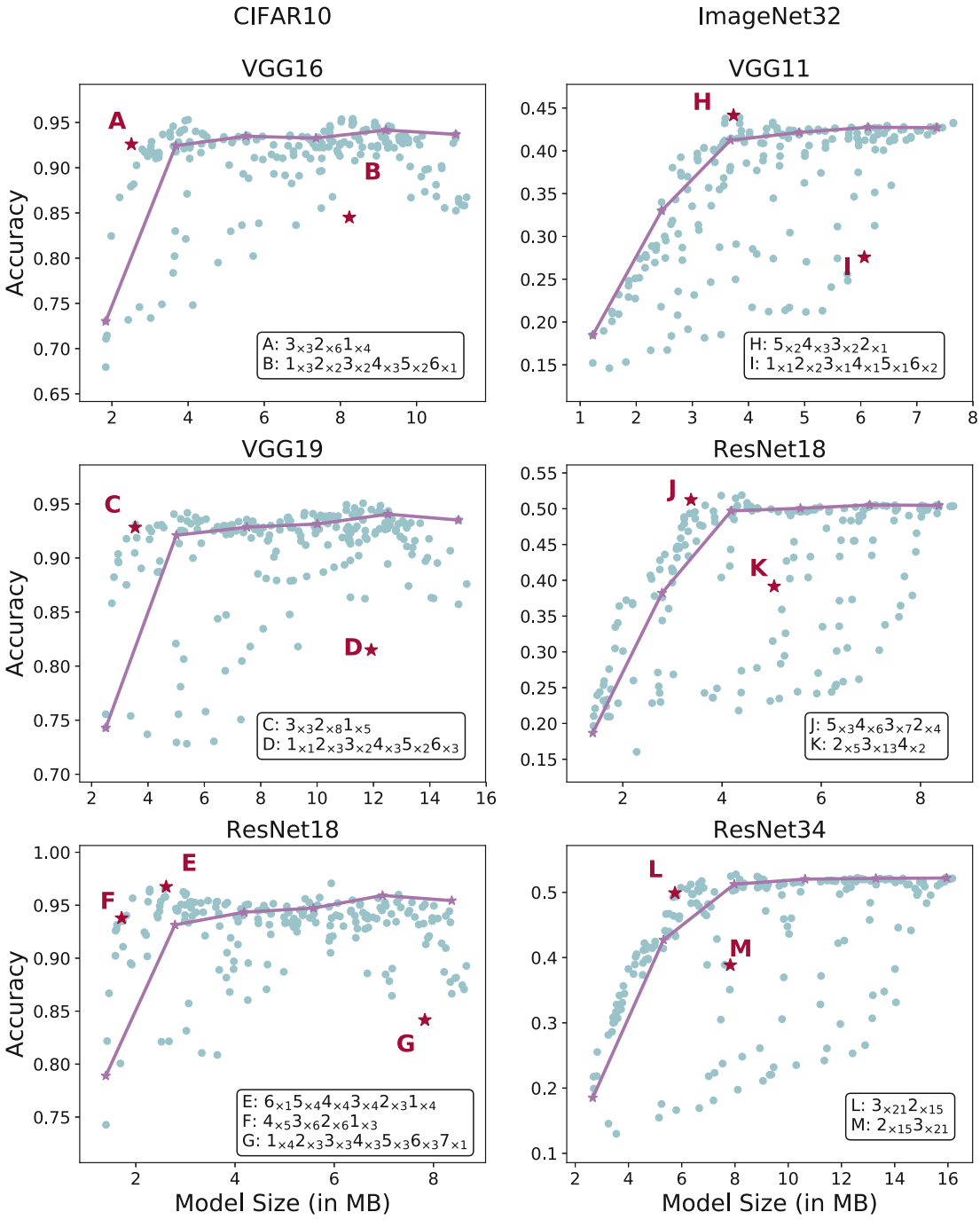


Fig. 5. Scatter plot of the effect on accuracy versus model size of different bit configurations. The left three plots use the CIFAR10 dataset and the right three plots use the ImageNet32 dataset. Note that this is the plot of the estimate as given by the trained GP, and not the actual accuracy given proper training. The solid line refers to the uniform configurations, starting with all 1s and ending with all 6s. Points A-M highlight different configurations as shown in the text boxes. The string of numbers shown refers to the bit size on each layer of the given network. Note that VGG11 has 8 convolutional layers, and therefore points A and B have only 8 numbers. This applies to VGG16 (13 layers), VGG19 (16 layers), ResNet18 (20 layers), and ResNet34 (36 layers) as well.

Table 4. Results of our method using the ImageNet dataset with the ResNet architecture.

Method	# of Layers	Bitwidths	Acc. Loss	Size (MB)
Ours	18	6 ×3 5 ×5 4 ×5 3 ×5 2 ×2	0.2%	**4.89**
DSConv [16]	18	4	0.0%	5.88
DSConv [16]	18	3	0.8%	4.55
Ours	50	6 ×6 5 ×15 4 ×14 3 ×15 2 ×3	0.6%	**11.89**
DSConv [16]	50	4	0.0%	14.54
DSConv [16]	50	3	0.9%	11.74
HAQ [29]	50	*flexible*	0.0%	12.14

5 Conclusion

In this paper, we demonstrate that a uniform distribution over bit-widths throughout a CNN is likely not the most efficient way to quantize a neural network. In order to demonstrate this, we used a Multi-Task Gaussian Process prior over different training epochs, and a Bayesian Optimization exploration procedure based on Information Maximization that estimated the accuracy of different configurations.

We have observed that starting a CNN with higher bit-widths and decreasing precision in later layers yield better accuracy and better memory usage than the traditional uniformly distributed bit-width. This can be interpreted either numerically (as in less error being propagate down the network), or can be interpreted as the functionality of each layer in the network (earlier layers are concerned with feature extraction and later layers are concerned with classification).

Acknowledgements. This research was supported by Intel and the EPSRC, and we thank our colleagues from the Programmable Solutions Group who greatly assisted in this work.

References

1. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167 (2016)
2. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the 30th International Conference on Machine Learning, vol. 28, pp. I–115. JMLR.org (2013)
3. Bonilla, E.V., Chai, K.M., Williams, C.: Multi-task gaussian process prediction. In: Advances in Neural Information Processing Systems, pp. 153–160 (2008)
4. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017. https://doi.org/10.1109/cvpr.2017.574

5. Chai, K.M.: Multi-task learning with Gaussian processes. Ph.D. thesis, The University of Edinburgh (2010)
6. Elthakeb, A.T., Pilligundla, P., Yazdanbakhsh, A., Kinzer, S., Esmaeilzadeh, H.: ReLeQ: a reinforcement learning approach for deep quantization of neural networks. arXiv preprint arXiv:1811.01704 (2018)
7. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems, vol. 28, pp. 2962–2970 (2015)
8. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149 (2015)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
10. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Quantized neural networks: training neural networks with low precision weights and activations. J. Mach. Learn. Res. **18**(1), 6869–6898 (2017)
11. Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., Xing, E.P.: Neural architecture search with Bayesian optimisation and optimal transport. In: Advances in Neural Information Processing Systems, pp. 2016–2025 (2018)
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. Complex Syst. **4**(4), 461–476 (1990)
14. Lin, X., Zhao, C., Pan, W.: Towards accurate binary convolutional neural network. In: Advances in Neural Information Processing Systems, vol. 30, pp. 345–353 (2017)
15. Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K.: Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436 (2017)
16. Nascimento, M.G.D., Fawcett, R., Prisacariu, V.A.: DSConv: efficient convolution operator. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5148–5157 (2019)
17. Olah, C., Mordvintsev, A., Schubert, L.: Feature visualization. Distill (2017). https://doi.org/10.23915/distill.00007. https://distill.pub/2017/feature-visualization
18. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)
19. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
21. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp. 2951–2959 (2012)
22. Snoek, J., et al.: Scalable Bayesian optimization using deep neural networks. In: International Conference on Machine Learning, pp. 2171–2180 (2015)
23. Song, J., Chen, Y., Yue, Y.: A general framework for multi-fidelity Bayesian optimization with Gaussian processes. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 3158–3167 (2019)

24. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
25. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
26. Tan, M., et al.: MnasNet: platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2820–2828 (2019)
27. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD 2013, pp. 847–855. ACM, New York (2013). https://doi.org/10.1145/2487575.2487629
28. Vanhoucke, V., Senior, A., Mao, M.Z.: Improving the speed of neural networks on CPUs. In: Deep Learning and Unsupervised Feature Learning Workshop, NIPS. Citeseer (2011)
29. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: HAQ: hardware-aware automated quantization with mixed precision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8612–8620 (2019)
30. Zhang, D., Yang, J., Ye, D., Hua, G.: LQ-Nets: learned quantization for highly accurate and compact deep neural networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11212, pp. 373–390. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01237-3-23
31. Zhong, Z., Yan, J., Wu, W., Shao, J., Liu, C.L.: Practical block-wise neural network architecture generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2423–2432 (2018)
32. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160 (2016)
33. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)
34. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8697–8710 (2018)

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	Finding non-uniform quantization schemes using multi-task gaussian processes
Publication Status	X Published □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Gennari do Nascimento, Marcelo, Theo W. Costain, and Victor Adrian Prisacariu. "Finding non-uniform quantization schemes using multi-task gaussian processes." In European Conference on Computer Vision, pp. 383-398. Springer, Cham, 2020.

Student Confirmation

Student Name:	Theo W. Costain		
Contribution to the Paper	• Conception of the ideas together with Marcelo Gennari do Nascimento • Designed, and wrote the code together with Marcelo Gennari do Nascimento • Performed the experiments in collaboration with Marcelo Gennari do Nascimento • Co-created all of the tables and diagrams with Marcelo Gennari do Nascimento • Co-wrote the manuscript with Marcelo Gennari do Nascimento		
Signature		Date	18/9/23

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 Victor Adrian Prisacariu		
Supervisor comments	I support the student's assesment of their contributions to this work.		
Signature		Date	18/9/23

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

3.3 Extension to 3D Tasks

As discussed in chapter 2, applying deep networks to dense voxel grid representations is extremely expensive. In tasks where dense voxel grids are used, quantisation represents a meaningful approach to minimising cost in terms of both energy and memory. Whilst the reductions in memory are useful, given general concerns around energy usage and the climate impact of machine learning (ML) [174], widespread deployment of deep models are likely to be increasingly concerned about their energy consumption. As a result, the potentially substantial reductions in energy consumption that quantisation offers, without meaningfully degrading the performance of models, are likely to be of significant importance to industry.

The paper presented above focused entirely on tasks and networks in 2D, as this is the focus of the existing literature as well as the source of methods against which we can compare our work. In the rest of this section, we discuss our further experiments applying the approach from the paper to a variety of 3D tasks. We start with experiments on video classification, followed by experiments on 3D classification and semantic segmentation, and finally experiments on implicit representations of 3D shapes. Lastly, we discuss the realities of using quantisation to improve the efficiency of deep learning on dense 3D grids, including the advantages, disadvantages, and alternatives.

There are very few methods that propose the application of quantisation methods to 3D tasks with the exception of Heinrich et al. [175] (based on [79]), the follow-up work of Paschali et al. [176], and separately Sun et al. [177]. The first two works focus on medical image segmentation, a setting in which the use of dense voxel grids cannot be avoided. The third work, VideoIQ[177], takes a somewhat similar approach to the work we presented in our paper, using non-uniform quantisation schemes, however rather than having different bit-widths in the layers of the network, they use the same bit-width throughout the whole network but use different bit-widths for each frame/snippet [178] in the video. However, to the best of our knowledge, no other

methods propose the use of quantisation for common (non-medical) computer vision tasks, particularly semantic segmentation or implicit representation of 3D shapes.

3.3.1 Video Comprehension

The most natural extension of experiments in the paper to 3D settings is the task of video action recognition, a close analogue to image classification.

	18-layer ResNet		18-layer ResNet3D	
Layer	output size	dimensions	output size	dimensions
1	112×112	7×7, 64, stride 2 3×3 max pool, stride 2		3×7×7, 64, stride 1×2×2
2	56×56	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times2$	L×56×56	$\begin{bmatrix} 3\times3\times3,\ 64 \\ 3\times3\times3,\ 64 \end{bmatrix}\times2$
3	28×28	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times2$	$\frac{L}{2}$×28×28	$\begin{bmatrix} 3\times3\times3,\ 128 \\ 3\times3\times3,\ 128 \end{bmatrix}\times2$
4	14×14	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}\times2$	$\frac{L}{4}$×14×14	$\begin{bmatrix} 3\times3\times3,\ 256 \\ 3\times3\times3,\ 256 \end{bmatrix}\times2$
5	7×7	$\begin{bmatrix} 3\times3,\ 512 \\ 3\times3,\ 512 \end{bmatrix}\times2$	$\frac{L}{8}$×7×7	$\begin{bmatrix} 3\times3\times3,\ 512 \\ 3\times3\times3,\ 512 \end{bmatrix}\times2$
	1×1	average pool, fc, softmax	1×1×1	st pool, fc, softmax

Table 3.1: A comparison of the architectures of 2D and 3D ResNet structures from He et al. [29] and Tran et al. [11] respectively. "st pool" means spatiotemporal pooling, L refers to video clip length.

Whilst an array of architectures for video comprehension have been proposed [179; 180; 181; 182; 183; 184], very few propose a fully 3D structure. One example of a filly 3D architecture comes from the work of Tran et al. [11], who proposed a number of different architectures for video comprehension based on the residual architecture from [29]. For our experiments, we focus on the fully 3D convolution based architecture (R3D), as the only method that consists solely of 3D convolutions, rather than the spatio-temporally separable convolutions also proposed in their work. Their fully 3D convolution based method simply lifts the well known ResNet-18 architecture into 3D by adding extra dimensions to each layer (see Table 3.1). We conduct our evaluations against the Kinetics-400 [185] dataset, a popular and large video action recognition dataset common in the literature.

We show the results of applying non-uniform quantisation schemes to the R3D network in Table 3.2, where we report the *clip* Top1 accuracy. As with the 2D

Table 3.2: Video action recognition experiments, reporting Top1 accuracy. The exceptionally poor performance when quantised to 1 bit hampers non-uniform schemes' performance compared to 2D and later 3D settings.

Configuration (bits per layer)	**Top1**	**Δ**	**# Bits**	**Memory (in Mb)**
32-bit Floating Point	51.5%	–	–	1060
88888888888888888	51.6%	0.1	136	265
66 555 4444 3333 2222	50.1%	-1.4	63	77.2
5555555 4444444 333	51.2%	-0.3	72	113
44444444444444444	51.3%	-0.2	68	133
4444444444 3333333	51.0%	-0.5	61	102
44444 333333 2222 11	44.22%	-7.28	48	57.3
33333333333 444444	50.8%	-0.7	57	128
33333333333333333	50.4%	-1.1	51	99.5
22222222222222222	46.4%	-5.1	34	66.3
2222 3333 4444 555 66	49.4%	-2.1	63	172
11111111111111111	5.0%	-46.5	17	33.2

experiments from the paper, we find that non-uniform schemes are able to maintain the same overall accuracy as uniform schemes, whilst further reducing the overall memory footprint of the network. However, we note that the effect is much less pronounced than in the 2D case. We suggest this may be due to the unusually poor performance when the network is entirely quantised to 1 bit uniformly, meaning that quantising much below 4 bits significantly degrades performance below what we might expect from our other experiments. By way of comparison, in the 2D case, quantising the weights of ResNet-18 down to 1 bit uniformly reduces performance from 95.4% down to 61.2%. This drop, though steep, is both perceptually and ratio-wise a smaller drop than in the 3D case. This suggests that the final few layers of the network might be the most significant to the overall performance, which is to be expected given the last layers have the largest temporal scope, and therefore are vital in aggregating information over the temporal dimension. As a result, it appears that given the inability to quantise later layers as agressively than in the

3D case, it may be that for this model, the optimal scheme is barely non-uniform, with only the last few layers quantised with a lower bit-width than the other layers.

3.3.2 3D Semantic Tasks

Classification and semantic segmentation are extremely common tasks in 3D understanding. In 3D settings, the increased cost of capturing data (compared to 2D), means that classification tasks are overwhelmingly limited to the object level, rather than scene level. The smaller scale that object level 3D tasks require means that low resolution 3D grids (*i.e.* 32^3 or 64^3) are likely sufficient to capture object level details, whereas the same resolutions would be far too small to capture the necessary details of scene scale data. Despite this, only a small number of methods used 3D convolutions on voxel grids for these tasks. Some early methods include VoxNet [186], which consists of two 3D convolutional layers (reducing from 32^3 to 6^3) followed by two linear layers, and ShapeNet [12], which consists of 3 convolutional layers ($32^3 \rightarrow 2^3$) and three linear layers. We apply our method to these two networks and evaluate them on the ModelNet40 [12] dataset. The results of quantising these methods are shown in Table 3.3, where we use the Top1 accuracy metric.

As with the 2D experiments, we see the same substantial reduction in memory requirements with minimal reduction in accuracy, and the further reduction gained from non-uniform schemes. However, the models used in these experiments are extremely small, and likewise the dataset. To better explore the method in these settings, we turn to larger models trained on larger and more challenging datasets.

Larger 3D tasks requiring deeper networks are more commonly applied to semantic segmentation tasks, however very few methods operate directly on 3D voxel grids. One of the few examples that do is 3DMV [187]. They propose a method that integrates a 3D occupancy grid, alongside RGB-D frames (using a back projection module, to project the 2D features into the 3D volume), to perform semantic segmentation. In our experiments, we focus only on the purely 3D pipeline. Importantly, rather than ingesting scenes whole, they apply a sliding window method over $31 \times 31 \times 62$ volumes, classifying only the center column

of a given volume. Like the original work, we evaluate our experiments on the ScanNet [20] dataset, and the results are shown in Table 3.4. For these experiments, we use the same Top1 accuracy metric as before.

Table 3.3: Experiments on ModelNet40 with simple networks. Voxnet is the network from Maturana and Scherer [186]. ShapeNets is the network from Wu et al. [12]. We report Top1 accuracy (higher is better). Even in these simple networks, non-uniform schemes are able to provide meaningful improvement.

CNN	Configuration (bits per layer)	Top1	Δ	# Bits	Memory (in Kb)
VoxNet	32-bit Floating Point	81.7%	–	–	1,013
	88	81.7%	0.0%	16	253
	77	81.7%	0.0%	14	222
	7 1	80.2%	-1.5%	8	56
	44	81.6%	-0.1%	8	126
	5 1	81.4%	-0.3%	6	71
	11	80.2%	-1.5%	2	32
ShapeNets	32-bit Floating Point	78.8%	–	–	198,824
	888	79.0%	0.2%	24	49,706
	777	79.1%	0.3%	21	43,493
	6 4 2	78.6%	-0.2%	12	14,388
	444	79.1%	0.3%	12	24,852
	4 3 1	77.8%	-1.0%	8	8,164
	111	77.1%	-0.7%	3	6,213

Whilst we observe the same expected pattern here as in previous experiments, we suggest this is not necessarily as obvious as it would be in the case of image/video input. In the case of RGB image or video input, any pixel/voxel takes a value in $\mathbb{R}^{3\times255}$, whereas in our 3D experiments (both classification and segmentation), the input space is much smaller ($\{0, 1\}$ for classification and $\{-1, 0, 1\}$ for segmentation). Accordingly, we might have expected to see less need for higher bit-widths in earlier layers of the network, and more need in later layers. This is somewhat shown by the configurations 444 555 666 77 and 888 777 666 55, where there is no difference in performance regardless of the direction of bit-width, but this is counterbalanced against 111 222 333 44 *vs.* 44 333 222 111, where the latter achieves better performance with lower memory.

Table 3.4: 3DMV [187] Results. We report Top1 accuracy (higher is better), across a variety of configurations

Configuration (bits per layer)	OA	Δ	# Bits	Memory (in Mb)
32-bit Floating Point	71.8%	–	–	13.2
88888888888	71.7 %	-0.1	88	3.31
888 777 666 55	71.7 %	-0.1	73	2.57
77 333 222 111	69.5 %	-2.3	41	1.14
444 555 666 77	71.7 %	-0.1	59	2.39
555 444 333 22	71.2 %	-0.6	40	1.33
44444444444	71.1 %	-0.7	44	1.66
44 333 222 111	69.2 %	-2.6	26	0.897
444 33333333	71.1 %	-0.7	36	1.25
666 555 444 33	71.6 %	-0.2	51	1.75
333 444 555 66	71.0 %	-0.8	48	1.98
33333333333	71.1 %	-0.7	33	1.24
222 333333 44	70.1 %	-1.1	32	1.24
22222222222	69.6 %	-2.2	22	0.828
111 222 333 44	65.6 %	-6.2	26	1.15
11111111111	47.3 %	-24.5	11	0.414

3.3.3 Implicit Representations

A more recent evolution in the literature, implicit representationss (IRs) aim to express the shape and structure of a 3D surface as a learnt function. We have discussed these in more detail in chapter 2, and will again in part II, however we give a brief overview here to aid the reader.

Generally, IRs are composed from two parts: an encoder that captures and transforms shape information from a conventional representation (*e.g.* point-cloud, mesh, voxel grid *etc.*) into a compact vector or set of vectors, and an implicit function (almost universally an multi-layer perceptron (MLP)) that takes both the encoded vector and a query position in space and returns some metric description of the represented shape, typically occupancy probability or signed/unsigned distance to the nearest surface.

For our experiments, we use the method proposed by Chibane et al. [156], NDF, as this method represents the state-of-the-art (SOTA) performance on room scale neural implicit representation (NIR) performance. Their work uses a hierarchical

multi-resolution convolutional neural network (CNN) to encode the input shape at multiple feature resolutions, and then in the decoder, interpolates these features at the query position to condition the implicit function. Unlike most prior methods, their method learns unsigned distance functions, which allow their method to better represent non-watertight surfaces, removes the need for extensive pre-processing on the training data to ensure watertight shapes, and eliminates the overly thick walls learnt by other methods.

The results of applying non-uniform quantisation schemes to the encoder network are presented in Table 3.5. The metric used is the L2 Chamfer Distance, for which a lower number is better.

Table 3.5: Implicit representation experiments, CL2 (L2 norm chamfer distance, lower is better) is the average of ($n = 3$) runs. The previous pattern of decreasing bitwidths through the layers is reversed here, with more bits at the start and fewer at the end appearing optimal.

Configuration (bits per layer)	**CL2 (↓, 10^{-4})**	**Δ**	**# Bits**	**Memory (in Mb)**
32-bit Floating Point	1.7048	–	–	84.5
88888888888	1.7169	-0.012	88	21.1
888 777 666 55	1.6583	0.047	73	15.4
777 666 555 44	1.7191	-0.014	62	12.8
666 555 444 33	1.7188	-0.014	51	10.1
555 666 777 88	1.7093	-0.005	70	18.9
55555555555	1.7841	-0.079	55	13.2
555 444 333 22	1.7364	-0.032	40	7.51
444 555 666 77	1.7112	-0.006	59	16.3
444444 88888	1.6935	0.011	64	19.4
222 444 666 88	1.6688	0.036	52	16.7
222 333 444 55	1.7332	-0.028	37	11.0
22222222222	1.7621	-0.057	22	5.28
111 222 333 44	1.7382	-0.033	26	8.34
11111111111	1.8671	-0.162	11	2.64

There is some volatility inherent in the results, given that the evaluation procedure for their method is unavoidably non-deterministic, however, the same overall trend can just be seen in the results. Notably, because of the multi-resolution/hierarchical nature of the shape encodings, the optimal direction of

bit-width change is reversed, with fewer bits needed at earlier scales representing coarse features, and more bits needed at later scales which represent more fine-grained details. Of note, because we quantise both the weights of layers as well as the activations, we benefit from *free* compression of the encodings as well as the encoder network. Given a major reason for the importance of IR is efficient representation of shapes, further increasing this efficiency as a side effect of improving the encoder's efficiency is extremely fortuitous.

3.3.4 Discussion

Whilst our method performs well in 3D settings, except in cases where full 3D convolutions on dense voxel grids are the only available choice, quantisation is likely not the best tool to improve efficiency, and other approaches to processing the data are much more likely to provide substantial efficiency improvements, though in many cases it may be possible to apply quantisation on top of these other alternative approaches. Practically, quantisation can only provide linear improvements to efficiency, whereas alternative methods and representations are able to mitigate the otherwise cubic growth.

Accordingly, whilst quantisation represents a meaningful method to reduce energy and memory requirements, by and large the literature has taken a different approach, favouring the use of differing representations that rely on different operations to 3D convolutions, rather than schemes that directly optimise 3D convolutions.

Despite this, it is quite possible that different quantization scheme than what we use in our experiments could be applied alongside our bayesian optimization (BO) based neural architecture search (NAS) to improve the efficiency of methods that do not rely on convolutional operations, but we suggest that this is beyond the scope of this chapter.

3.4 Conclusion

In this chapter we presented our work *Finding Non-Uniform Quantization Schemes using Multi-Task Gaussian Processes*, a method for automatically discovering optimal

non-uniform quantisation schemes in deep networks. By describing the quantisation scheme of a network in terms of a low dimensional parametric function, we are able to cast NAS as hyperparameter search. This allows the use of a multi-task Gaussian process (GP) to efficiently explore the configuration space by choosing samples that maximise information gain, before using the trained GP to find optimal (subject to a chosen metric) configurations.

Our method provides an effective and efficient approach to further improving the benefits of quantisation, agnostic to the particular method used. We find that by gradually decreasing the fidelity of successive layers in a deep network, it is possible to reduce the memory footprint of a deep network beyond what is possible with uniform quantisation schemes alone, and our extension experiments demonstrate that techniques can be extended to 3D tasks.

4

Approximating Continuous Convolutions for Deep Network Compression

Contents

4.1 Introduction

Whilst quantisation presents a viable approach to reducing the memory requirements of deep networks, in order to achieve the reduction in computational and energy requirements, non-standard hardware is usually needed. Although the general availability of this hardware is increasing, for example recent generations of NVIDIA graphical processing units (GPUs) provide support for operations on 1, 4, and 8 bit integers, these hardware operations are less common in compute constrained and edge settings. As previously mentioned [15; 76] (Section 1.1.2), in compute constrained (edge) environments minimising model size, thereby increasing cache occupancy of the model itself, will likely see better efficiency benefits than reducing

the total number of operations, or using lower energy operations.

To this end, we present our work on deep network compression: *Approximating Continuous Convolutions for Deep Network Compression*. We propose to compress the weight kernels of deep networks by approximating the values of the weights using functional approximations like the Chebyshev and Cosine series. Considering the values in discrete convolution as point samples from continuous functions over space, we are able to compute an approximation to the piecewise linear function representing the original values, in the form of our kernel functions. Unlike other similar compression methods, our approach does not require training from scratch and can be applied as a post-processing step after training, needing only a small amount of fine-tuning. Further, we demonstrate our method is complimentary to quantisation, which can be applied to the parameters of our approximating kernel functions, allowing for further reduction in memory size.

Whilst the work presented in the paper is primarily concerned with 2D networks and tasks, as we touch upon in the coming method section, the underlying mathematics are trivially extensible to 3 dimensions and higher. However given a lack of literature with which to compare, we did not feel an exploration of 3D methods and network would benefit the paper.

4.1.1 Contributions

- We propose ApproxConv, an approach to compress the weights of a deep network using approximation.
- We present a framework that allows our approach to be applied as a post processing step avoiding the need to train the network from scratch.
- Through experimental evaluation, we demonstrate the effectiveness of our approach, validating its formulation, and further, we demonstrate its compatibility with other approaches such as quantisation that can be leveraged together to further decrease the memory requirements.

Approximating Continuous Convolutions for Deep Network Compression

Theo W. Costain
costain@robots.ox.ac.uk
Victor A. Prisacariu
victor@robots.ox.ac.uk

Active Vision Laboratory
University of Oxford
https://code.active.vision

Abstract

We present ApproxConv, a novel method for compressing the layers of a convolutional neural network. Reframing conventional discrete convolution as continuous convolution of parametrised functions over space, we use functional approximations to capture the essential structures of CNN filters with fewer parameters than conventional operations. Our method is able to reduce the size of trained CNN layers requiring only a small amount of fine-tuning. We show that our method is able to compress existing deep network models by half whilst losing only 1.86% accuracy. Further, we demonstrate that our method is compatible with other compression methods like quantisation allowing for further reductions in model size.

1 Introduction

Deep vision models have demonstrated outstanding performance across a range of problems from semantic segmentation of microscopy images[33], to object detection for nature conservation efforts[31]. In many cases, there is strong desire to apply these methods in compute constrained environments, *i.e.* mobile devices, and as a result the desire for space and compute efficient CNN models is as strong as ever.

Despite this, recent works[4, 7] have seen explosive growth in parameter count for highly performant models, and there is no reason to expect this trend to abate soon[3].

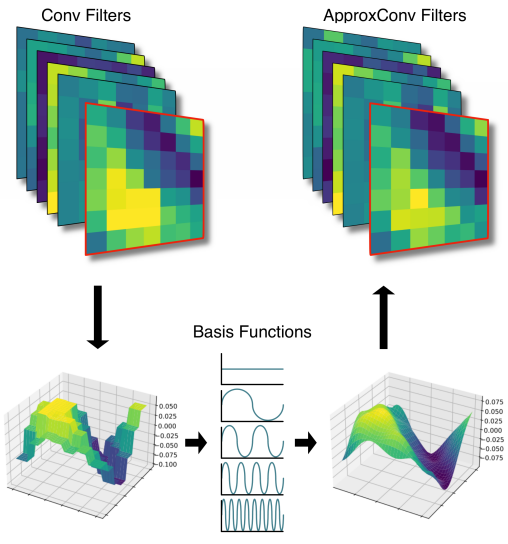


Figure 1: Our method compresses network filter weights, by approximating them using cosine and Chebyshev series.

Approaches to reduce the size and compute requirements of deep models have been the focus of significant attention. A number of works have proposed models designed explicitly for efficiency[19, 21, 44], and other efforts to improve the efficiency of deep models have included methods such as low-rank factorisation[23], knowledge distillation[18], quantisation[45], and network pruning[11, 25].

In this work, we propose to use functional approximations to reduce the number of parameters required to fully describe a CNN layer's filters. We re-cast conventional discrete convolution on grids as discrete convolution on parametrised continuous functions of space. In this way, we are able to replace the kernel weight tensor with an approximating function that is able to exploit the structures commonly found in CNN filters. Specifically, we make use of cosine and Chebyshev series that are able to preserve the low frequency information that allows the networks to function, whilst minimising the effects of high frequency information that are often exploited by adversarial attacks[46]. Our proposed method is particularly effective for larger kernel sizes, which despite falling out of favour in earlier deep CNN work[16, 37], have recently seen a resurgence[26].

In our experiments, we show that our method is able to significantly reduce the number of parameters in a variety of models, and unlike comparable methods[35, 41] is able to achieve this with only a limited amount of fine-tuning. Further, we demonstrate that our approach is complementary to other network size reduction methods such as quantisation. Complimenting our method with other approaches, we are able to increase the overall reduction in total model size with minimal further impact on accuracy.

In summary, the contributions of this work are as follows:

- A novel method to approximate kernel weights using functional approximations based on cosine and Chebyshev series.
- A framework that allows this method to be used without completely retraining models from scratch.

In the rest of the paper, we discuss works related and relevant to our method(Section 2), our methodology (Section 3), our experiments and results (Section 4), before a final a discussion and the conclusion of the work (Section 5).

2 Related Work

A number of works[20, 21, 44] have attempted to reduce the size of models by designing networks with efficiency in mind, such as MobileNets[19] which introduced depthwise convolutions.

Quantisation methods[8, 15, 30, 38, 45] have seen great success in reducing the memory footprints of deep models. Quantising 32-bit floating point numbers down to 8, 4 or even as low as 1 bit, these methods are able to drastically reduce the memory footprint of deep models. Other works[9, 12, 40] demonstrated that non-uniform quantisation schemes, where different layers of the network can be quantised to different numbers of bits, can further reduce the size of the model without compromising accuracy. These non-uniform schemes influence some of our experiments, where we employ a similar approach.

Knowledge distillation methods[18, 29, 32, 42], typically use larger (teacher) models to guide the training of smaller (student) ones, often allowing the student models to achieve performance parity despite their smaller size.

Network pruning methods[1, 14, 25, 27] attempt to discover redundant parameters in deep networks. Having identified these parameters, they can then be removed either individually, yielding sparse networks, or as a group, sometimes removing whole filters or channels. Seminal works, like [11, 28], demonstrate that it can be possible to prune networks before training, realising the size reduction benefits for training as well as inference.

Low rank factorisation methods[6, 23, 34, 43] reduce both the total number of parameters in CNN layers, as well as the number of FLOPs required to compute the outputs of the layer. Using a variety of methods, these approaches decompose the weight tensors of layers into a series of smaller tensors. These smaller tensors can then be convolved with the inputs sequentially (in a fashion similar to separable filters from classical methods), yielding an approximation of the original output.

Both methods learn parametrised functions over space, however rather than generalised approximations, they make use of gaussian and gaussian derivative functions. Zamora et al. [41] learns only simple classical image filters (*e.g.* Sobel filter) and Saldanha et al. [35] attempt to replicate Jacobsen et al. [22] using fewer parameters. Using fractional calculus, they are able to efficiently parametrise a family of gaussian and gaussian-derivative functions using only 3[35] and 6[41] parameters per filter. While this approach allows for significant parameter savings, in comparison to our proposed method, the shape of kernels that can be learnt is substantially limited. As a result, whereas their methods require training from scratch, our approach can be applied to pre-trained models, .

In 3D settings, dense regular grid structures are too memory intensive and so receive little attention. Instead, much of the effort focuses on irregular data structures like point-clouds or meshes. In these settings it becomes desirable to have a kernel function that can vary continuously over 3D space. Similar to our method, SplineCNN[10] uses b-Splines to generate a continuous function over space. FlexConv[13] learns a kernel weight function based on the distance between sampled point locations and a hyperplane. Simonovsky and Komodakis [36] use shared MLPs with edge information as input to define the kernel weight functions.

3 Method

We begin by covering the general form of continuous convolution and how this formulation can allow for network compression. Next we outline how we come to our choice of approximation functions, before finally discussing how we apply our method to pre-trained networks.

3.1 Continuous Convolution

In general, discrete n-dimensional convolution can be expressed in the form

$$x'_{ij} = \sum_{a=-k}^{k} \sum_{j=-k}^{k} x_{(i-a)(j-b)} \times w_{(i-a)(j-b)} \tag{1}$$

where k is the size of the convolutional kernel, x the input features, and w, the kernel. If we replace the discrete grid in the above equation with set of locations sampled from a continuous function over space, we can re-write the above equation as

$$x'(\mathbf{p}_i) = \sum_{p_j \in \mathcal{N}_k(\mathbf{p}_i)} x(\mathbf{p}_j) \times w(\mathbf{p}_j) \tag{2}$$

where $\mathbf{p} \in \mathbb{R}^n$, and $\mathbf{p}$ is drawn from $\mathbf{P} \in \mathbb{R}^{n \times l}$, $x : \mathbb{R}^n \mapsto \mathbb{R}^d$, and $w : \mathbb{R}^n \mapsto \mathbb{R}^d$. Further, $\mathcal{N}_k(\mathbf{p}_i)$ represents a function that returns the k neighbouring points of $\mathbf{p}_i$ including $\mathbf{p}_i$ itself.

To recover simple 2D convolution with a 3×3 kernel, we can simply let p be the set of 2-vectors $[(-1,-1),(-1,0),(-1,1),\cdots]$.

This formulation has two key advantages: i) The same convolutional operation can be used on both regular nD grids, and on arbitrary nD structures (*e.g.* point-clouds or meshes) over which we can define a neighbourhood function $\mathcal{N}$; ii) Replacing the tensor w with a carefully chosen function $w(\cdot)$, we can reduce the memory requirements of 2D convolutional layers by exploiting the structures present in the tensors.

3.2 Network Compression

Our work is concerned with the second of the two advantages above: that a judicious choice of $w(\cdot)$ can permit parameter savings. As has been shown by other works[25, 28] deep networks often contain significant amounts of redundant information, and often filters that contain low-frequency information. Accordingly, it is often possible to substantially reduce the size of a model by removing redundant parameters[14, 17] or preserving only *relatively* low-frequency structures[35].

The choice of function $w(\cdot)$ is primarily constrained by the number of parameters it requires. As discussed previously, methods like FracSRF[35] and Fractional Filters[41] choose efficiently parametrised gaussian and gaussian-derivatives as $w(\cdot)$. But these functions are not able to approximate arbitrary functions.

Separate to these approaches, the field of functional approximation has been of interest for over a century[39], motivated by applications ranging from solving PDEs to numerical integration. As a result a significant body of work is concerned with using various orthogonal basis functions for approximations. From this corpus, we use two common families of functions to allow our method to learn arbitrary kernel functions: cosine and Chebyshev series.

3.3 Cosine and Chebyshev Series

Cosine and Chebyshev series are able to approximate any periodic function that satisfies specific boundary conditions and the Dirichlet conditions. These conditions require that the function: i) is absolutely integrable; ii) is of bounded variation; iii) has a finite number of non-infinite discontinuities. As any discrete weight kernel can be fully described by a piecewise linear function, it is trivial to demonstrate[1] that this piecewise function satisfies the above conditions. Further, the requirement that the function be periodic (in the case of the cosine series) can be trivially satisfied by repeating the function at the boundaries, although careful choice of boundary conditions and approximation interval is necessary.

Although the Fourier series is generally more well known than the cosine series, extensions of the Fourier series to higher dimensions introduces a number of "cross terms" that grow in complexity as the number of dimensions increases. An alternative is to use the sine or cosine series, which are equivalent to the fourier series under certain conditions. Specifically, for an *even* function, the coefficients of the sine terms go to zero leaving only cosine terms, and vice versa for *odd* functions. By shifting the center of our approximation interval, it is possible to coerce the function to behave as an even or odd function. The advantages of the sine and cosine forms is that their extensions into higher dimensions are trivial and, for

[1] We leave a complete proof as an exercise for the reader

$\mathbf{x} \in \mathbb{R}^n$ with N harmonics, take the following (for cosine) form

$$\hat{w}(\mathbf{x}) = \sum_{i_0=0}^{N} \cdots \sum_{i_n=0}^{N} a_{i_0 \cdots i_n} \cos(i_0 x) \cdots \cos(i_n y) \tag{3}$$

The choice of cosine over sine is motivated by concerns over boundary conditions. Specifically, for cosine series the boundary conditions require/enforce symmetric periodic repetition which minimises the potential for discontinuity at the boundary.

In our experiments, we also make use of the Chebyshev series as it is closely related to the cosine series through the definition that $T_n(\cos(x)) = \cos(nx)$, which conveniently yields a very similar expression to eq. (3)

$$\hat{w}(\mathbf{x}) = \sum_{i_0=0}^{N} \cdots \sum_{i_n=0}^{N} a_{i_0 \cdots i_n} T_{i_0}(x_0) \cdots T_{i_n}(x_M) \tag{4}$$

where T_n is the n^{th} Chebyshev polynomial of the first kind, and $\mathbf{x} \in \mathbb{R}^M$. The Chebyshev series can have faster "convergence" (*i.e.* better approximation with fewer terms) than the trigonometric series in eq. (3).

To achieve compression for a $C_{\text{out}}, C_{\text{in}}, K \times K$ kernel, we choose $\hat{w}$ to be a 2D cosine or Chebyshev series with N "harmonics"or "orders", where $N \leq K$. Replacing a 3×3 kernel, with a series 2 harmonics leads to an over 50% reduction in parameters. Our experiment show however, that 2 harmonics *can* be insufficient to properly approximate a 3×3 kernel. However, for 7×7 kernels, which have seen a resurgence in recent work[26], the increased number of harmonics permitted whilst still maintaining a reduction in parameters, can allow for much better compression depending on the complexity of the filters in question. In our experiments, we refer to replacing the kernel weights with a cosine series as **CosConv**, and **ChebConv** when using a Chebyshev series.

Whilst it is not necessary to use the same number of harmonics in the x and y directions, doing so significantly increases the hyperparameter space to be evaluated for our method. As we are not aware of any popular or common methods that make use of anisotropic kernels, we do not investigate differing spatial harmonics in this work. However, with an appropriate search method, anisotropic harmonics might permit further increases in memory savings.

Also not investigated is another possibility for reducing parameter count using our method, through variable harmonics for each filter. Although, as above, this is essentially a problem of hyperparameter/architecture search and is outside the scope of this work.

3.4 Finding Weights

Depending on the number of harmonics used in the approximating function, there may exist a closed form solution to find the approximation. In the case of the cosine series, this takes the form of the discrete cosine transform. Whilst a closed form solution exists, as we aim to use fewer harmonics than kernel parameters, the solution is overdefined. In our early experiments we found that the solutions found using a closed form solution performed worse during the subsequent fine tuning than those found using simple iterative gradient descent, and as a result we chose to use the gradient descent method to find the values.

$$\mathcal{L} = \frac{1}{K^2} \sum_{i,j=0}^{K} (w_{ij} - \hat{w}(\mathbf{p}_{ij}))^2 \tag{5}$$

In conventional numerical approximation, the choice of "sample points" $\mathbf{p}_{ij}$ is extremely important[39]. Normally, performing approximation using Chebyshev polynomials requires careful choice of sample points[2] to avoid Runge's phenomenon, where approximation error at the ends of the interval increases with the number of harmonics[39]. However, in our case, as we only care about the value of the approximation at the individual sample points, we can safely ignore this phenomenon.

In our experiments, we found that the Chebyshev series was significantly more sensitive to the initialisation of the weights before the initial approximation. For ChebConv, we found the best results were achieved by setting the weights of the "DC" harmonics to the mean of the kernel weights for each filter, and the weights of all higher harmonics are set to 0. Conversely, we found CosConv is less sensitive to the initial choice of weights, and that sampling weights from $\mathcal{N}(0, \frac{1}{C_{\text{in}}K^2})$ was sufficient. In both cases, we suspect that there is likely a better motivated initialisation scheme, but we leave this for future work. However for CosConv, we do not expect a better initialisation will improve performance.

4 Experiments & Results

Following this initial approximation, we fine-tune the network for a small number of epochs. This allows the network to correct for any small approximation errors that the initial approximation of weights might have caused. In all our experiments, we fine-tune the networks for 5 epochs. Specific details of training schedules, and the datasets used, are presented in the supplementary material.

4.1 CIFAR

		ResNet-20			ResNet-32				
	Layer	Pre Top 1	Post Top 1	Δ	Pre Top 1	Post Top 1	Δ	# Params	Δ %
	Conv2D	-	91.73	-	-	92.78	-	0.46M	0.0
	Fractional[41]	91.25	91.29	0.04	-	-	-	-	-
	FracSRF[35]	-	-	-	92.28	91.60	-1.18	0.16M	34.00
	FracSRF†	10.00	38.08	-53.65	9.82	37.91	-54.87	0.15M	33.43
CosConv	3,3,3,2	87.22	90.75	-0.98	87.74	91.51	-1.27	0.27M	57.74
CosConv	3,3,2,2	51.43	89.55	-2.18	75.62	90.99	-1.79	0.22M	47.35
CosConv	3,2,2,2	21.16	86.99	-4.74	16.52	88.4	-4.38	0.21M	44.57
CosConv	2,2,2,3	39.63	86.87	-4.86	20.92	88.64	-4.14	0.40M	86.65
CosConv	2,2,2,2	17.55	85.98	-5.75	15.39	87.87	-4.91	0.21M	44.48
ChebConv	3,3,3,2	87.22	90.76	-0.97	87.74	91.48	-1.30	0.27M	57.74
ChebConv	3,3,2,2	51.43	89.83	-1.90	75.62	91.16	-1.62	0.22M	47.35
ChebConv	3,2,2,2	21.16	87.79	-3.94	16.52	88.48	-4.30	0.21M	44.57
ChebConv	2,2,2,3	23.63	87.95	-3.78	20.92	89.26	-3.52	0.40M	86.65
ChebConv	2,2,2,2	17.55	86.66	-5.07	15.39	88.16	-4.62	0.21M	44.48

Table 1: Results showing the application of our method to the ResNet-20 and ResNet-32 models on CIFAR10. The sequence of numbers for the CosConv and ChebConv rows denote the number of "harmonics" for each of the blocks in the network. † indicates using our regression and retraining approach.

[2]For Chebyshev series, the Chebyshev-Gauss-Lobatto points. For cosine series, equispaced points.

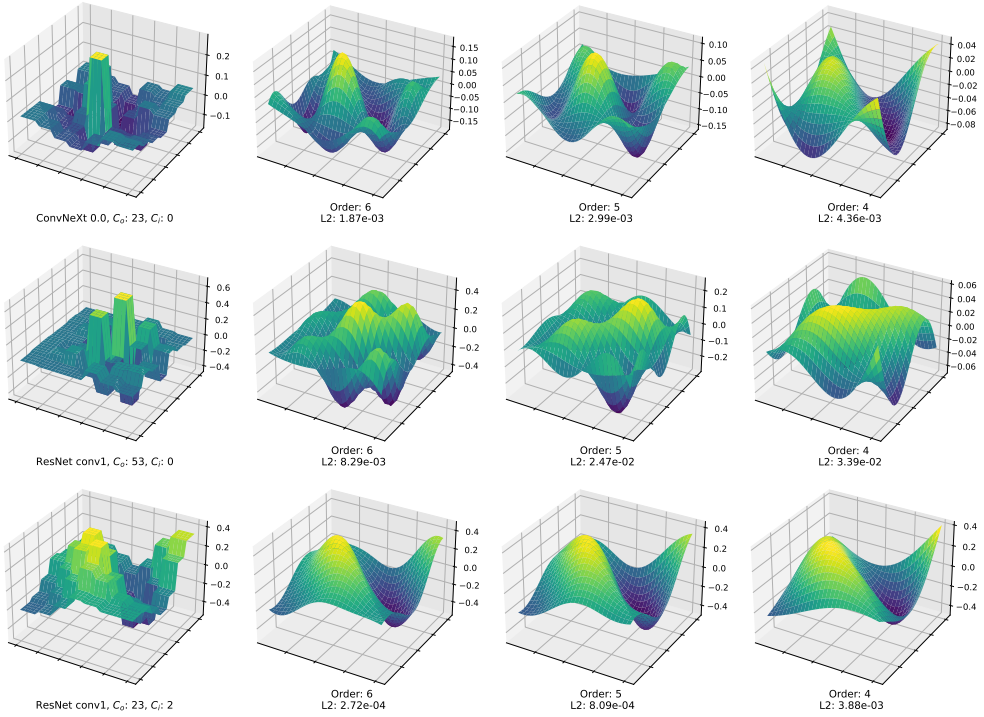


Figure 2: Filters from the ConvNeXt and ResNet networks compressed using our method. Left: Original Kernel, Right (in columns): Compressed version of this kernel using CosConv with 6, 5, and 4 orders/harmonics respectively.

We first conduct experiments on CIFAR-10[24] to validate our method. We use the smaller ResNet-20 and ResNet-32 networks from He et al. [16]. These models consist of 4 "blocks" of filters, with different numbers of channels for each block. The results of our experiments are presented in table 1. The sequence of numbers in the layer column next to the CosConv and ChebConv labels represent the number of harmonics used for the layers in each block. The columns show the accuracy after the initial approximation (Pre Top 1), and the accuracy after 5 epochs of fine tuning. The Δ columns show: i) the difference in Top 1 accuracy between the method/configuration and the baseline 2D convolution; ii) the percentage change in number of parameters. Our method is able to reduce the size of ResNet-20 by 42% losing only 1% accuracy. For both the 20 and 32 variants, the final block contains the majority of the parameters because of the larger channel dimension. Accordingly, reducing the number of parameters of this final block alone has more of an effect on the overall number of parameters than all of the other blocks combined (see Δ% for 2,2,2,2 *v.s.* 2,2,2,3). Whilst not designed to approximate previously learned filters, we investigate applying our "regression and re-training" approach to both Fractional Filters[41] as well as FracSRF[35]. FracSRF was not able to approximate the kernels initially, but was able to recover some of the accuracy during fine-tuning. We found that Fractional Filters were not able to approximate the kernels to any degree even after fine tuning, and the results were no better than random guessing (10% Top 1).

4.2 ImageNet

We conduct further experiments on the ImageNet[5] dataset. These experiments use both the ResNet-18[16] and ConvNeXt-T[26] models. The results of our experiments on ResNet-18 are presented in table 2. We see that our method is able to reduce the size of the model, with a modest cost to accuracy. Again, like ResNet-20 and ResNet-32, the structure of ResNet-18

	CosConv			ChebConv				
Config	Pre Top 1	Post Top 1	Δ %	Pre Top 1	Post Top 1	Δ %	# Params.	Δ %
Conv2D	-	69.76	0.0	-	69.76	0.0	11.68M	0.0
6,3,3,3,3	69.46	70.19	0.43	69.32	70.11	0.35	11.68M	99.98
6,3,3,3,2	64.33	68.97	-0.79	64.11	68.62	-1.14	7.09M	60.70
6,3,3,2,2	56.59	67.90	-1.86	56.35	67.23	-2.53	5.94M	50.88
6,3,2,2,2	14.56	66.89	-2.87	14.68	65.63	-4.13	5.66M	48.43
5,3,3,3,3	66.71	69.80	0.04	65.79	69.74	-0.02	10.99M	94.10
5,3,3,3,2	60.81	68.53	-1.23	59.71	68.15	-1.61	7.09M	60.68
5,3,3,2,2	51.92	67.48	-2.28	13.58	65.44	-4.32	5.94M	50.86
4,3,3,3,2	56.21	68.05	-1.71	54.37	67.34	-2.42	7.09M	60.67
4,3,3,2,2	45.99	66.84	-2.92	11.35	64.66	-5.1	5.94M	50.85

Table 2: Results showing the application of our method to ResNet-18 on the ImageNet dataset. The sequence of numbers for the CosConv and ChebConv rows denote the number of "harmonics" for each of the blocks in the network.

	Pre Top 1	Post Top 1	Δ %	# Comp. Params.	Δ %
Conv2D	-	82.10		0.33M	0.0
$7_{\times 3}, 7_{\times 3}, 7_{\times 9}, 6_{\times 3},$	80.27	81.19	-0.91	0.30M	90.90
$7_{\times 3}, 7_{\times 3}, 7_{\times 4}, 6_{\times 5}, 6_{\times 3}$	79.10	80.87	-1.23	0.27M	83.32
$7_{\times 3}, 7_{\times 3}, 6_{\times 9}, 6_{\times 3}$	70.10	79.94	-2.16	0.25M	77.25
$7_{\times 3}, 6_{\times 3}, 6_{\times 9}, 6_{\times 3}$	49.60	79.44	-2.66	0.25M	74.98
$7_{\times 3}, 7_{\times 3}, 5_{\times 9}, 5_{\times 3}$	62.40	79.28	-2.82	0.19M	58.01
$7_{\times 3}, 7_{\times 3}, 5_{\times 9}, 4_{\times 3}$	57.67	78.39	-3.71	0.17M	51.71
$7_{\times 3}, 7_{\times 3}, 4_{\times 9}, 4_{\times 3}$	20.30	75.94	-6.16	0.14M	42.26
$6_{\times 3}, 6_{\times 3}, 6_{\times 9}, 6_{\times 3}$	24.75	76.26	-5.84	0.24M	73.84
$5_{\times 3}, 5_{\times 3}, 5_{\times 9}, 5_{\times 3}$	14.94	77.29	-4.81	0.17M	51.71

Table 3: Results showing our method (CosConv) used on the 7×7 depth-wise filters from ConvNeXt, showing the accuracy after the initial approximation and then after fine tuning for 5 epochs. Comp. Params. (Compressible Parameters) shows the number of parameters making up the 7×7 convolutions in the network.

		CosConv			ChebConv				
bits	Config	Top1	Quant. Top1	Delta	Top1	Quant. Top1	Delta	# Params	Model size (MB)
32	Conv2D	69.76	69.76	0.0	69.76	69.76	0.0	11.68M	46.7
8	Conv2D	69.76	69.74	-0.02	69.76	69.74	-0.02	11.68M	23.4
8	6,3,3,3,2	68.97	68.12	-0.85	67.23	67.39	0.16	7.09M	14.2
8	6,3,3,2,2	67.90	65.23	-2.67	65.63	65.20	-0.43	5.94M	11.9
8	5,3,3,3,2	68.53	67.56	-0.97	68.15	66.80	-1.35	7.09M	14.2
8	5,3,3,2,2	67.48	64.69	-2.79	65.44	64.47	-0.97	5.94M	11.9
4	Conv2D	69.76	69.28	-0.48	69.76	69.28	-0.48	11.68M	17.5
4	6,3,3,3,2	68.97	66.70	-2.27	67.23	66.50	-0.73	7.09M	10.6
4	6,3,3,2,2	67.90	64.10	-3.80	65.63	63.88	-1.75	5.94M	8.9
4	5,3,3,3,2	68.53	66.38	-2.15	68.15	65.30	-2.85	7.09M	10.6
4	5,3,3,2,2	67.48	62.76	-4.72	65.44	63.10	-2.34	5.94M	8.9

Table 4: Results showing how our method applied to ResNet-18 on the ImageNet dataset performs under quantisation.

has increasing parameter counts through the blocks, and that compressing the final layers of the network provides outsize reduction in parameter count. Although the top configuration reduces the number of parameters in the first layer by 30% with no loss in Top 1 accuracy, the distribution of parameters in the network mean this has a barely noticeable effect on the overall parameter count (0.02%). However, decreasing the order further (config 5,3,3,3,3), reduces the number of parameters in the first layer by ≈50% with no loss to Top 1 accuracy, but still only resulting in a 6% reduction in model size. The deficiencies in the initialization of the ChebConv layers is more apparent here, where the same configuration loses more accuracy than an equivalent CosConv.

Figure 2 shows two filters from the first layer of ResNet-18 and one from the ConvNeXt-T model. We show CosConv approximations of various orders, as well as the L2 error between the approximation and the original kernel. Whilst our method is less able to recreate the strong discontinuities found in the depth-wise ConvNeXt kernels than the less discontinuous ResNet filters, our method is able to preserve much of the filter's structure.

To better investigate the capability of our method to reduce the parameter counts of layers with a larger kernel size, we evaluate our method on the 7×7 depthwise kernels that are present throughout the network, and present these results in table 3. Because the *vast* majority of the parameters in ConvNeXt pertain to operations are either pointwise/linear operations, or 2×2 downsampling kernels for which our method cannot be meaningfully applied, we consider only the "compressible parameters" which are the parameters of the 7×7 depth wise convolutions. The results show that our method is able to compress the applicable layers of the network by 42% with a 2.8% loss in Top 1 accuracy. We suspect that the impact of our method on ConvNeXt is not as strong as on ResNet-18, because ConvNeXt contains significantly more discontinuous kernels.

4.3 Quantization

To show that our method can complement other existing methods for reducing the size of deep models, we perform experiments to show the application of our method in tandem with quantisation. We use a basic quantisation scheme and apply it on top of our method, quantizing the parameters of our kernel functions, as well as the activations of the network. Further details of the setup for these experiments are presented in the supplementary material. Table 4 shows our experiments quantising ResNet-18 as well as applying our method. The 'Top 1' column shows the performance of the unquantised configuration, and 'Quant. Top1' shows the results after fine-tuning, applying both our method and quantisation. The results show that a further 3-4× reduction in model size, compared to unquantised models (40% reduction compared to quantised conventional convolution), is achieved with a 0.37% reduction in accuracy.

5 Conclusion

In summary, we have presented a novel method to reduce the number of parameters required to represent a conventional 2D convolution. Using functional approximations in the form of cosine and Chebyshev series, we are able to represent the weights of a kernel using fewer parameters. We outline an approach to learn an initial set of weights based on pre-trained models that can be further refined through a small amount of fine-tuning. Through experiments, we demonstrated that our method is able to reduce the size of common deep vision

models by as much as $\sim 50\%$ with a minor reduction in accuracy, as well as the compatibility between our method and quantisation.

In future work, we hope to examine how anisotropic and variable numbers of harmonics per filter might improve the compression.

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council [EP/R513295/1]. We would also like to thank Henry Howard-Jenkins for constructive discussions regarding this work.

A Datasets

We use two common datasets in our experiments: CIFAR-10[24] and ImageNet[5].

CIFAR-10 consists 50,000 training images and 10,000 validation images, distributed across 10 categories. During fine-tuning, images are randomly cropped to 32×32, randomly horizontally flipped and normalised. During validation/testing images are normalised identically to training.

The much larger ImageNet dataset consists of 1.28M training images and 50,000 validation images across 1000 categories. During fine-tuning, images are randomly cropped to 224×224, randomly horizontally flipped and normalised. At test time, images are center cropped to the same size, and normalised identically to training.

B Training Schedules

For ResNet-20, ResNet-32, and ResNet-18 we fine-tune using an SGD optimiser with the learning rate set to 1×10^{-4}, a momentum of 0.9, and weight decay of 5×10^{-4}. We reduce the learning rate by a factor of 10 after 3 iterations, although this had marginal impact on the final validation error.

For ConvNeXt-T, we fine-tune with with the same optimiser as above, but with the learning rate set to 5×10^{-5}. We also do not use any of the data augmentations from the paper[26], except for Stochastic Depth[20].

C Quantisation

For our quantisation we make use of the BFP quantisation scheme from [8, 30, 38], with a block size of 1. We quantize both weights and activations. In the backwards pass, we use the straight through estimator[2] to calculate the gradients. We use 7 bits for the exponent in all configurations, and quantize all weights and activations to the same number of bits.

D Further Kernel Visualisations

Figure 3, fig. 4, and fig. 5 show more visualisations of kernel functions compressed using our method. Figure 3 shows the same kernels as Figure 2. in the main paper, but using

ChebConv rather than CosConv. The remaining two figures show visualisations of further kernels using CosConv (fig. 4) and ChebConv (fig. 5)

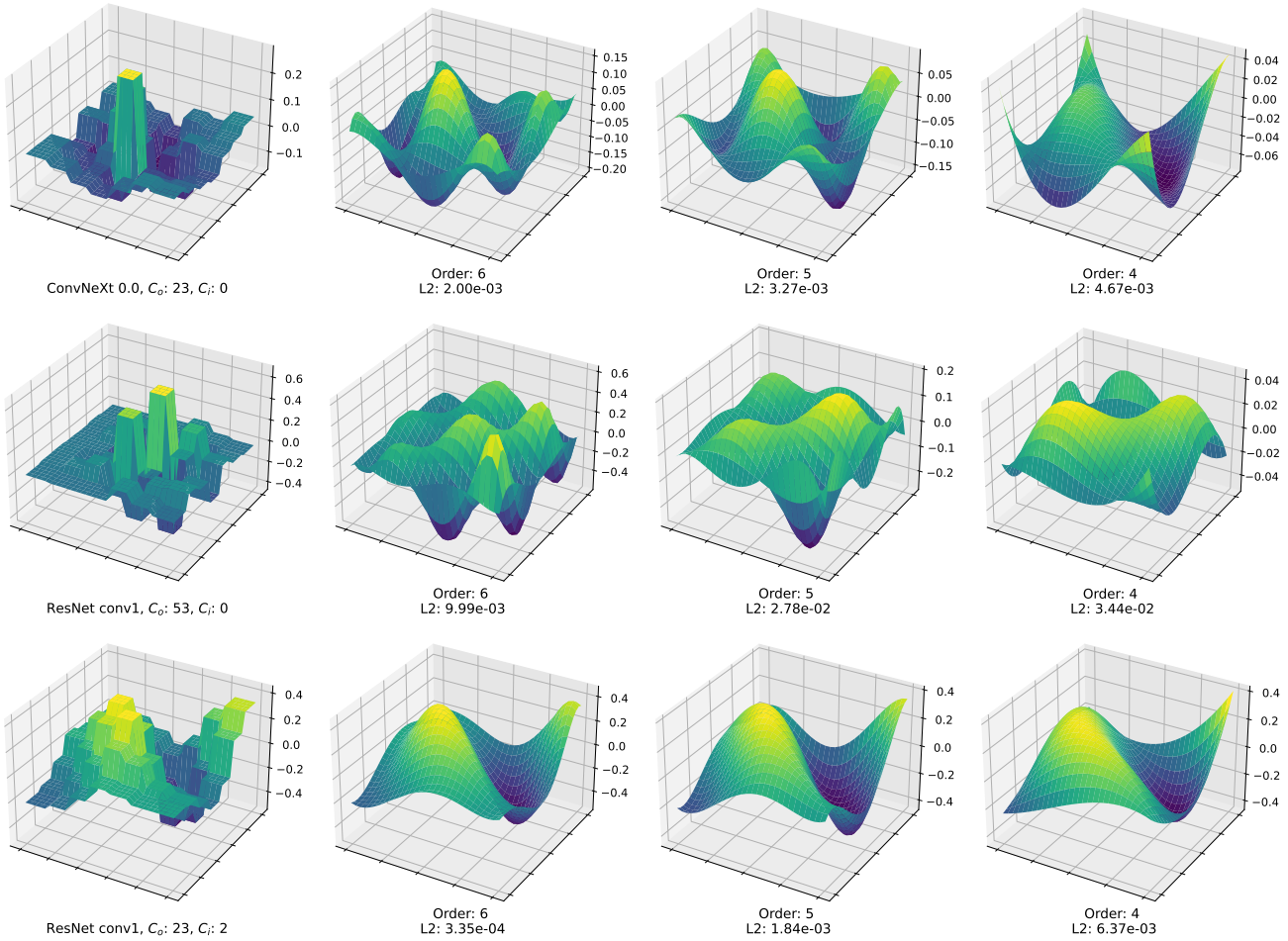


Figure 3: Filters from the ConvNeXt and ResNet networks compressed using our method. Left: Original Kernel, Right (in columns): Compressed version of this kernel using ChebConv with 6, 5, and 4 orders/harmonics respectively.

References

[1] Sajid Anwar and Wonyong Sung. Coarse pruning of convolutional neural networks with random masks. 2016. URL https://openreview.net/forum?id=HkvS3Mqxe.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language

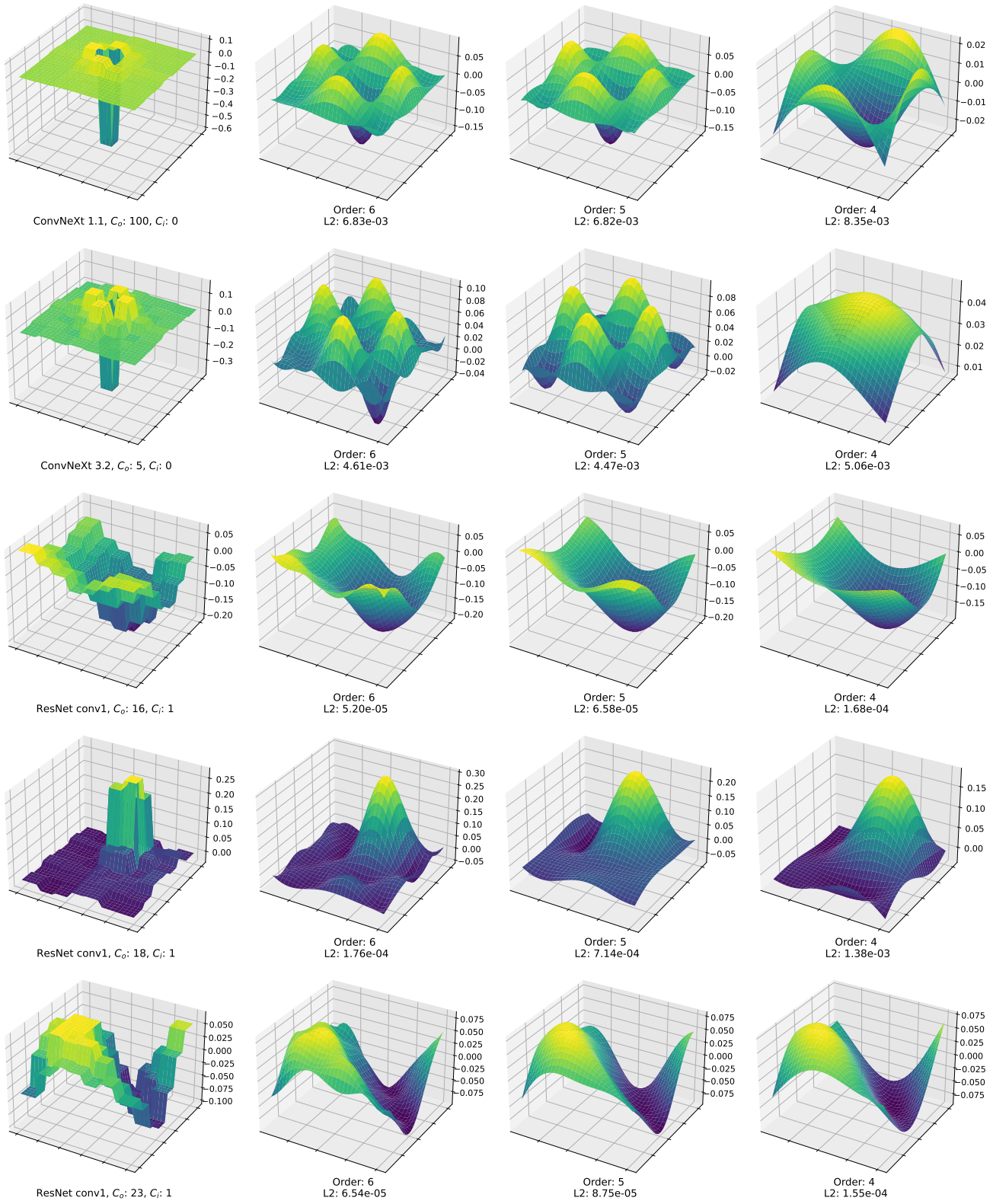


Figure 4: Filters from the ConvNeXt and ResNet networks compressed using our method. Left: Original Kernel, Right (in columns): Compressed version of this kernel using CosConv with 6, 5, and 4 orders/harmonics respectively.

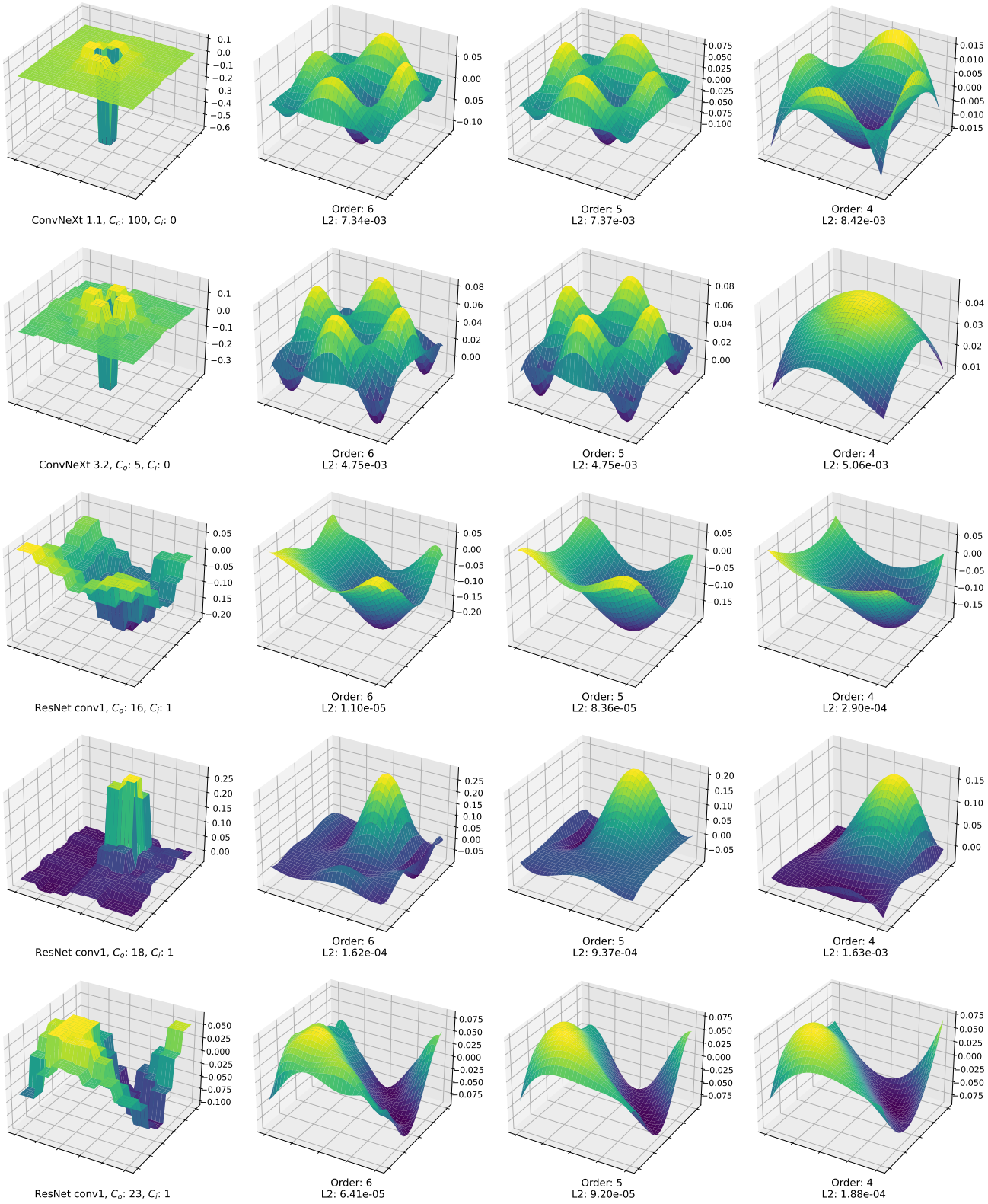


Figure 5: Filters from the ConvNeXt and ResNet networks compressed using our method. Left: Original Kernel, Right (in columns): Compressed version of this kernel using ChebConv with 6, 5, and 4 orders/harmonics respectively.

models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, October 2021.

[5] 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.

[6] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[8] Mario Drumond, Tao Lin, Martin Jaggi, and Babak Falsafi. Training dnns with hybrid block floating point. *Advances in Neural Information Processing Systems*, 31, 2018.

[9] Ahmed T Elthakeb, Prannoy Pilligundla, Fatemehsadat Mireshghallah, Amir Yazdanbakhsh, and Hadi Esmaeilzadeh. Releq: A reinforcement learning approach for automatic deep quantization of neural networks. *IEEE micro*, 40(5):37–45, 2020.

[10] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018.

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[12] Marcelo Gennari do Nascimento, Theo W Costain, and Victor Adrian Prisacariu. Finding non-uniform quantization schemes using multi-task gaussian processes. In *European Conference on Computer Vision*, pages 383–398. Springer, 2020.

[13] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch. Flex-convolution (million-scale point-cloud learning beyond grid-worlds). In *Asian Conference on Computer Vision (ACCV)*, Dezember 2018.

[14] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

[16] 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, 2016.

[17] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[21] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[22] Jorn-Henrik Jacobsen, Jan Van Gemert, Zhongyu Lou, and Arnold WM Smeulders. Structured receptive fields in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2610–2619, 2016.

[23] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[25] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[26] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, June 2022.

[27] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. Thinet: pruning cnn filters for a thinner net. *IEEE transactions on pattern analysis and machine intelligence*, 41(10):2525–2538, 2018.

[28] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

[29] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, pages 5191–5198, 2020.

[30] Marcelo Gennari do Nascimento, Roger Fawcett, and Victor Adrian Prisacariu. Dsconv: efficient convolution operator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5148–5157, 2019.

[31] Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S Palmer, Craig Packer, and Jeff Clune. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, 115(25):E5716–E5725, 2018.

[32] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[34] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.

[35] Nikhil Saldanha, Silvia L Pintea, Jan C van Gemert, and Nergis Tomen. Frequency learning for structured cnn filters with gaussian fractional derivatives. In *BMVC*, 2021.

[36] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[38] Zhourui Song, Zhenyu Liu, and Dongsheng Wang. Computation error analysis of block floating point arithmetic oriented convolution neural network accelerator design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[39] Lloyd N Trefethen. *Approximation theory and approximation practice [electronic resource]*. Other titles in applied mathematics. SIAM, Philadelphia, extended edition. edition, 2019. ISBN 9781611975949 (ebook).

[40] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019.

[41] Julio Zamora, Jesus A. Cruz Vargas, Anthony Rhodes, Lama Nachman, and Narayan Sundararajan. Convolutional filter approximation using fractional calculus. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 383–392, October 2021.

[42] 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, 2019.

[43] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015.

[44] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[45] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[46] Yue Zhou, Xiaofang Hu, Jiaqi Han, Lidan Wang, and Shukai Duan. High frequency patterns play a key role in the generation of adversarial examples. *Neurocomputing*, 459:131–141, 2021.

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	Approximating continuous convolutions for deep network compression
Publication Status	X Published □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Theo W Costain and Victor Adrian Prisacariu. Approximating continuous convolutions for deep network compression. In British Machine Vision Conference, 2022

Student Confirmation

Student Name:	Theo W. Costain		
Contribution to the Paper	• Conception and validation of the ideas • Designed, and wrote the code • Performed the experiments • Researched prior work • Wrote the manuscript • Presented poster at the conference		
Signature		Date	18/9/23

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 Victor Adrian Prisacariu		
Supervisor comments	I support the student's assesment of their contributions to this work.		
Signature		Date	18/9/23

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

4.3 Conclusion

With this chapter, we have presented our work *Approximating Continuous Convolutions for Deep Network Compression* that tackles the problem of compressing the weight tensors of a deep network. By considering the values of the weights as piecewise linear functions over space, we are able to approximate them using Cosine and Chebyshev series. In this way, we are able to apply the approximation after the network has been trained, requiring only a small amount of fine-tuning after application to recover full performance.

We test our method on the ResNet[29] and ConvNext[59] architectures, demonstrating the capability of our method to reduce the size of models by as much as half with minimal impact on accuracy. Further, we demonstrate that our method is compatible with other compression approaches like quantisation.

Whilst other tensor decomposition approaches are able to reduce the total number of operations used during inference on top of reducing model size, compared to other compression only approaches, our method is substantially easier and faster to apply.

Part II

Efficient Representations

5

Towards Generalising Neural Implicit Representations

Contents

5.1 Introduction

Implicit representations present and exciting new approach to the representation of 3D shapes and scenes. Encoding structure and shape information as a function over space, rather than explicit representations like point-clouds or meshes, they are able to capture complex geometry at arbitrary resolutions with a much smaller memory footprint than conventional representations. Whilst the efficiency of neural implicit representations (NIRs) has been established by others [16; 17; 164], we suggest that implicit representations present a different possible direction from which to consider efficiency. With implicit representations, we suggest that here we can consider the representational efficiency of the approaches, rather than simply

considering the efficiency of bits and floating point operations (FLOPs). From this point of view, we consider not just how something is represented, but what can be done with it in its current form.

Our work *Towards Generalising Neural Implicit Representations*, which we present in this chapter, investigates how and if implicit representations can be integrated into semantic task pipelines common in computer vision, and was one of the first works to investigate the use of NIRs in semantic pipelines. We show that when trained for reconstruction tasks alone, the shape encodings learnt by NIRs provide insufficient information to perform common semantic tasks like classification and segmentation. However, we argue that training for even a single semantic task alongside reconstruction tasks admits shape encodings that recover full performance on both reconstruction and the semantic tasks, and importantly these encodings maintain this performance in later "unseen" tasks.

5.1.1 Contributions

- We investigate the use of NIRs on a number of common computer vision tasks, and show that when trained for reconstruction tasks alone, the encodings learnt provide insufficient information to perform satisfactorily in semantic tasks.

- We demonstrate that the addition of a simple semantic task alongside reconstruction in a joint training fashion, is able to substantially improve performance on new tasks without compromising reconstruction performance.

Towards Generalising Neural Implicit Representations

Theo W. Costain[0000−0002−7803−6965]
Victor A. Prisacariu[0000−0002−0630−6129]

Active Vision Lab, University of Oxford
{costain,victor}@robots.ox.ac.uk

Abstract. Neural implicit representations have shown substantial improvements in efficiently storing 3D data, when compared to conventional formats. However, the focus of existing work has mainly been on storage and subsequent reconstruction. In this work, we show that training neural representations for reconstruction tasks alongside conventional tasks can produce more general encodings that admit equal quality reconstructions to single task training, whilst improving results on conventional tasks when compared to single task encodings. We reformulate the semantic segmentation task, creating a more representative task for implicit representation contexts, and through multi-task experiments on reconstruction, classification, and segmentation, show our approach learns feature rich encodings that admit equal performance for each task. Further, through hold-out experiments, we show that adding semantic supervision when training implicit encoders can significantly improve performance on later unseen tasks, without requiring encoder retraining.

1 Introduction

Implicit neural representations have garnered significant interest recently for their ability to reconstruct complex 3D structures and shapes. The appeal of these methods stems from a number of useful properties they possess for both reconstructing 3D shapes, as well as storing them efficiently. By learning to reconstruct the shapes, networks are able to encode and use a rich set of priors over the 3D domain to improve the quality of the reconstructions over what can be achieved with classical methods[24, 27]. Efficiency in storage is achieved by decoupling the encoding from the input and output modality so, unlike voxel based representation, the storage requirements do not grow cubically with the output resolution. Further,

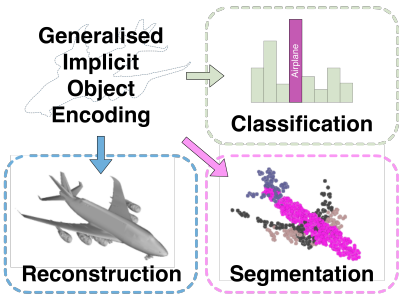


Fig. 1. Through multi-task training, implicit representations can be enriched creating a more general representation of a shape or object, and allowing for their use in a number of tasks rather than reconstruction alone.

implicit representations do not suffer from the limitations of mesh and point-cloud based representations, where the quality of the reconstruction is typically limited by the output size constraints of a single feed forward pass[24].

Conditioning a "decoder" network on an encoded representation of the input data, neural representations query the network at sample point locations for occupancy or distance function information. This approach allows for reconstructions to be generated with arbitrary resolutions at run-time[24].

Whilst these properties are impressive, we argue that further useful properties have been left on the table. In many of the works making use of implicit methods, training is performed with the loss function targeted only at reconstruction accuracy. This approach, whilst clearly effective, misses a significant potential benefit. We argue that using a multi task loss, including loss terms related to common tasks such as classification, produces encodings that are equally effective for reconstruction, but that still provide a richer set of features for use in other downstream tasks. We suggest that in applications such as augmented reality, where efficient representations are very useful, the ability to encode more than just shape information into the representation is likely to be useful.

Whilst a number of works have produced impressive and high quality reconstructions using a number of different approaches, there is a general aim in the design neural network encoders to produce features that have meaningful uses beyond a single task. However, we observe that this aim has not yet translated to implicit representation works. Many practical applications of implicit representations to real world problems would benefit from the ability to perform multiple tasks using the same stored data, rather than having to render and re-encode before performing other downstream tasks.

In this paper, we examine the generation of more descriptive neural representation encodings. Through experiments, we show that encodings generated purely for reconstruction can produce poor results on other tasks. We demonstrate the expected result that the encodings used in neural representations can be trained to develop properties useful for other tasks common in computer vision, without any appreciable reduction in reconstruction performance. We then detail surprising results in our hold-out experiments, showing that training encodings on even a single semantic task significantly improves performance on later held out tasks. We also argue that the conventional 3D semantic segmentation task does not translate well to implicit representations, where the object being reconstructed is not or cannot be operated on directly. To address this, we propose, among other experiments, a re-formulation of the semantic segmentation task that is a more representative formulation when applied to implicit representations.

In summary, the key contributions of this paper are

- Investigation of the use of implicit encodings on a number of common computer vision tasks (in a multi-task setting), showing improved performance in these tasks when compared to reconstruction-only encodings, without compromising reconstruction accuracy.

- Showing that the addition of a simple semantic task alongside reconstruction is able to substantially improve performance on new tasks, *without* requiring retraining of the encoder. For example, we show training an encoder for both reconstruction and classification significantly improves later segmentation results.
- A re-formulation of the semantic segmentation task, that is more representative of a real world task in the context of implicit representations.

2 Related Work

Implicit (or Neural) representations have been the subject of much recent work. Early works focused on single objects[24, 35, 32, 27, 25, 11, 2, 12, 30, 41, 5], encoding either image or point-cloud input into a feature vector, which is used to condition a decoder network. These decoder networks typically come in one of two forms: concatenation/biasing or hyper-networks. In concatenation based conditioning ([10] argue that biasing and concatenation are analogous), the encoding is concatenated with the point being queried and then passed through the network. In hyper-networks, the encodings are passed through a small network, whose outputs are the weights used in the network that predicts the value for a given query point. The outputs of these decoders can typically be divided into two categories, namely occupancy generating or signed distance function (SDF) generating[1].

Early works such as [35, 27, 24, 5] showed that simple MLP networks were capable of representing complex distance functions and occupancy functions. [27] (DeepSDF) also detailed the use of auto-decoders to estimate optimal encodings for a given input, using a fixed decoder and simple backpropagation. [24] (Occupancy Networks) demonstrated the alternative occupancy paradigm for implicit representations, as well as proposing a procedure to extract high quality meshes in an efficient manner from the implicit representation, using an octree like approach. Concurrently, [5] (IM-Net), also using the occupancy paradigm, showed impressive results for single view reconstruction, as well as both 2D and 3D shape generation. [25] learn level sets to represent shapes. Similarly, [2] (SAL), used unsigned function priors to train a signed distance function. [30] combine both explicit atlas based reconstruction and implicit neural reconstruction, enforcing consistency between the two methods.

Later works investigated representing larger scenes[3, 28]. Many of these methods did not expand the size of the area described by a given embedding, instead proposing methods to recover encodings for a small local region where an implicit representation can then extract local shape information. [28] (Convolutional Occupancy Networks) interpolated between encoded points in a volume

[1] Arguably occupancy networks are simply SDF networks with the sign function applied to their output, however this ignores the increased complexity in regressing SDF values rather than simply their sign. We discuss this point in more detail in Sec. 3.1

or plane, to generate the conditioning vector for a occupancy network in a region around the encoded point. [6] took a similar approach, adding also multiple resolutions of encoded volumes. A separate group of works[17, 3, 38] all took a slightly different approach from above and divided the scene into regions, generating a small encoding for each region. Both [17] and [3] made use of a grid of small local encodings, whereas [38] made use of a number of oriented spherical patches of differing radii each with an encoding.

Further improvements to implicit representations in general were proposed by [33] and [36] showing that adding higher frequency information to simple networks drastically improved their ability to generate high quality reconstructions. [9] proposed a curriculum based learning approach for implicit representations, improving reconstruction quality of complex local details.

Other works have used implicit representations for a number of other tasks, most notably novel view synthesis[21, 26, 34, 43].

There have also been a number of works considering the application of multi-task approaches to 3D problems[29, 19, 14, 20]. Multi-task learning has enabled improvements where tasks are related or closely coupled, such as semantic and instance segmentation[19, 29]. As well, [14] made use of a multi-task setup in unsupervised training to learn a useful embedding space for 3D point-cloud inputs.

Latent representations have been used to bridge the gap between point-cloud and volumetric representations[23], whilst others have sought to learn directly on meshes rather than point clouds or voxel grids[13, 16].

The concurrent work of [43] examines a similar use of implicit representations in a multi-view synthesis setting (based on [21]). Their work showed that the addition of geometric information can improve robustness in the semantic task, achieving strong performance in noisy environments while being able to achieve remarkable accuracy with little semantic supervision. [18] also examine semantics in a multi-view setting, using [35] for the internal representation. Their method also demonstrates the close relationship between appearance, geometry, and semantics, being able to synthesise novel appearance views from semantic images and vice versa.

3 Tasks

In this section, we first cover the principles of implicit representations. We then describe the other tasks we consider, including our variation to the normal task of segmentation that we use in our experiments, for a fairer representation of the segmentation task in implicit contexts.

3.1 Implicit Representation

Neural implicit representations attempt to estimate the function describing the surface of a given object. A common formulation is to map from a point in space, $\mathbf{p} \in \mathbb{R}^3$, to the smallest signed distance between the point and the outer

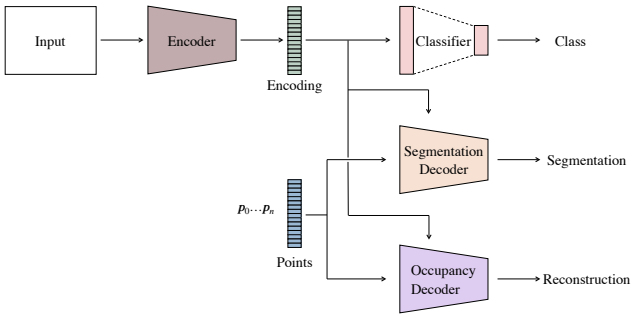


Fig. 2. An overview of our network architecture. The network takes as input either images or point-clouds, generating an encoding from them. This encoding can then be used in a number of ways. For classification, the encoding is passed directly into a simple classifier. For segmentation and reconstruction, the encoding is used to condition the decoder networks. The decoder networks take a number of points as input and returns for each point either, the probability that that point lies inside the encoded shape, or semantic label probabilities.

face of a surface, *i.e.* a SDF. This gives rise to an expression[27, 41] of the form $s : \mathbb{R}^3 \rightarrow \mathbb{R}$. Another common formulation is to estimate the probability that a given point lies within the object (*i.e.* probability of occupancy), rather than regressing the SDF directly. This gives a function[24] of the form $o : \mathbb{R}^3 \rightarrow \{0, 1\}$. However, we note the following relationship

$$o = \sigma(-s) > \tau \tag{1}$$

where τ is a threshold parameter, and σ is a sigmoid function. This relationship suggests that the occupancy function is a simplified version of the SDF, moving the problem from a regression context and into a classification context. Further, this reformulation suggests that where the extra information provided by the SDF (but not the occupancy function) such as the surface normal (given as, $\frac{\partial s}{\partial \mathbf{p}}$, the spatial gradient of the SDF[27]) is not needed, the occupancy function is likely to be easier to learn. If this is true, we suggest this is the result of the occupancy network only needing to learn a decision boundary over $\mathbb{R}^3$ rather than having to learn both the boundary and then regress a points distance from it. In addition, we note that DeepSDF applies clamping to its loss function, such that points away from the surface do not impact the loss. Further, DeepSDF does not enforce the Eikonal constraint[2] like other works[25, 12]. Accordingly the network is mainly learning the zero crossing point of the SDF rather than a metric SDF.

This then implies that differences between Occupancy Networks and DeepSDF lie mostly in the training and design of the two networks, and not in the functions they embed. For this reason, we believe that although the evaluation in this paper is conducted soley on Occupancy Networks, the results should also apply

[2] A requirement for a "true" SDF

to DeepSDF based encodings. Given this, and the simplicity of the method, we chose to use the formulation of Occupancy Networks[24] for our experiments.

3.2 Other Tasks

The conventional 3D segmentation task as explored in a number of papers[31, 40] typically involves predicting a semantic label for each point in an given input point-cloud. However, in the context of implicit representations, this task loses much of its meaning, as we want to learn semantic information at locations not in the input pointcloud. This of particular concern when the input is a degraded and noisy point-cloud (Sec. 4). As we are considering the occupancy of a given spatial location, it makes more sense to consider the task as determining the semantic label of regions within the shape.

Hence, given a mesh $\mathcal{X}$, for each vertex $\mathbf{x} \in \mathcal{X}$ with a semantic label $c_{\mathbf{x}}$, the semantic class of any location $\mathbf{p} \in \mathbb{R}^3 \cap \mathcal{X}$ lying inside the mesh, has the semantic class of the nearest vertex of $\mathcal{X}$.

$$c_{\mathbf{p}} = c_{\mathbf{z}} \qquad \text{where } \mathbf{z} = \arg\min_{\mathbf{x}} ||\mathbf{p} - \mathbf{x}||_2$$

This scheme has the same effect as producing a Voronoi partitioning of the space inside the mesh. Points that lie outside the mesh, are considered to be background and therefore have no valid semantic class. The segmentation task then becomes predicting the label of a point inside the shape according to the nearest neighbour assignment. During both training and inference, we evaluate the semantic label task at the same locations as the reconstruction task.

As well as segmentation, we also investigate the performance of our approach in the task of classification. Unlike the segmentation task which requires the implicit code to encode information about the properties of spatial regions (similarly also with the reconstruction task), classification requires that the encodings allow simple classification networks to discriminate between them. Later experiments (Sec. 5.2 & Sec. 5.3), show that the requirements classification has for the encodings are noticeably different to reconstruction.

Our results show that implicit representations can be encouraged to be more representative of objects, rather than merely encoding their shape. We focus on two particular tasks that are common tasks, but expect that generalising the encodings over further tasks is likely to also be possible.

4 Experiments

The experiments are divided into three parts. First we consider the original dataset from [24], establishing a baseline and some preliminary experiments involving reconstruction and classification. Next we examine a more challenging classification task, before turning finally to a semantic segmentation task.

An overview of our network architecture is shown in Fig. 2. Specific details of the architecture for each experiment are outlined in Sec. 4.2. The loss functions

used depends on the task. For the reconstruction loss, $\mathcal{L}_{\text{rec}}$, we use binary cross entropy as in[24]. Both classification, $\mathcal{L}_{\text{cls}}$, and segmentation, $\mathcal{L}_{\text{seg}}$, use the cross entropy loss. In the multi task settings, the losses are combined in a weighted linear fashion as $\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{cls}} \times \mathcal{L}_{\text{cls}} + \lambda_{\text{seg}} \times \mathcal{L}_{\text{seg}}$. For all experiments $\lambda_{\text{cls}} = \lambda_{\text{seg}} = 1.0$. We use an ADAM optimiser with learning rate of 10^{-4}. Training takes approximately 4 days on a NVIDIA GeForce GTX 1080Ti.

4.1 Datasets

We perform our experiments on a number of datasets. The original dataset (hereafter Choy) from [24] is the subset of ShapeNetCore[4] from [7]. We also make use of ModelNet40[39] for further classification experiments and ShapeNetPart[42] for our segmentation experiments. Data pre-processing was accelerated with GNU Parallel[37].

We limit our experiments to datasets with similar properties to those used in [24], as we are not seeking to validate the specific implicit representation format we are using, rather the benefits of more feature rich encodings. This means that we do not consider larger scale datasets such as Stanford3D[1] that our chosen method might struggle with. We leave this to future experiments with other methods such as [28] or [3] that are better able to reconstruct larger scenes.

For all experiments, the properties of the inputs remain constant. For point-clouds we sample 300 points from the ground truth point-cloud, and apply noise using a Gaussian distribution with zero mean and standard deviation 0.05 to the sampled point clouds, identically to [24]. For images we crop and resize the images identically to [24].

Choy / ShapeNetCore The dataset used in [24] from which our work builds on, uses the renderings and voxelisations[7] of a subset of the ShapeNetCore[4] dataset. We use the rendered images to train the image based encoder in later experiments. The fully processed dataset was provided by [24] as part of their publication. Briefly, meshes are loaded and a large number of depth images are rendered. These depth images are fused to form a watertight mesh from which points and their corresponding occupancy value can be sampled. Although the occupancy samples are not provided as part of the dataset in[7], to reduce ambiguity we will refer to the dataset from [24] as the Choy dataset throughout this paper. The dataset consists of 30,648 training meshes, 4,358 validation meshes and 3,738 test meshes across 13 object categories.

We use the Choy dataset both for our baseline experiments, as well as some preliminary classification experiments. Our experiments with this dataset are outlined in Sec. 5.1.

ModelNet40 For further classification experiments, we make use of the ModelNet40[39] dataset. As rendered images were not readily available, we rendered images using Pyrender[22] in the same fashion as [7], choosing 24 viewpoints with constant radius and altitude, but random azimuth. The occupancy samples are

generated with the code provided by [24]. The dataset consists of 9,843 training meshes and 2,468 testing meshes across 40 object categories. Our experiments with this dataset are outlined in Sec. 5.2.

ShapeNetPart For our semantic segmentation experiments, we make use of the dataset from [42], which we refer to as ShapeNetPart. Again the occupancy samples were generated using the code from [24]. Semantic labels were assigned to the occupancy samples using the procedure outlined in Sec. 3.2 from the ground truth semantic labels in[42]. The dataset consists of 12,121 training, 1,854 validation, and 2,858 testing meshes following the corresponding splits from ShapeNetCore. Our experiments with this dataset are outlined in Sec. 5.3.

4.2 Architecture

Our network takes either point-clouds or images as input to the encoder. In all the experiments, we use the same two encoders: one for point-cloud input, and another for image input.

Point-cloud input We use the same variation on the original network from [31] as [24]. In this formulation, the fully connected (FC) layers normally present in the original network are replaced by residual FC blocks[15]. During training the network samples 300 points from the input point cloud and applies Gaussian noise ($\sigma = 0.005$) before passing these into the encoder (identically to [24]).

Image input We use a pre-trained ResNet-18[15], followed by a linear layer to reduce the output dimension following [24].

The encoded features are then passed to a decoder. For decoding point locations into either occupancy values or semantic labels we use one or more of the following, depending on the task(s). For classification, the encoding is passed directly to the classifier.

Task Decoders Occupancy Decoder This is the same decoder used in [24]. The network takes a number of points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$ as input and uses conditional batchnorms[8], which take the encoding as their input, to condition the network.
Classifier A simple 2 layer MLP, that takes the encoding directly as input and returns class probabilities.
Segmentation Decoder The same network as the occupancy decoder but with a larger output channel dimension.
Parallel Segmentation and Occupancy Decoders This is the configuration shown in Fig. 2, in which the segmentation and occupancy decoders operate in parallel.
Joint Segmentation and Occupancy Decoder Also the same network as the occupancy decoder, however rather than two separate networks for each task,

	Recon.		Class.
	IOU ↑	Chamfer L1 ↓	Accuracy ↑
ONet baseline	0.78	0.0081	–
Classification baseline	–	–	0.92
Classification w/ ONet encoder	–	–	0.80
Joint Classification & ONet	0.77	0.0084	0.92

Table 2. Experiments on the Choy dataset with point-cloud input, showing shape IOU, Chamfer L1, and classification accuracy. For "Classification w/ ONet encoder" the encoder is pre-trained on the ONet baseline, fixed, and only the classification decoder trained.

the same network performs both tasks simultaneously. The output is then sliced along the channel dimension to yield two tensors, one containing the occupancy probability, the other containing the semantic label probabilities.

4.3 Encoder Baselines

To fairly compare later experiments to Occupancy Networks, we use the encoders from the original paper throughout. To provide comparisons between our tasks and conventional tasks on point clouds, we run two small experiments to show how the encoder (a modified PointNet) from Occupancy Networks performs compared to a baseline PointNet[31]. These experiments are shown in Table 1, using results directly from [31]. For classification on ModelNet40, the network receives 1024 input points perturbed by Gaussian noise with zero mean and 0.02 standard deviation, as in [31]. For segmentation on ShapeNetPart, the network receives 2048 points as input to the encoder, and the input is also used as the query points ($p_0 \dots p_n$ in Fig. 2). We note that for these experiments the network receives at least 3x more input points, with less additive noise, than in later experiments.

(a) Classification on ModelNet40

	Accuracy
PointNet[31]	**87.1**
ONet Encoder + Classifier	85.9

(b) Segmentation on ShapeNetPart

	mIOU
PointNet[31]	**83.7**
ONet Encoder + Segmentation Decoder	83.0

Table 1. Encoder comparison baselines.

5 Results

5.1 Choy Experiments

We begin with the dataset from [24]. Our experiments with point-cloud input are shown in Table 2. Given the small number of classes and fairly distinct visual properties of the classes in this dataset, the high accuracy in classification is not unexpected, even with the reduced quality of the input point-clouds. To evaluate the classification performance of the baseline encoder, we fix its parameters

	Recon.		Class.
	IOU ↑	Chamfer L1 ↓	Accuracy ↑
ONet baseline	(0.58 / 0.57)	(0.021 / 0.021)	–
Classification baseline	–	–	(0.92 / –)
Classification w/ ONet encoder	–	–	(– / 0.63)
Joint Classification & ONet	(0.59 / 0.57)	(0.020 / 0.023)	(0.92 / 0.91)

Table 3. Experiments on the Choy dataset with image input, showing shape IOU, Chamfer L1, and classification accuracy. Values in brackets: left with ResNet pre-training, right without.

	Recon.		Class.
	IOU ↑	Chamfer L1 ↓	Accuracy ↑
ONet baseline	0.73	0.011	–
Classification baseline	–	–	0.82
Classification w/ ONet encoder	–	–	0.57
Joint Classification & ONet	0.70	0.012	0.82

Table 4. Experiments on the ModelNet40 dataset with point-cloud input, showing shape IOU, Chamfer L1, and classification accuracy.

and train a simple 2 layer MLP classifier to operate on its output encodings. Classification on this fixed encoder shows a substantial reduction in accuracy compared to the jointly trained case, *i.e.* where the encoder is *not* fixed. Notably in this joint training scenario, full accuracy in both tasks is recovered.

Our experiments with image input are shown in Table 3. The results are similar to the point-cloud experiments. As discussed in [24], the lower performance in reconstruction for the ONet can potentially be attributed to occlusion. We also train the baseline from scratch without ResNet pre-training. Without pre-training the same "Classification w/ONet encoder" experiment shows a similar result to the point-cloud experiment where the network performs poorly on the classification task, as the network is not encouraged to generate features meaningful to other tasks. Further, without pre-training, the network was significantly less stable, and convergence was slower. The joint training result shows that the encoding is capable of performing both tasks without loss of accuracy.

5.2 ModelNet40 Experiments

To better evaluate the classification performance, as well as the shortcomings of the reconstruction encodings in classification, we run the same experiments as in Sec. 5.1 on ModelNet40, a more conventional 3D classification benchmark.

Our experiments with point-cloud input are shown in Table 4. The results follow a similar pattern to the point-cloud results from the Choy dataset. As we expected, when we train the classifier using the fixed encoder from the reconstruction task, the classification performance is poor. This reduction in performance is much more severe than on the Choy dataset, but is consistent with the

	Recon.		Class.
	IOU ↑	Chamfer L1 ↓	Accuracy ↑
ONet baseline	(0.54 / 0.49)	(0.034 / 0.038)	–
Classification baseline	–	–	(0.85 / –)
Classification w/ ONet encoder	–	–	(– / 0.51)
Joint Classification & ONet	(0.51 / 0.48)	(0.036 / 0.042)	(0.84 / 0.81)

Table 5. Experiments on the ModelNet40 dataset with image input, showing shape IOU, Chamfer L1, and classification accuracy. Values in brackets: left with ResNet pre-training, right without.

increased difficulty shown by the lower accuracy figure on the classification baseline. However, this performance loss is completely recovered in the joint training, with only a minor decrease in reconstruction performance.

Our experiments with image input are shown in in Table 5. Here we see that the joint training is able to recover much of the performance on either of the single tasks. Again, the experiments without pre-training show similar trends to the fixed encoder point-cloud experiments, where the classifier struggles with the ONet encoder, but full accuracy is recovered in joint training. We suspect the lower performance of the non-pre-trained networks can be attributed to the more varied nature of ModelNet40.

We also suggest that the reduction in performance for the joint task IOU in Tables 4 & 5 is more a result of specifics of the IOU metric than a meaningful reduction in performance. This can be seen from Table 3 where the Chamfer L1 loss is reduced by the same amount, but the IOU performance is less affected than in Tables 4 & 5.

5.3 ShapeNetPart Experiments

Our metric for the segmentation task is mean average Intersection over Union (mIOU). Points are sampled within the shape and assigned semantic labels by the decoder. The same sample points are used for both segmentation and reconstruction. Whilst in a real world scenario points would be sampled both inside and outside the shape, we wish to assess the performance of the segmentation decoder independently of the reconstruction performance, and so only consider points inside the shape. The IOU is computed for each part of the shape, and averaged to give a shape IOU. If there are no ground truth points for a given part (*e.g.* 'armrest' is a part of the chair class, but several chair instances do not have armrests), the part is automatically assigned an IOU of 1. We can then compute mIOU as the average of the shape IOUs. At inference time, points are sampled randomly from a padded bounding box of the ground truth object, as in [24].

We evaluate two different part segmentation decoders: the joint decoder where one decoder produces both segmentation and reconstruction at the same time, and the parallel decoder (the configuration shown in Fig. 2) where two

	Recon.		Seg.	Class.
	IOU ↑	Chamfer L1 ↓	mIOU ↑	Accuracy ↑
ONet baseline	0.69	0.010	–	–
Classification baseline	–	–	–	0.95
Segmentation baseline	–	–	0.53	–
Joint Segmentation & ONet	0.70	0.0098	0.50	–
Joint Segmentation & Classification & ONet	0.72	0.0086	0.50	0.95
Parallel Segmentation & ONet	0.68	0.011	0.53	–
Parallel Segmentation & Classification & ONet	0.70	0.0095	0.53	0.95

Table 6. Experiments on the ShapeNetPart dataset with pointcloud input, showing shape IOU, Chamfer L1, segmentation mIOU, and classification accuracy.

similar decoders handle segmentation and reconstruction respectively. A more detailed explanation is presented in Sec 4.2.

Table 6 shows the reconstruction accuracy, mIOU, and classification accuracy of our different experiments on the ShapeNetPart dataset. The results show little to no accuracy being lost in any of the tasks for the jointly trained settings. In the reconstruction task, the network is attempting to learn an encoding that represents the shape properties of a given region of space, such as the curvature and boundaries. These properties are likely also useful for the task of segmentation, *i.e.* the semantic class probabilities are potentially somewhat dependant on properties like local curvature.

Table 7 shows the per-class segmentation results for the baseline, joint training, and the reconstruction IOU for the baseline. These results show that the parallel architecture is superior to the joint architecture, and suggest that, although these particular tasks are well correlated at the encoding level, they may be less well corre-

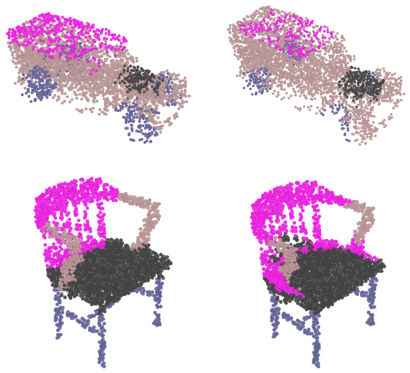


Fig. 3. Qualitative joint reconstruction and segmentation results on the ShapeNetPart[42] dataset, showing and example from the car (top) and chair (bottom) classes. Right shows a Joint segmentation & classification & ONet reconstructed mesh coloured with semantic class labels. Left shows the same reconstruction with ground truth labels.

Decoder Type	ONet	Seg.	Class.	Airplane	Bag	Cap	Car	Chair	Earphone	Guitar	Knife	Lamp	Laptop	Motorbike	Mug	Pistol	Rocket	Skateboard	Table	Mean
Reconstruction IOU	✓			*0.75*	*0.71*	*0.56*	*0.8*	*0.70*	*0.56*	*0.75*	*0.7*	*0.54*	*0.81*	*0.53*	*0.76*	*0.75*	*0.73*	*0.68*	*0.70*	*0.69*
Seg. Only		✓		0.59	0.46	0.45	0.52	0.63	0.31	0.68	0.55	0.52	0.59	0.53	0.58	0.59	0.39	0.51	0.57	0.53
Joint Decoder	✓	✓		0.55	0.45	0.40	0.51	0.62	0.30	0.66	0.53	0.49	0.59	0.47	0.53	0.56	0.30	0.47	0.56	0.50
	✓	✓	✓	0.55	0.42	0.38	0.49	0.62	0.32	0.66	0.54	0.49	0.59	0.45	0.55	0.56	0.3	0.46	0.56	0.50
Parallel Decoders	✓	✓		0.59	0.50	0.45	0.53	0.63	0.33	0.68	0.56	0.52	0.59	0.53	0.58	0.60	0.39	0.50	0.57	0.53
	✓	✓	✓	0.59	0.51	0.42	0.53	0.63	0.34	0.68	0.55	0.53	0.60	0.55	0.57	0.58	0.38	0.50	0.57	0.53

Table 7. Experiments on the ShapeNetPart dataset with point-cloud input, detailing per class results, showing segmentation mIOU. Note the first row is reconstruction IOU.

lated within the decoder. The poor performance on some of the classes such as rocket and headphones may be explained by the thin sections in parts of those objects. Because the network samples points within the shapes randomly, thin sections like the fins(rocket), cable(earphones), or handlebar(motorbike) are likely to be undersampled and therefore have poor performance at inference time. As well as this imbalance, there is also significant imbalance in the number of models in certain categories, which can negatively affect accuracy at inference time. This is reflected in the higher mIOU scores (across all the experiments), for the classes with more shapes.

Fig. 3 shows selected qualitative joint reconstruction & segmentation results.

5.4 Hold-out Experiments

Table 8 outlines our hold-out experiments, in which we aim to show the task generalisation of the network. In these experiments, we train the encoder and reconstruction decoder alongside either the segmentation decoder or the classifier. These weights are then frozen and the remaining decoder is trained.

As the above results show, an encoder trained for reconstruction alone learns features that encode shape information well. However the results suggest that shape information alone is not sufficient information for either segmentation or classification where significant accuracy is recovered for both tasks in the joint setting. Table 8 shows a particularly interesting result in that the joint training obtains similar results regardless of whether classification or segmentation was used alongside reconstruction to train the encoder. We suggest the following explanation for this behaviour: the reconstruction task teaches geometric information to the network, but not necessarily any semantic information. Both segmentation and classification are able to provide sufficient semantic information that, when combined with the geometric information, give sufficient capability for new tasks that require semantic and/or geometric information. To support this we point out that training for classification and reconstruction,

	Recon. IOU ↑	Seg. mIOU ↑	Class. Accuracy ↑
Classification w/ ONet encoder	–	–	0.89
Segmentation w/ ONet encoder	–	0.48	–
Multi Task w/ Seg & ONet encoder	0.70	0.53	0.95
Multi Task w/ Classification & ONet encoder	0.70	0.53	0.96

Table 8. Hold-out experiments on the ShapeNetPart dataset with point-cloud input, showing shape IOU, segmentation mIOU, and classification accuracy.

despite providing no further point specific information than reconstruction only, still improves the point specific segmentation task by 5%. Furthermore, training the encoder for segmentation and reconstruction, despite providing no explicit class information, allows the network to recover the full accuracy on the later classification task, giving a performance improvement of 6%. We note that, given this dataset is a fairly simple and quite biased as a classification task, an improvement of this amount suggests noticeably degraded performance (for the reconstruction-only encoder) on this task.

6 Conclusion

In this paper we have discussed generalising the encodings used by implicit representations to a broader range of tasks. We discuss the current narrow focus of implicit representations, and the potential issues this raises for applications of implicit representations in the real world. We introduced a modified formulation of the conventional segmentation task that is more applicable to implicit contexts, and detail an appropriate network to use for this new formulation. We show that as currently used, implicit encodings struggle to match the performance of the original data they aim to replace on common computer vision tasks, such as classification or part segmentation. We choose two common tasks and demonstrate that through multi-task training, we can enrich the encodings, achieving strong performance across the tasks without any loss in reconstruction accuracy. Through hold-out experiments, we showed improved performance on unseen tasks, when compared to single task training, without needing to retrain the encoder.

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council [EP/R513295/1]

Bibliography

[1] Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1534–1543 (2016)

[2] Atzmon, M., Lipman, Y.: Sal: Sign agnostic learning of shapes from raw data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2565–2574 (2020)

[3] Chabra, R., Lenssen, J.E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., Newcombe, R.: Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. arXiv preprint arXiv:2003.10983 (2020)

[4] Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)

[5] Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5939–5948 (2019)

[6] Chibane, J., Alldieck, T., Pons-Moll, G.: Implicit functions in feature space for 3d shape reconstruction and completion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6970–6981 (2020)

[7] Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: European conference on computer vision. pp. 628–644. Springer (2016)

[8] De Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., Courville, A.C.: Modulating early visual processing by language. In: Advances in Neural Information Processing Systems. pp. 6594–6604 (2017)

[9] Duan, Y., Zhu, H., Wang, H., Yi, L., Nevatia, R., Guibas, L.J.: Curriculum deepsdf. arXiv preprint arXiv:2003.08593 (2020)

[10] Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H.d., Courville, A., Bengio, Y.: Feature-wise transformations. Distill (2018). https://doi.org/10.23915/distill.00011, https://distill.pub/2018/feature-wise-transformations

[11] Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W.T., Funkhouser, T.: Learning shape templates with structured implicit functions. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 7154–7164 (2019)

[12] Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. arXiv preprint arXiv:2002.10099 (2020)

[13] Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: Meshcnn: a network with an edge. ACM Transactions on Graphics (TOG) **38**(4), 1–12 (2019)

[14] Hassani, K., Haley, M.: Unsupervised multi-task feature learning on point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 8160–8171 (2019)

[15] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[16] Hu, S.M., Liu, Z.N., Guo, M.H., Cai, J.X., Huang, J., Mu, T.J., Martin, R.R.: Subdivision-based mesh convolution networks. arXiv preprint arXiv:2106.02285 (2021)
[17] Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T.: Local implicit grid representations for 3d scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6001–6010 (2020)
[18] Kohli, A.P.S., Sitzmann, V., Wetzstein, G.: Semantic implicit neural scene representations with semi-supervised training. In: 2020 International Conference on 3D Vision (3DV). pp. 423–433. IEEE (2020)
[19] Lahoud, J., Ghanem, B., Pollefeys, M., Oswald, M.R.: 3d instance segmentation via multi-task metric learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9256–9266 (2019)
[20] Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7345–7353 (2019)
[21] Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. arXiv preprint arXiv:2008.02268 (2020)
[22] Matl, M.: Pyrender (2020), https://github.com/mmatl/pyrender, (Version 0.1.43)
[23] Meng, H.Y., Gao, L., Lai, Y.K., Manocha, D.: Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8500–8508 (2019)
[24] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
[25] Michalkiewicz, M., Pontes, J.K., Jack, D., Baktashmotlagh, M., Eriksson, A.: Deep level sets: Implicit surface representations for 3d shape inference. arXiv preprint arXiv:1901.06802 (2019)
[26] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. arXiv preprint arXiv:2003.08934 (2020)
[27] Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019)
[28] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: European Conference on Computer Vision (ECCV). Springer International Publishing, Cham (Aug 2020)
[29] Pham, Q.H., Nguyen, T., Hua, B.S., Roig, G., Yeung, S.K.: Jsis3d: joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8827–8836 (2019)
[30] Poursaeed, O., Fisher, M., Aigerman, N., Kim, V.G.: Coupling explicit and implicit surface representations for generative 3d modeling. arXiv preprint arXiv:2007.10294 **2** (2020)
[31] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
[32] Sitzmann, V., Chan, E.R., Tucker, R., Snavely, N., Wetzstein, G.: Metasdf: Meta-learning signed distance functions (2020)

[33] Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions (2020)
[34] Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2437–2446 (2019)
[35] Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: Advances in Neural Information Processing Systems. pp. 1121–1132 (2019)
[36] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. Advances in Neural Information Processing Systems **33** (2020)
[37] Tange, O.: Gnu parallel - the command-line power tool. ;login: The USENIX Magazine **36**(1), 42–47 (Feb 2011). https://doi.org/http://dx.doi.org/10.5281/zenodo.16303, http://www.gnu.org/s/parallel
[38] Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Stoll, C., Theobalt, C.: Patchnets: Patch-based generalizable deep implicit 3d shape representations. In: European Conference on Computer Vision. pp. 293–309. Springer (2020)
[39] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
[40] Xie, Y., Tian, J., Zhu, X.: A review of point cloud semantic segmentation. IEEE Geoscience and Remote Sensing Magazine (GRSM) (2020)
[41] Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In: Advances in Neural Information Processing Systems. pp. 492–502 (2019)
[42] Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016)
[43] Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.J.: In-place scene labelling and understanding with implicit scene representation. arXiv preprint arXiv:2103.15875 (2021)

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	Towards generalising neural implicit representations
Publication Status	X Published □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Theo W Costain and Victor Adrian Prisacariu. Towards generalising neural implicit representations. In European Conference on Computer Vision Workshop, 2022

Student Confirmation

Student Name:	Theo W. Costain		
Contribution to the Paper	• Conception and validation of the ideas • Designed, and wrote the code • Performed the experiments • Researched prior work • Wrote the manuscript • Presented material at the workshop		
Signature		Date	18/9/23

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 Victor Adrian Prisacariu		
Supervisor comments	I support the student's assesment of their contributions to this work.		
Signature		Date	18/9/23

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

5.3 Conclusion

This chapter considered our work *Towards Generalising Neural Implicit Representations*. We suggest that our experiments identify a crucial flaw in common uses of NIRs and demonstrate a simple approach to its mitigation.

We demonstrated that when NIRs are trained for reconstruction tasks only, the shape encodings they learn do not provide enough semantic information for use in later semantic tasks. However, we show that the addition of just a single semantic task alongside reconstruction tasks during training admits encodings that recover full performance (when compared to the relevant baseline), and even maintain this performance when used for "unseen" semantic tasks.

6

Contextualising Implicit Representations for Semantic Tasks

Contents

6.1 Introduction

Whereas in the previous chapter we outlined the shortcomings of neural implicit representations (NIRs) when trained for reconstruction tasks alone, the joint training solution proposed is less than ideal. We demonstrated that joint training could resolve these shortcomings, but this solution is relatively expensive and ideally, given the shape encodings ought to contain all the information necessary for semantic tasks, we would like to extract this information without needing to alter how the encodings are generated. Further, given that densely labelling 3D data is extremely expensive, having to provide dense labels for *all* of the training data may be prohibitively expensive.

To address this, we present our work *Contextualising Implicit Representations for Semantic Tasks*, which removes the need for joint training when using NIRs for semantic tasks. We achieve this using our lightweight contextualising module, which is trained separately to the shape encoder, and extracts meaningful semantic information hidden in the shape encodings, producing a "context" encoding that can be combined with the original shape encoding to provide full performance in semantic tasks. With our approach, it becomes possible to train NIRs on large unlabelled datasets to provide superior reconstruction performance, and then apply this learnt encoder to smaller labelled datasets without the need for potentially costly retraining of the shape encoder.

6.1.1 Contributions

- We present our simple lightweight contextualising module that reveals the meaningful semantic information hidden in shape encodings when they are learnt from reconstruction tasks alone.
- Using this new module, in contrast to existing works, we are able to separate training of reconstruction and semantic tasks.
- This separation not only allows us to avoid potentially expensive retraining of the encoder, but importantly allows us to train on large unlabelled datasets, to drive high quality general reconstruction, before training the lightweight contextualising module on smaller unlabelled datasets.
- We validate both the effectiveness as well as the formulation of our method on experiments on the ScanNet [20], 2D-3D-S [188], and SceneNN [189] datasets.

Contextualising Implicit Representations for Semantic Tasks

Theo W. Costain
Active Vision Lab
University of Oxford
costain@robots.ox.ac.uk

Kejie Li
Active Vision Lab
University of Oxford
kejie@robots.ox.ac.uk

Victor A. Prisacariu
Active Vision Lab
University of Oxford
victor@robots.ox.ac.uk

Abstract

Prior works have demonstrated that implicit representations trained only for reconstruction tasks typically generate encodings that are not useful for semantic tasks. In this work, we propose a method that contextualises the encodings of implicit representations, enabling their use in downstream tasks (e.g. semantic segmentation), without requiring access to the original training data or encoding network. Using an implicit representation trained for a reconstruction task alone, our contextualising module takes an encoding trained for reconstruction only and reveals meaningful semantic information that is hidden in the encodings, without compromising the reconstruction performance. With our proposed module, it becomes possible to pre-train implicit representations on larger datasets, improving their reconstruction performance compared to training on only a smaller labelled dataset, whilst maintaining their segmentation performance on the labelled dataset. Importantly, our method allows for future foundation implicit representation models to be fine-tuned on unseen tasks, without retraining the encoder.

1. Introduction

The explosion of interest in augmented reality in recent years has spurred a renewed search for more efficient representations of 3D data. Whilst point-clouds, meshes, and various other representations have been proposed over the years, the recent introduction of implicit representations like NeRF and DeepSDF have reignited interest in the *representation* rather than the processing of the data.

Opposed to "classical" representations that discretise the underlying structure, implicit representations (IRs) learn a continuous function over 3D space. IRs are able to represent the structure at arbitrary resolutions, trading spatial complexity for time complexity required to extract the structure from the representation. In the most current approaches, an encoder takes input in one or more modalities producing an encoding that is used to condition an MLP that composes the function. Early works, such as DeepSDF [35] and Occupancy Networks [31], learned functions that separated space into "inside" and "outside" regions, however, this ensured that the network could only learn closed surfaces. Subsequently, methods were proposed that resolved this limitation by learning an unsigned distance function (UDF), where the surface of the object lies on the zero level set of the function. More work followed, and improved on various aspects of these approaches including training ambiguities [55, 59] and extraction/rendering [59] (further discussion in Sec. 2), but despite this, relatively little attention was given to applications of these approaches in conventional pipelines, such as semantic segmentation or classification.

Foundation models, *i.e*. large pre-trained generalist networks and models (*e.g*. [16, 34]), that are trained on vast amounts of data, allowing adaptation to a variety of downstream tasks, are increasingly an essential building block in deep learning pipelines. It is not impossible that (as is already beginning to happen [34]) foundation like IR encoders may not be feasible or possible for most users to train from scratch, or even fine-tune encoding network, especially when the original training data is available (as is increasingly less common). Given Costain and Prisacariu [8] demonstrated that, when trained for reconstruction tasks only, IRs learn encodings that are not necessarily meaningful for semantic tasks. Accordingly, without the ability to train (or fine-tune) the encoder with semantic supervision [50], the performance on semantic tasks, using these encodings is

likely to be unsatisfactory.

To address this problem, we propose a novel method to contextualise the encodings learnt by networks supervised on reconstruction tasks alone, even when the original reconstruction training data (*i.e*. ground truth distance function information) is not available. Basing our experiments on the approach of Wang et al. [50], we show the semantic limitations of encodings generated by training on reconstruction tasks alone. Then we propose our lightweight contextualising module that takes the learnt encoding and produces a small additional context encoding. This context encoding can then be combined with the existing encoding allowing the network to completely recover performance on the semantic tasks.

By separating the geometric tasks from the semantic tasks, our approach allows the geometric pipeline to be trained on much cheaper to produce datasets where complete semantic labels are not available, before our contextualising module is applied to a smaller fully labelled dataset enabling semantic segmentation alongside reconstruction. Rather than complex approaches [26], our method presents a simple, but effective and performant approach that address a major shortfall in existing implicit representation approaches.

Our key contributions are:

- Our contextualising module which reveals hidden semantic information contained in the feature encodings of IR.
- A novel and simple approach to train existing implicit representations for unseen semantic tasks without access to the original training data.

In the rest of this paper we cover: relevant existing works in the literature Section 2, our method and contextualising module Section 3, details of our experimental setup Section 4, the results of our experiments Section 5, and finally the limitations of our approach Section 6.

2. Related Work

Early IR works [2, 5, 13, 31, 32, 35, 38] focused mainly on reconstructing single objects. Both Occupancy Networks [31] and IM-Net [5], learn a function mapping from points in space to the probability that point lies within the object to be reconstructed. Occupancy Networks further proposed a hierarchical, octree based, extraction method to efficiently extract the mesh. In contrast, DeepSDF [35] learns a function mapping from space to a signed distance function. Although they proposed an encoder-decoder structure, they also introduced an auto-decoder structure, where the representation encoding is found by freezing decoder and optimising the encoding/embedding. Scene Representation Networks (SRN) [44] proposed a "Neural Renderer" module, which maps from 3D world coordinates to a feature representation of the scene at that location. Sign Agnostic Learning (SAL) [2] proposed to remove the need for signed ground truth information, whilst still learning a signed distance function. Crucial to this effort is an initialisation scheme that the initial level set was approximately a sphere of some chosen radius. Gropp et al. [13] introduce the Eikonal Loss term amongst other improvements to the loss function from SAL [2]. These new terms encourage the representation to develop a unit norm gradient, like a metric SDF, and acts as a geometric regularisation over the learned function, improving smoothness and accuracy of the reconstructions. Later methods [29] include approaches to better allow networks to represent high frequency information [45, 46], the former of which is vital to the performance of NeRFs [4, 30, 33].

These early works focused on single object reconstruction, with typically a single encoding or embedding per object. This limits the scale of objects these representations could represent, a concern later works proposed several solutions to. Many works arrived at a similar solution to this problem, using either planes [37] or grids [3, 6, 21, 37], to improve both the scale and detail of the reconstructions. Convolutional Occupancy Networks [37] make use of planes or grids of features, and IF-Net [6] learns learns a hierarchy of multi-scale features, both interpolating between these features at queried locations to predict occupancy probabilities and signed distance function respectively. Rather than interpolating features, Deep Local Shapes [3] and Jiang et al. [21], learn a grid of encodings, dividing scenes into small simple geometric shapes.

All the above methods share a common trait in separating space into inside *vs*. outside, however in the case where watertight meshes are not available (as is the case for common 3D Datasets [1, 9, 19]) training is not possible without complicated pre-processing, or learning overly thick walls. Chibane et al. [7] addressed this issue by learning an UDF as well as proposing a gradient based rendering scheme to extract the surface, a requirement given Marching Cubes [28] cannot be applied to UDFs. Various works followed in this vein [14, 47, 50, 55, 56, 59]. Notably, Guillard et al. [14], Zhou et al. [59] who independently proposed an approach to significantly improve the extraction/rendering of UDFs, by modifying Marching Cubes to look for diverging gradients rather than zero crossings allowing its use on UDFs.

A number of works have considered semantic tasks alongside NeRFs [24, 49, 51, 54, 58], however far fewer works [8, 23, 29, 50] consider semantic tasks alongside IRs. Costain and Prisacariu [8] argued that training IRs on geometric tasks alone produce encodings that are poor for semantic tasks. However, Luigi et al. [29] show that these encodings still contain the semantic information, and that it is possible to transform these encodings into a form that are more meaningful for semantic tasks. We leverage this insight in designing our contextualising module. Wang et al. [50], as well as proposing a UDF based IR, train a "surface-aware" segmentation branch alongside the UDF.

As a fundamental problem in computer vision, a vast array of works [10, 12, 15, 17, 18, 20, 25, 27, 39–43, 48, 52, 53, 57] have tackled semantic segmentation of point clouds, however a detailed discussion of these methods falls outside the scope of this work.

3. Method

Implicit representations seek to learn a functional mapping, f, from a query point, $q \in \mathbb{R}^3$, in space to the distance from that query point to the nearest point on the surface being represented. In this work we consider UDFs, further constraining $f : q \in \mathbb{R}^3 \mapsto \mathbb{R}_0^+$. In this work, we use UDFs, as these are the currently preferred way to represent non watertight scenes, however, this should not affect the generality of the approach, and should a dataset containing large watertight scenes be released, we expect our method should also apply. This function is typically implemented as a simple MLP, but to avoid overfitting the MLP to every surface to be represented, it is often desirable to condition the function on some global [31, 35] or local [7, 50] encoding of shape, giving $f : q \in \mathbb{R}^3, E \in \mathbb{R}^d \mapsto \mathbb{R}_0^+$, where E is some encoding vector and d its dimension.

As the only method that performs semantic tasks alongside learning implicit representations for large scenes, we use the RangeUDF method proposed by Wang et al. [50] as our baseline for our work. Their approach takes a sparse input point-cloud $P \in \mathbb{R}^{N\times 3}$, where N is the number of points, and uses an encoder to learn some encoded features, $E_g \in \mathbb{R}^{N\times d}$. These encoded features are then passed to the decoder(s), alongside a set of query points $Q \in \mathbb{R}^{M\times 3}$ (where M is the number of query points). KNN is used to collect the K nearest encoding vectors for each query point $q \in Q$, which are then combined using a simple attention module. These combined features, alongside the corresponding query point, are then fed into the UDF and semantic segmentation modules.

3.1. The Problem

A common pipeline in computer vision tasks is to take a pre-trained model, that produces meaningful features for a given task, and either fine-tune it, or use the generated features as input to another module that performs some desired task. The arc of research so far has resulted in this pre-training often [16] taking the form of classification tasks on extremely large datasets [11], ensuring these pre-trained models learn features that are semantically meaningful.

On the other hand, IR methods have arisen to tackle a different challenge: the representation of 3D shape and structure. Typically, this is in service of reducing the memory required to represent a given scene or object at high resolutions [31, 35, 37], compared to other conventional representations such as point-clouds, meshes, or voxel grids. Whilst much of the research on implicit representations to date has focused on the reconstruction task alone, there has been little consideration of how IRs might be used to replace conventional representations in existing pipelines, such as performing classification or segmentation, and other works [8] have shown that encodings learnt for reconstruction alone can provide insufficient information for semantic tasks.

When training the encodings that condition a UDF, the desire is for the network to learn some set of encodings E that holistically represent local structural information about the underlying shape. Our experiments at the beginning of Sec. 5, confirm [8] that the encodings, E_g, learnt when training the UDF for geometric reconstruction alone, show poor separability in semantic space (Fig. 2a). Whilst this can obviously be addressed by training for both semantic and reconstruction tasks jointly [8, 50], it is trivial to imagine scenarios where it is extremely desirable to be able to fine-tune on semantic tasks, without requiring either access to the original training data (original training data may not be publicly available), or having to potentially expensively retrain the *entire* pipeline. To address this, our contextualising module allows for the *effective* training of segmentation despite fixing the encoder/encodings. It also bears noting that whilst it is possible to exhaustively render/extract each scene from the encoding and UDF, and use this to re-create the training data, current implicit representation methods are far from perfect, and so taking this approach would almost certainly compound errors (akin to repeated photocopying of a paper document), not to mention the substantial cost of labelling the extracted scenes. Our proposed method avoids this entire problem, with a simple process.

3.2. Contextualising Module

The results of Zhou et al. [59] suggest that although not necessarily in present in a separable form, the semantic information is still present in the representation. Accordingly, we propose our simple contextualising module, which produces *compact* context features that carry substantial semantic information (Fig. 2b), that when combined with the original encoded features, produces features useful for semantic segmentation as well as reconstruction. An overview of our method is presented in Fig. 1.

Taking the encoded features, our module uses a small encoder-decoder UNet-like network, specifically PointTransformer [57], to re-capture semantic information present in the encoding. Important to its function is the contextualising modules ability to consider wider shape context, than either the UDF or segmentation decoder, which has repeatedly been shown as vital to capturing semantic information [39, 40]. This re-capturing of a wider scene context gives rise to the naming of our module. This is achieved through the Point-Transformer's downsampling, interpolation, and upsampling performed across 5 different scales (similar to [40]).

The formulation of RangeUDF [50] which predicts the

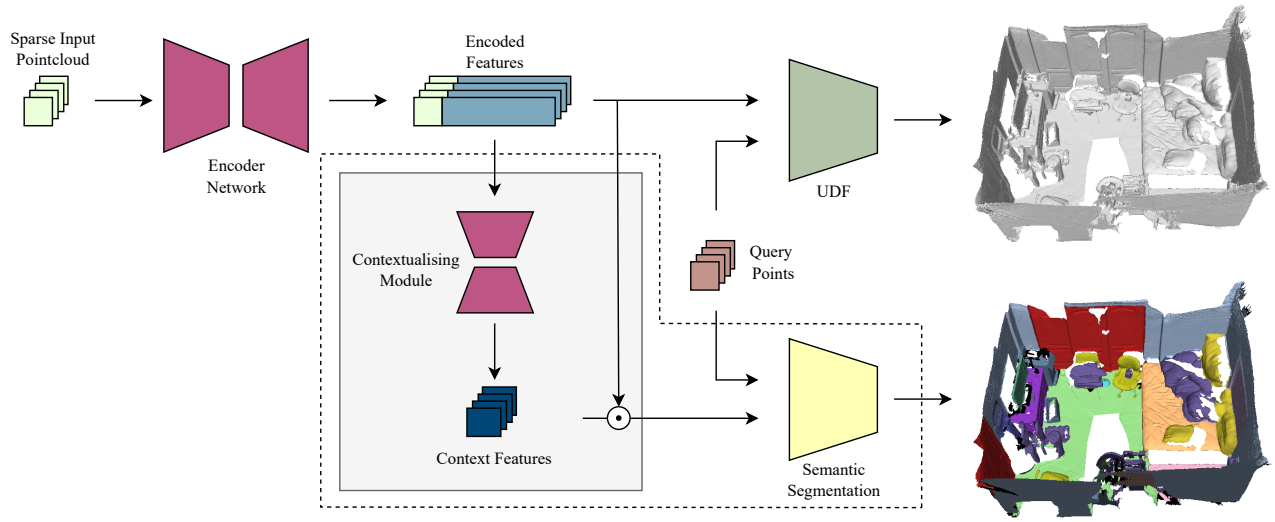
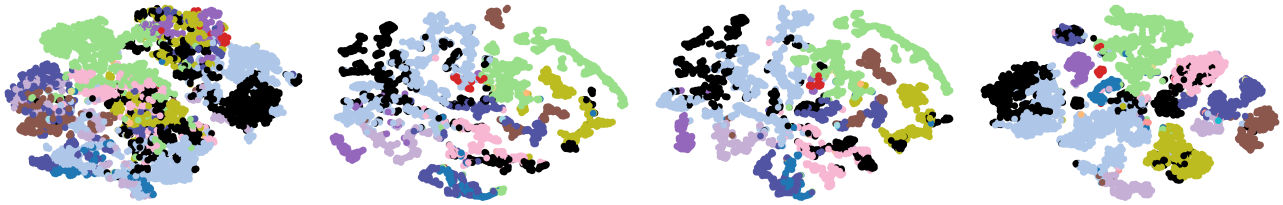


Figure 1. Our proposed contextualising module (light grey box) allows already trained implicit representations, to be fine-tuned (training only elements inside the dotted line) on semantic tasks without the need for the original training data. Learning a compact contextualising vector which is concatenated with the original encoding, our module allows full semantic performance to be recovered from encoders trained only on reconstruction tasks.



(a) Geometry trained (b) Context features (c) Concatenated geometry and context (d) Joint semantic & reconstruction trained

Figure 2. t-SNE embeddings of the features of an encoding of a particular scene. The features trained for geomery tasks only, show poor separability according to semantic label. Our proposed contextualising module produces context features that are clearly more separable, and become even more so when combined with the existing features. Figure best viewed in colour.

semantic class, $s_i \in \mathbb{R}^C$ where C is the number of classes, of a given point $q_i \in Q$ as

$$s_i = f_{\text{sem}}(q_i | E_g) \tag{1}$$

Instead, our contextualising module, f_{ctx}, takes the fixed encoded features (trained on only the reconstruction task), $E_g \in \mathbb{R}^{M \times d}$, and predicts a set of context features, $E_c \in \mathbb{R}^{M \times l}$. We then concatenate the context features with the original encoded features to give the semantic features, $E_s \in \mathbb{R}^{M \times (d+l)}$, which we feed into the segmentation module alongside the query points, giving instead

$$s_i = f_{\text{sem}}(q_i | E_g \oplus f_{\text{ctx}}(E_g)) = f_{\text{sem}}(q_i | E_s) \tag{2}$$

where $\oplus$ represents concatenation in the feature dimension.

Despite the simplicity of our contextualising module and its implementation, our results demonstrate the performance improvements it provides to the semantic task are substantial. Our contextualising module is implemented as a substantially shrunk version of the PointTransformer, reducing the number of parameters from roughly 7.8 million to around 379,000. This is achieved through a reduction of the number of channels at each scale from $[32, 64, 128, 256, 512]$ to $[32, 32, 64, 64, 128]$ and reducing the number of "blocks" at each scale to 1.

During training, we use the L1 loss, with the same clamping as Chibane et al. [7], for supervising the reconstruction task, and the standard cross entropy loss for the segmentation task. Our method focuses on mainly on separately training each task, in which case, our loss contains only a single objective, avoiding the need to balance loss terms entirely. However, in the case of the joint training baseline, following [50] we use the uncertainty loss [22] to avoid manually tuning loss weightings between the semantic and reconstruction tasks.

4. Experiments

In this section, we cover details of the datasets and metrics used in our experiments, as well as the relevant details of our implementations and the resources used to perform our experiments.

4.1. Datasets & Metrics

We train and evaluate our method on three datasets: ScanNet [9], SceneNN [19], and 2D-3D-S [1], all three of which are captured using RGB-D cameras.

ScanNet The ScanNet dataset consists of 1613 scans of real-world rooms. The data is split into 1201 scans for training and 312 scans for validation with a further 100 scans held out for online benchmarks. Following Wang et al. [50], we use the validation set for testing as ground truth annotations for the test set are not publicly available. Semantic labels are provided for 40 classes, however following other methods [18, 27, 39, 40, 48, 50, 52], we train and test on only the 20 class subset used in the online benchmark.

2D-3D-S The 2D-3D-S dataset consists of 6 *very* large-scale indoor scans, capturing rooms, hallways and other educational and office like environments using an RGB-D The data is divided into a total of 271 rooms, divided into 6 "Areas" based on the scan they are contained in. Area 5 is split into two scans without a provided registration between them, preventing their use in the data preparation pipeline described below. Following Wang et al. [50], we use Areas 1-4 for training and Area 6 for testing. The semantic labels are provided for 13 classes.

SceneNN The SceneNN dataset consists of 76 indoor scans divided into 56 scenes for training and 20 scenes for testing [20]. Semantic labels are provided for the same 40 classes as ScanNet, where again we use the 20 class subset.

4.1.1 Data Preparation

We follow the same processing steps as Chibane et al. [7], normalising each scene's mesh to a unit cube, and sampling 10k surface points (for the encoder input) and 100k off surface points for which we compute the distance to the closest point on the surface.

4.1.2 Metrics

When evaluating the reconstruction tasks, we use the standard Chamfer L1 & L2 distance measures (lower is better) as well as the F1-δ and F1-2δ score (higher is better). All CD-L1 values are reported $\times 10^{-2}$ and CD-L2 values $\times 10^{-4}$, and we set $\delta = 0.005$. For the segmentation task, we use mean Intersection-over-Union (higher is better) as well as mean F1-δ, which is calculated by determining the per-class F1-δ score then averaging over the classes.

Whilst we report both mF1-δ and mIOU for the semantic tasks, we suggest that more attention should be paid to the mF1-δ, as it better captures performance in the join task given that the IOU metric does not consider reconstruction performance at all. And the more extreme class imbalance present in the smaller SceneNN dataset can lead to greater instability and noise in this metric during evaluation.

At test time, like others [7, 50], we extract a mesh from the implicit representation, as well as semantic labels for 100k points on the surface of that mesh. We then compute the chamfer and F metrics against 100k points sampled directly from the ground truth meshes.

4.2. Implementation Details

We implement our work in PyTorch [36], and perform our experiments on 3 Nvidia RTX6000 GPUs and an Intel Xenon Gold 6226R CPU. We use the Adam optimiser with default parameters and a learning rate of 10^{-3} for all experiments, we use a batch size of 12, and set the dimension of the context features to 4. During training, we feed 10,240 points into the encoder, and 50k points to the UDF and segmentation decoder. For experiments on ScanNet, we train the model for 500 epochs. For both 2D-3D-S and SceneNN, we train for 1k epochs.

For the encoder network, we use the PointTransformer [57] network. To drastically speed up the evaluation of our experiments, we use the surface extraction algorithm from Zhou et al. [59] rather than Algorithm 1 from Chibane et al. [7].

5. Results

Comparing reconstruction-only trained features with our contextualised features We start with our experiments confirming that the findings of [8] apply to larger implicit representations. For our baseline, given no source code is publicly available, we use our implementation (Sec. 4.2) of RangeUDF [50] (the current SOTA method for joint reconstruction and segmentation), jointly training reconstruction alongside segmentation, as in the original paper. To confirm the findings, we train only the encoder network and UDF on the reconstruction task for a given dataset (2$^{\text{nd}}$ row Tabs. 1a to 1c), which we refer to as "Geometric Only". Then, freezing the encoder network and UDF (with these reconstruction only features), we train the segmentation decoder on the semantic labels of the same dataset, using the frozen encodings (3$^{\text{rd}}$ row Tabs. 1a to 1c), which we refer to as "Frozen Encoder". Its clear from the mIOU and mF1 scores that the frozen encodings are insufficient for reasonable quality segmentation results, with the frozen encodings giving less than half the performance of the baseline jointly trained model in the case of ScanNet. We also again train the segmentation decoder on the semantic labels, but provide no geometric supervision and do not freeze the encoder (4$^{\text{th}}$ row Tabs. 1a

	L1 (↓)	F1-δ (↑)	mF1-δ (↑)	mIOU (↑)
Baseline	0.321	0.861	0.662	0.724
Geometric Only	0.302	0.884	-	-
Frozen Encoder	0.298	0.888	0.280	0.296
Unfrozen Encoder	0.768	0.506	0.587	0.744
Context Features	0.297	0.889	0.640	0.694

(a) ScanNet

	L1 (↓)	F1-δ (↑)	mF1-δ (↑)	mIOU (↑)
Baseline	0.859	0.603	0.604	0.692
Geometric Only	0.832	0.633	-	-
Frozen Encoder	0.864	0.624	0.575	0.600
Unfrozen Encoder	1.18	0.430	0.532	0.657
Context Features	0.865	0.629	0.608	0.681

(b) SceneNN

	L1 (↓)	F1-δ (↑)	mF1-δ (↑)	mIOU (↑)
Baseline	0.364	0.819	0.695	0.727
Geometric Only	0.389	0.822	-	-
Frozen Encoder	0.358	0.837	0.458	0.435
Unfrozen Encoder	0.971	0.31	0.359	0.734
Context Features	0.357	0.838	0.684	0.700

(c) 2D-3D-S

Table 1. Comparison of semantic segmentation and geometric reconstruction, on three datasets. The rows from top to bottom: joint training baseline, geometry reconstruction supervision only, semantic training on frozen encodings from geometry only, semantic training on frozen encodings with our contextualising module.

	L1 (↓)	F1-δ (↑)	mF1-δ (↑)	mIOU (↑)
ScanNet Labels	0.297	0.889	0.639	0.694
SceneNN Labels	0.344	0.836	0.584	0.621
Stanford Labels	0.328	0.861	0.692	0.693

(a) ScanNet trained geometric features

	L1 (↓)	F1-δ (↑)	mF1-δ (↑)	mIOU (↑)
ScanNet Labels	0.781	0.681	0.563	0.65
SceneNN Labels	0.793	0.631	0.562	0.648
Stanford Labels	0.740	0.626	0.5578	0.679

(b) SceneNN trained geometric features

	L1 (↓)	F1-δ (↑)	mF1-δ (↑)	mIOU (↑)
ScanNet Labels	0.357	0.852	0.623	0.690
SceneNN Labels	0.379	0.806	0.611	0.648
Stanford Labels	0.357	0.838	0.684	0.700

(c) 2D-3D-S trained geometric features

	L1 (↓)	F1-δ (↑)	mF1-δ (↑)	mIOU (↑)
ScanNet Labels	0.299	0.887	0.634	0.690
SceneNN Labels	0.343	0.837	0.593	0.631
Stanford Labels	0.325	0.863	0.728	0.729

(d) Triad trained geometric features

Table 2. Cross training and validation using our contextualising module. For each table, we use the fixed feature encodings trained on reconstruction only for one dataset, and then train for segmenation with our contextualising module on each of the datasets. Triad represents the amalgamation of all three of the datasets.

to 1c), referred to as "Unfrozen Encoder". Whilst obviously unfair, this experiment demonstrates that although, as you would expect, semantic performance can be recovered, significant reconstruction capability is forgotten in the process.

Finally, we train the segmentation decoder with the frozen encodings combined with the context features produced by our contextualising module (5^{th} row Tabs. 1a to 1c), which we refer to as "Context Features". The mIOU and mF1 scores show that our contextualising module allows nearly full performance on the segmentation task to be recovered.

Qualitative results are shown in Fig. 3, where the middle left shows the baseline results, and middle right shows the frozen encoder results, and the far right shows the results using the context features. Whilst bulk areas are easily segmented with the frozen encodings, there is significant confusion and error with smaller objects, and structurally similar surfaces are misclassified (*e.g.* in the 2^{nd} row, the orange beam in the left corner and yellow blinds on the back wall are entirely missed by the fixed encoding).

Cross Training & Validation One of the key advantages of our method is that it allows for the separation of training for reconstruction tasks and semantic tasks without compromising on the performance of either. To evaluate this, we cross-train fixed reconstruction-only trained feature encodings (geometric only) with our contextualising module on each of the datasets (*i.e.* We train for reconstruction only on ScanNet and then for semantics on *etc.*).

We also train an additional set of fixed encodings on the amalgamation of the three datasets, which we refer to as the Triad dataset. To preserve train-test splits, the train split for Triad is sum of the three training splits, and likewise for the validation splits.

Our results in Tab. 2, specifically the mF1-δ score (which best describe the joint task performance), demonstrate that our method not only allows cross training between different datasets for reconstruction and semantic tasks, but *importantly*, our results in the 2^{nd} and 3^{rd} rows in Tab. 2d show that by leveraging the ability to train for reconstruction on larger datasets and then then semantics on a different smaller dataset, we can maintain the same semantic performance as a jointly trained baseline, whilst improving the quality of the reconstructions generated.

Whilst the triad dataset provides meaningful improvement to the semantic results (when the geometric features trained on the Triad are used to train the contextualising mod-

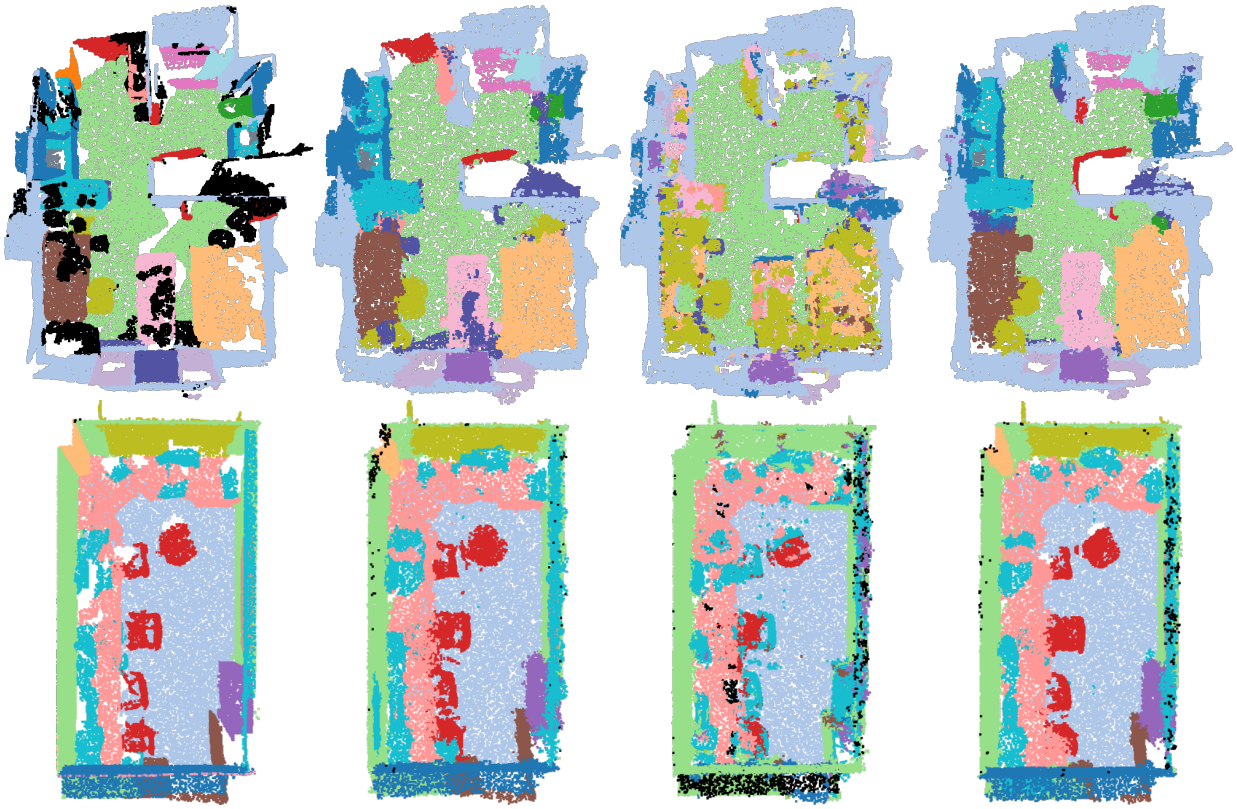


Figure 3. Qualitative comparison of segmentation and reconstruction on the ScanNet dataset. From left to right: Ground truth (semantics and geometry), jointly trained geometry and segmentation, segmentation training on frozen encodings, and finally segmentation training on frozen encodings using our contextualising module. The frozen encodings trained only for reconstruction seriously inhibit the network's performance on segmentation, misclassifying small objects, and entirely missing structurally similar classes (*e.g.* yellow curtain on back wall in 2nd row.)

ule on both the SceneNN and Stanford datasets) the same improvement is not seen for the results on ScanNet. We suggest that this arises from the relative sizes of the datasets, as the Triad dataset os only 30% larger than ScanNet, but is over $\times 5$ larger than the Stanford dataset and over $\times 27$ larger than SceneNN. As a result, whilst this means that a much broader feature space may be learned on the triad dataset compared to Stanford or SceneNN, on ScanNet this new feature space is arguably not meaningfully larger.

5.1. Ablations

To validate our design of the contextualising module and its compactness, we perform ablation experiments on the structure of both our contextualising module, as well as the context features we generate. For our ablation experiments, we use the 2D-3D-S dataset, all parameters are kept the same as in the above experiments except for the modifications described below.

In our experiments (Tab. 3), we evaluate the following:

	mF1-δ (↑)	mF1-2δ (↑)
MLP Context Module	0.490	0.568
Shallower network (3 scales)	0.644	0.745
More blocks	0.669	0.768
More channels	0.662	0.763
More blocks & channels	0.664	0.765
$l = 2$	0.606	0.706
$l = 1$	0.566	0.662
Ours	0.664	0.763

Table 3. Results of our ablation experiments on the 2D-3D-S dataset. For the experiments using more blocks we increase the number of blocks from $[1, 1, 1, 1, 1]$ to $[1, 2, 3, 5, 2]$. For the experiments using more channels, we increase the number of channels at each resolution scale from $[32, 32, 64, 64, 128]$ to $[32, 64, 128, 256, 512]$. l refers to the dimension of the context feature.

Contextual information To confirm that information from across the whole feature encodings for a given shape is vital to our contextualising module, rather than individual feature vectors, we implement our contextualising module as an MLP first, and second as a shallower version of the PointTransformer normally used. Our results show that the MLP provides little to no advantage over the raw fixed encodings, and that whilst the shallower PointTransformer recovers some of the performance, there is still a gap in performance compared to the baseline. These results demonstrate the importance of capturing contextual information across the whole encoding in our proposed module, and that simple re-projection of the fixed encodings is insufficient.

Compactness To show that our contextualising module is as compact as possible whilst maintaining performance, we evaluate the effects of increasing the size of the contextualising module, either by increasing the number of blocks at each scale, or by increasing the number of channels at each scale, or both simultaneously. Whilst these provide very marginal increase to performance, they both (particularly increasing the number of blocks) substantially increase the number of parameters in the contextualising module. We also demonstrate that further reducing the dimension of the context features below 4 harms the performance of the contextualising module. However, this specific parametrisation applies only to the datasets we use, and may be different for more complex or simpler datasets.

6. Limitations

Although Initial convergence of semantic segmentation performance when training the context module is faster than the joint training baseline, 90% of the performance with 17% of the training time, full convergence is not materially faster than the baseline. We suspect, however, this slowness arises from the segmentation module proposed in Wang et al. [50], as training the encoder used to generate the encoded features on a simple segmentation task converges substantially faster.

Ultimately, the main limitation of our approach is that it requires labelled data to train the semantic branch. However, as our approach separates the training of the reconstruction and semantic tasks, it is theoretically possible to extract meshes from the decoders at a coarse scale, and then manually label them to train the network for semantic tasks. There are also a number of weaknesses that arise from the design original RangeUDF [50], however these improvements would not necessarily represent any novelty, rather incremental improvements that would improve numerical performance, such as replacing their scalar attention module with vector attention or adding positional encoding [46] to the decoders.

7. Conclusion

In this work, we propose a novel approach to training implicit representations for downstream semantic tasks without needing access to the original training data or retraining the encoding network. We introduce our contextualising module that reveals hidden semantic information contained in the encodings of implicit representations trained only for geometric tasks. We demonstrate our contextualising module on the task of semantic segmentation and show that without it, the encoded features learnt by implicit representations for geometric tasks alone lack sufficient separability to provide meaningful results. Finally, we show that using our module, it becomes possible to leverage larger unlabelled datasets to pre-train implicit representations and then fine-tune on smaller labelled semantic datasets, achieving higher reconstruction performance than would be possible with only the smaller labelled datasets.

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 2, 5

[2] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 2

[3] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020. 2

[4] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5799–5809, 2021. 2

[5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 2

[6] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. 2

[7] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 3, 4, 5

[8] Theo W Costain and Victor Adrian Prisacariu. Towards generalising neural implicit representations. *arXiv preprint arXiv:2101.12690*, 2021. 1, 2, 3, 5

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet:

Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 2, 5

[10] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2018. 3

[11] 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. 3

[12] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. 3

[13] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020. 2

[14] Benoît Guillard, Federico Stella, and Pascal Fua. MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks. In *Computer Vision – ECCV 2022*, pages 576–592. Springer Nature Switzerland, Cham, 2022. Series Title: Lecture Notes in Computer Science. 2

[15] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 3

[16] 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, 2016. 1, 3

[17] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, Yaonan Wang, Qiang Fu, Yanmei Zou, and Ajmal Mian. Deep learning based 3d segmentation: A survey. *arXiv preprint arXiv:2103.05423*, 2021. 3

[18] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020. 3, 5

[19] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 fourth international conference on 3D vision (3DV)*, pages 92–101. Ieee, 2016. 2, 5

[20] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 984–993, 2018. 3, 5

[21] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020. 2

[22] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 4

[23] Amit Pal Singh Kohli, Vincent Sitzmann, and Gordon Wetzstein. Semantic implicit neural scene representations with semi-supervised training. In *2020 International Conference on 3D Vision (3DV)*, pages 423–433. IEEE, 2020. 2

[24] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022. 2

[25] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018. 3

[26] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2

[27] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2020. 3, 5

[28] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. 2

[29] Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi di Stefano. Deep learning on implicit neural representations of shapes. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[30] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2

[31] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 2, 3

[32] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*, 2019. 2

[33] 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 *European conference on computer vision*, pages 405–421. Springer, 2020. 2

[34] OpenAI. Gpt-4 technical report, 2023. 1

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous

signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2, 3

[36] 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 *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[37] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020. 2, 3

[38] Omid Poursaeed, Matthew Fisher, Noam Aigerman, and Vladimir G Kim. Coupling explicit and implicit surface representations for generative 3d modeling. In *European Conference on Computer Vision*, pages 667–683. Springer, 2020. 2

[39] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3, 5

[40] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017. 3, 5

[41] Zhongwei Qiu, Kai Qiu, Jianlong Fu, and Dongmei Fu. Dgcn: Dynamic graph convolutional network for efficient multi-person pose estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11924–11931, 2020.

[42] Dario Rethage, Johanna Wald, Jurgen Sturm, Nassir Navab, and Federico Tombari. Fully-convolutional point networks for large-scale point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 596–611, 2018.

[43] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017. 3

[44] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32:1121–1132, 2019. 2

[45] 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, 2020. 2

[46] 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. 2, 8

[47] Jiapeng Tang, Jiabao Lei, Dan Xu, Feiying Ma, Kui Jia, and Lei Zhang. Sa-convonet: Sign-agnostic optimization of convolutional occupancy networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6504–6513, 2021. 2

[48] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019. 3, 5

[49] Suhani Vora*, Noha Radwan*, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi S. M. Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *Transactions on Machine Learning Research*, 2022. https://openreview.net/forum?id=ggPhsYCsm9. 2

[50] Bing Wang, Zhengdi Yu, Bo Yang, Jie Qin, Toby Breckon, Ling Shao, Niki Trigoni, and Andrew Markham. Rangeudf: Semantic surface reconstruction from 3d point clouds. *arXiv preprint arXiv:2204.09138*, 2022. 1, 2, 3, 4, 5, 8

[51] Qianyi Wu, Xian Liu, Yuedong Chen, Kejie Li, Chuanxia Zheng, Jianfei Cai, and Jianmin Zheng. Object-compositional neural implicit surfaces. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pages 197–213. Springer, 2022. 2

[52] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 3, 5

[53] Haotian Xu, Ming Dong, and Zichun Zhong. Directionally convolutional networks for 3d shape segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2698–2707, 2017. 3

[54] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. 2

[55] Xianghui Yang, Guosheng Lin, Zhenghao Chen, and Luping Zhou. Neural vector fields: Implicit representation by explicit learning. *arXiv preprint arXiv:2303.04341*, 2023. 1, 2

[56] Jianglong Ye, Yuntao Chen, Naiyan Wang, and Xiaolong Wang. Gifs: Neural implicit function for general shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12829–12839, 2022. 2

[57] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 3, 5

[58] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. *arXiv preprint arXiv:2103.15875*, 2021. 2

[59] Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Learning consistency-aware unsigned distance functions progressively from raw point clouds. In *Advances in Neural Information Processing Systems*, 2022. 1, 2, 3, 5

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	Contextualising implicit representations for semantic tasks
Publication Status	□ Published □ Accepted for Publication X Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Theo W Costain, Kejie Li, and Victor Adrian Prisacariu. Contextualising implicit representations for semantic tasks. In manuscript style

Student Confirmation

Student Name:	Theo W. Costain		
Contribution to the Paper	• Conception and validation of the ideas • Designed, and wrote the code • Performed the experiments • Researched prior work • Created all of the tables and diagrams • Wrote the manuscript		
Signature		Date	18/9/23

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 Victor Adrian Prisacariu		
Supervisor comments	I support the student's assesment of their contributions to this work.		
Signature		Date	18/9/23

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

6.3 Conclusion

In this chapter, we have presented our work *Contextualising Implicit Representations for Semantic Tasks*, which seeks to provide an approach to use implicit representations for semantic tasks without having to re-train the shape encodings.

We proposed a lightweight contextualising module that extracts meaningful semantic clues hidden in shape encodings learnt on reconstruction tasks alone, and use this module to allow for learning of semantic tasks without the need to retrain the shape encodings or encoder.

Our approach and design is validated by our experiments on the ScanNet [20], 2D-3D-S [188], and SceneNN [189] datasets, and further, with our experiments on our "Triad" dataset, we demonstrate how our method can be used to train NIRs on large unlabelled datasets to gain superior reconstruction performance, before using our method to apply them to semantic tasks.

7 Conclusion

In each chapter where we have presented our work, we have finished with a conclusion that summarises the key points and achievements of our work. With this final chapter, we will again briefly cover the accomplishments of each chapter, but focus on a broader discussion of the work including a discussion of avenues for future exploration.

7.1 Achievements and Future Work

Non-Uniform Quantisation Schemes In Chapter 3, we proposed an approach to automatically discovering optimal non-uniform quantisation schemes for deep networks using using a bayesian optimization (BO) based neural architecture search (NAS), which provides for a more sample efficient search than comparable reinforcement learning (RL) methods. Using parametric functions to describe the quantisation schemes, we simplify the search space, and make use of Multi-Task Gaussian processs (GPs) to predict the performance of a given scheme, minimising the amount of searching/exploration. Our method is able to find non-uniform schemes that are as performant as uniform schemes whilst using less memory, and we demonstrate this on experiments in both 2D (as is common in the literature) as well as 3D settings.

The work presented is motivated by the efficiency improvements to both memory and energy that are possible when hardware capable of performing quantised computations is available, and further, by observations from others that the overall accuracy of a deep network is more dependent on some layers than on others.

Although effective, there are some limitations to our method. In comparison to training times of common deep networks, sampling from the GP is very quick, but on smaller networks and datasets, sampling can become the dominant time sink in our implementation. Further, the choice about how many samples the GP should take before predicting optimal configurations is manually set, and the number of samples required may vary substantially between network architectures.

Turning to future work, the use of transformers [128] is increasingly popular in both 2D and 3D settings, and most of the state-of-the-art (SOTA) methods [129; 130; 131] on common 3D tasks have exploited them. In the case of 3D tasks, the cubic memory growth means that dense voxel grids have seen little interest, and instead most work has focused on other representations (*i.e.* point-clouds and meshes), to which transformers can readily be applied. Given that transformer architectures typically consist of deeply stacked repetitions of the same building block, they should be well suited for exploration using our method, provided a suitable quantisation method can be found.

Weight Compression by Approximation With Chapter 4, we demonstrated how through the use of functional approximations like cosine and Chebyshev series, the values of a deep network layer's weights can be compressed, reducing the memory footprint. We motivated this by discussion of the importance of model memory size to energy consumption in compute constrained environments, where quantisation hardware is less likely to be available, and demonstrated the effectiveness of our approach on a variety of networks, both old [29] and new [59].

The main limitation of our method is that its effectiveness is closely related to both the "size" of the kernels in the network, as well as the distribution of learnt parameters, *i.e.* it is less effective when one weight in a filter is substantially

larger than the others (as is the case for many layers in the ConvNeXt [59] networks we investigated).

In the method as presented, we use the same number of "harmonics" for every filter in the kernel. One possible avenue for future work, would be to explore the possibility of using a different number of harmonics for different filters in the same kernel. Whilst this would substantially increase the number of hyperparameters needed to fully tune the method, it might be possible to automate this by formulating a cost function that trades off between the reduction in memory, and the approximation error between the series function and the original weights. Further, in the same vein as the above, the approximations learnt are isotropic, however, we observed there are many kernels that contain much of their variance along a single axis. Therefore further improvements might be found by allowing the use of anisoropic approximations (*i.e.* more harmonics/resolution along the x axis than along the y).

Generalising NIR Shape Encodings Chapter 5 covered our work on investigating the generalisation of shape encodings learnt by neural implicit representations (NIRs). Whilst NIRs present an exciting new avenue to improving the efficiency of deep learning piplines for 3D tasks given their superior efficiency compared to conventional representations, we demonstrated the shortcomings of training NIRs on reconstruction tasks alone, in that the shape encodings learnt provide insufficient information for use in semantic tasks. Our experiments show that this shortfall can be addressed by simple join training, and that further, these new more general encodings contain sufficient information to be meaningful for later "unseen" tasks.

As this work was of a more investigative nature, the limitations and future work align very closely. The work we presented in Chapter 6 addresses one of the limitations of this work by removing the need for re-training of the encoder, but other potential follow up works would likely include an investigation confirming the trends in this work apply to other NIR formulations, like DeepSDF [17] and others. Further, the inclusion of more downstream tasks might be of interest, however,

given our understanding of the availability of datasets, tasks, and baselines, it is not clear what these tasks might, cloud, or ought to be.

Contextualising Shape Encodings Finally, we outlined our work on contextualising the shape encodings of NIRs for use in semantic tasks in Chapter 6. We proposed a lightweight contextualising module that reveals hidden semantic information in the encodings learnt by NIRs on reconstruction tasks alone. Combining the compact context encoding generated with the original shape encoding, our method is able to provide substantially superior semantic task performance than the shape encoding alone. Additionally, using the contextualising module, our method allows for training on large unlabelled datasets, before fine-tuning on a labelled dataset for a given semantic tasks. This procedure admits superior reconstruction performance, compared to training on the labelled dataset alone, without compromising semantic task performance, whilst minimising data acquisition costs.

Ultimately, the main limitation of this work is the persistence of the need for semantic labelled data. Future work might investigate whether similar contextualising can be achieved through the use of self supervised tasks [190], such that the context encodings are meaningful in semantic tasks without needing labelled data.

Bibliography

[1] Seppo Linnainmaa. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. PhD thesis, Master's Thesis (in Finnish), Univ. Helsinki, 1970.

[2] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10): 947–954, 1960.

[3] G.W.F. von Leibniz, J.M. Child, and C.I. Gerhardt. *The Early Mathematical Manuscripts of Leibniz: Translated from the Latin Texts Published by Carl Immanuel Gerhardt with Critical and Historical Notes*. Open court publishing Company, 1920. URL `https://books.google.co.uk/books?id=bOIGAAAAYAAJ`.

[4] 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_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[5] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2014.09.003. URL `https://www.sciencedirect.com/science/article/pii/S0893608014002135`.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[8] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[9] 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.

[10] OpenAI. Gpt-4 technical report, 2023.

[11] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[12] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[13] IEEE. Ieee standard for binary floating-point arithmetic. *ANSI/IEEE Std 754-1985*, pages 1–20, 1985. doi: 10.1109/IEEESTD.1985.82928.

[14] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[15] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.

[16] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[17] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[18] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[20] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[21] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[22] Marcelo Gennari do Nascimento, Theo W Costain, and Victor Adrian Prisacariu. Finding non-uniform quantization schemes using multi-task gaussian processes. In *European Conference on Computer Vision*, 2020.

[23] Theo W Costain and Victor Adrian Prisacariu. Approximating continuous convolutions for deep network compression. In *British Machine Vision Conference*, 2022.

[24] Theo W Costain and Victor Adrian Prisacariu. Towards generalising neural implicit representations. In *European Conference on Computer Vision Workshop*, 2022.

[25] Theo W Costain, Kejie Li, and Victor Adrian Prisacariu. Contextualising implicit representations for semantic tasks. *Submitted For Publication*, 2023.

[26] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[27] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[28] 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.

[29] 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, 2016.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[31] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[32] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[33] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[34] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[35] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=r1Ue8Hcxg`.

[36] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2820–2828, 2019.

[37] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[38] Ahmed T Elthakeb, Prannoy Pilligundla, Fatemehsadat Mireshghallah, Amir Yazdanbakhsh, and Hadi Esmaeilzadeh. Releq: A reinforcement learning approach for automatic deep quantization of neural networks. *IEEE micro*, 40(5):37–45, 2020.

[39] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=S1eYHoC5FX`.

[40] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[41] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE transactions on neural networks and learning systems*, 2021.

[42] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1946–1956, 2019.

[43] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural*

Information Processing Systems, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/f33ba15effa5c10e873bf3842afb46a6-Paper.pdf`.

[44] Geoffrey F Miller, Peter M Todd, and Shailesh U Hegde. Designing neural networks using genetic algorithms. In *ICGA*, volume 89, pages 379–384, 1989.

[45] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[46] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary intelligence*, 1:47–62, 2008.

[47] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, 2019.

[48] Pradnya A Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)*, pages 261–265. IEEE, 2016.

[49] Thomas Bartz-Beielstein, Jürgen Branke, Jörn Mehnen, and Olaf Mersmann. Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(3):178–195, 2014.

[50] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018.

[51] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International conference on machine learning*, pages 2902–2911. PMLR, 2017.

[52] Aditya Rawal and Risto Miikkulainen. From nodes to networks: Evolving recurrent neural networks. *arXiv preprint arXiv:1803.04439*, 2018.

[53] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[54] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[55] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[56] Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In *International Conference on Machine Learning*, pages 1655–1664. PMLR, 2017.

[57] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Towards automatically-tuned neural networks. In *Workshop on automatic machine learning*, pages 58–65. PMLR, 2016.

[58] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1946–1956, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330648. URL `https://doi.org/10.1145/3292500.3330648`.

[59] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, June 2022.

[60] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[61] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[62] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, pages 5191–5198, 2020.

[63] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[64] 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, 2019.

[65] Sajid Anwar and Wonyong Sung. Coarse pruning of convolutional neural networks with random masks. arxiv, 2016. URL `https://openreview.net/forum?id=HkvS3Mqxe`.

[66] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[67] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[68] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

[69] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. Thinet: pruning cnn filters for a thinner net. *IEEE transactions on pattern analysis and machine intelligence*, 41(10):2525–2538, 2018.

[70] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[71] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

[72] Elad Hoffer, Itay Hubara, and Daniel Soudry. Fix your classifier: the marginal value of training the last weight layer. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1Dh8Tg0-`.

[73] Federico Pernici, Matteo Bruni, Claudio Baecchi, and Alberto Del Bimbo. Regular polytope networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4373–4387, 2021.

[74] Federico Pernici, Matteo Bruni, Claudio Baecchi, and Alberto Del Bimbo. Fix your features: Stationary and maximally discriminative embeddings using regular polytope (fixed classifier) networks. *arXiv preprint arXiv:1902.10441*, 2019.

[75] Federico Pernici, Matteo Bruni, Claudio Baecchi, and Alberto Del Bimbo. Maximally compact and separated features with regular polytope networks. In *CVPR Workshops*, pages 46–53, 2019.

[76] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

[77] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV*, pages 525–542. Springer, 2016.

[78] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[79] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[80] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018.

[81] Alasdair Paren and Rudra PK Poudel. Training binarized neural networks the easy way. In *BMVC*, page 35, 2022.

[82] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[83] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach 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. *CoRR*, abs/1912.01703, 2019. URL `http://arxiv.org/abs/1912.01703`.

[84] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[85] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[86] Marcelo Gennari do Nascimento, Roger Fawcett, and Victor Adrian Prisacariu. Dsconv: efficient convolution operator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5148–5157, 2019.

[87] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *Advances in neural information processing systems*, 30, 2017.

[88] Guandao Yang, Tianyi Zhang, Polina Kirichenko, Junwen Bai, Andrew Gordon Wilson, and Chris De Sa. Swalp: Stochastic weight averaging in low precision training. In *International Conference on Machine Learning*, pages 7015–7024. PMLR, 2019.

[89] Roberto Rigamonti, Amos Sironi, Vincent Lepetit, and Pascal Fua. Learning separable filters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2754–2761, 2013.

[90] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.

[91] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015.

[92] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[93] Yunpeng Chen, Xiaojie Jin, Bingyi Kang, Jiashi Feng, and Shuicheng Yan. Sharing residual units through collective tensor factorization to improve deep neural networks. In *IJCAI*, pages 635–641, 2018.

[94] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using finetuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[95] Peisong Wang, Qinghao Hu, Zhiwei Fang, Chaoyang Zhao, and Jian Cheng. Deepsearch: A fast image search framework for mobile devices. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(1):1–22, 2017.

[96] Henry Howard-Jenkins, Yiwen Li, and Victor Adrian Prisacariu. Gross: Group-size series decomposition for grouped architecture search. In *European Conference on Computer Vision*, pages 18–33. Springer, 2020.

[97] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[98] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part ii: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1033–1066, 2008.

[99] Nikhil Saldanha, Silvia L Pintea, Jan C van Gemert, and Nergis Tomen. Frequency learning for structured cnn filters with gaussian fractional derivatives. In *BMVC*, 2021.

[100] Julio Zamora, Jesus A. Cruz Vargas, Anthony Rhodes, Lama Nachman, and Narayan Sundararajan. Convolutional filter approximation using fractional calculus. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 383–392, October 2021.

[101] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017.

[102] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017.

[103] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020.

[104] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[105] Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, and Enrico Magli. Learning graph-convolutional representations for point cloud denoising. In *European conference on computer vision*, pages 103–118. Springer, 2020.

[106] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[107] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10296–10305, 2019.

[108] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

[109] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019.

[110] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[111] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[112] Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. Flex-convolution: Million-scale point-cloud learning beyond grid-worlds. In *Asian Conference on Computer Vision*, pages 105–122. Springer, 2018.

[113] Dario Rethage, Johanna Wald, Jurgen Sturm, Nassir Navab, and Federico Tombari. Fully-convolutional point networks for large-scale point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 596–611, 2018.

[114] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018.

[115] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

[116] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European conference on computer vision (ECCV)*, pages 87–102, 2018.

[117] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539, 2018.

[118] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015.

[119] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018.

[120] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019.

[121] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[122] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9785–9795, 2019.

[123] Victor Adrian Prisacariu, Olaf Kähler, Stuart Golodetz, Michael Sapienza, Tommaso Cavallari, Philip HS Torr, and David W Murray. Infinitam v3: A framework for large-scale 3d reconstruction with loop closure. *arXiv preprint arXiv:1708.00783*, 2017.

[124] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[125] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[126] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[127] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[128] 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.

[129] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022.

[130] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

[131] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding, 2023.

[132] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021.

[133] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 9224–9232, 2018.

[134] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[135] 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 *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[136] 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.

[137] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

[138] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[139] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings*

of the IEEE/CVF International Conference on Computer Vision, pages 5752–5761, 2021.

[140] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[141] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

[142] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12949–12958, 2021.

[143] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022.

[144] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

[145] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021.

[146] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[147] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. *arXiv preprint arXiv:2103.15875*, 2021.

[148] Suhani Vora*, Noha Radwan*, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi S. M. Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *Transactions on Machine Learning Research*, 2022. https://openreview.net/forum?id=ggPhsYCsm9.

[149] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021.

[150] Qianyi Wu, Xian Liu, Yuedong Chen, Kejie Li, Chuanxia Zheng, Jianfei Cai, and Jianmin Zheng. Object-compositional neural implicit surfaces. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pages 197–213. Springer, 2022.

[151] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022.

[152] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 34:27171–27183, 2021.

[153] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.

[154] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[155] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020.

[156] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[157] Jianglong Ye, Yuntao Chen, Naiyan Wang, and Xiaolong Wang. Gifs: Neural implicit function for general shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12829–12839, 2022.

[158] Benoît Guillard, Federico Stella, and Pascal Fua. MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, volume 13663, pages 576–592. Springer Nature Switzerland, Cham, 2022. ISBN 978-3-031-20061-8 978-3-031-20062-5. doi: 10.1007/978-3-031-20062-5_33. URL `https://link.springer.com/10.1007/978-3-031-20062-5_33`. Series Title: Lecture Notes in Computer Science.

[159] Jiapeng Tang, Jiabao Lei, Dan Xu, Feiying Ma, Kui Jia, and Lei Zhang. Sa-convonet: Sign-agnostic optimization of convolutional occupancy networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6504–6513, 2021.

[160] Bing Wang, Zhengdi Yu, Bo Yang, Jie Qin, Toby Breckon, Ling Shao, Niki Trigoni, and Andrew Markham. Rangeudf: Semantic surface reconstruction from 3d point clouds. *arXiv preprint arXiv:2204.09138*, 2022.

[161] Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Learning consistency-aware unsigned distance functions progressively from raw point clouds. 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=KqI-bX-TfT`.

[162] Xianghui Yang, Guosheng Lin, Zhenghao Chen, and Luping Zhou. Neural vector fields: Implicit representation by explicit learning. *arXiv preprint arXiv:2303.04341*, 2023.

[163] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[164] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, Cham, August 2020. Springer International Publishing.

[165] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020.

[166] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. *arXiv preprint arXiv:2003.10983*, 2020.

[167] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.

[168] Amit Pal Singh Kohli, Vincent Sitzmann, and Gordon Wetzstein. Semantic implicit neural scene representations with semi-supervised training. In *2020 International Conference on 3D Vision (3DV)*, pages 423–433. IEEE, 2020.

[169] Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi di Stefano. Deep learning on implicit neural representations of shapes. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=OoOIW-3uadi`.

[170] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2, 11 2017. doi: 10.23915/distill.00007.

[171] Roman Ilin, Thomas Watson, and Robert Kozma. Abstraction hierarchy in deep learning neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 768–774, 2017. doi: 10.1109/IJCNN.2017.7965929.

[172] Sergei Bernstein. D´emonstration du th´eor‘eme de weierstrass fond´ee sur le calcul des probabilit´es. *Comm. Kharkov Math. Soc*, 12, 1912.

[173] Pafnutiĭ L'vovich Chebyshev. *Théorie des mécanismes connus sous le nom de parallélogrammes*. Imprimerie de l'Académie impériale des sciences, 1853.

[174] Annette Ekin. Ai can help us fight climate change. but it has an energy problem, too. *Horizon*, 2019. URL `https://ec.europa.eu/research-and-innovation/en/horizon-magazine/ai-can-help-us-fight-climate-change-it-has-energy-problem-too`.

[175] Mattias P Heinrich, Max Blendowski, and Ozan Oktay. Ternarynet: faster deep model inference without gpus for medical 3d segmentation using sparse and binary convolutions. *International journal of computer assisted radiology and surgery*, 13:1311–1320, 2018.

[176] Magdalini Paschali, Stefano Gasperini, Abhijit Guha Roy, Michael Y-S Fang, and Nassir Navab. 3dq: Compact quantized neural networks for volumetric whole brain segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part III 22*, pages 438–446. Springer, 2019.

[177] Ximeng Sun, Rameswar Panda, Chun-Fu (Richard) Chen, Aude Oliva, Rogerio Feris, and Kate Saenko. Dynamic network quantization for efficient video inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7375–7385, October 2021.

[178] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

[179] Yi Zhu, Xinyu Li, Chunhui Liu, Mohammadreza Zolfaghari, Yuanjun Xiong, Chongruo Wu, Zhi Zhang, Joseph Tighe, R Manmatha, and Mu Li. A comprehensive study of deep video action recognition. *arXiv preprint arXiv:2012.06567*, 2020.

[180] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

[181] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the*

IEEE/CVF international conference on computer vision, pages 5552–5561, 2019.

[182] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4597–4605, 2015.

[183] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[184] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 305–321, 2018.

[185] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[186] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*, 2015.

[187] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[188] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[189] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 fourth international conference on 3D vision (3DV)*, pages 92–101. Ieee, 2016.

[190] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8160–8171, 2019.