From Matrix Factorisation to Signal Propagation in Deep Learning: Algorithms and Guarantees

Michael Murray

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2021

Acknowledgements

First and foremost I would like to thank my supervisor Jared Tanner for taking me under his wing and guiding me through this first stage of an academic career. It's been a real pleasure and I couldn't have asked for a more insightful, generous, kind and thoughtful mentor. In addition, I would like to express my gratitude to the postdocs I have had the privilege to learn from and work with along the way, namely Hemant Tyagi, Jeremias Sulam and Vinayak Abrol. I am also very grateful to The Alan Turing Institute and the Ana Leaf Foundation for the studentship that has enabled me to undertake these studies.

To all my friends and colleagues at both the Alan Turing Institute and the Oxford Mathematics Institute I would like to say a massive thanks for making this experience such a rich, fun and happy one. Last but far from least I need to say a huge thanks to my family and newly married wife Sophie for all their love, support, encouragement and for putting up with me regressing back to being a student!

Abstract

Many problems in data science amount to computing an appropriate representation of the data for the task at hand: two examples related to this thesis are 1) reducing memory, sensing or transmission costs by computing a sparse representation of the data and 2) performing classification by transforming the data into a representation in which the members of different classes are linearly separable. The unifying theme of this thesis is the design and analysis of algorithms which are guaranteed, at least with high probability, to compute useful representations. We study this in two contexts: first a matrix factorisation problem inspired by compressed sensing, and second signal propagation in deep learning. In the first context we seek to recover sparse representations of the data from a set of linear measurements without access to the underlying sensing mechanism. In the second context we analyse both the forward and backward pass of deep neural networks, investigating how the choice of activation function impacts the sequences of representations and error vectors that these generate respectively. In Chapter 1 we provide the necessary background to understand the contributions of this thesis, which we now summarise.

In Chapter 2 we study the problem of recovering the factors $\mathbf{A}$ and $\mathbf{X}$ from $\mathbf{Y} := \mathbf{AX}$, where $\mathbf{A}$ is an $m \times n$ sparse, binary matrix with a fixed number d nonzeros per column, and $\mathbf{X}$ is an $n \times N$ sparse real matrix whose columns have at most k nonzeros and are dissociated. Matrices defined by this factorisation can be expressed as a sum of n rank one sparse matrices, whose nonzero entries, under the appropriate permutation, form dense rectangles or blocks with a constant value in each column. We therefore refer to them as Permuted Striped Block (PSB) matrices. Motivated by multimeasurement vector combinatorial compressed sensing (MMV-CCS), we define the PSB model as a particular distribution over this set of matrices. For matrices drawn from the PSB model, we provide a computationally efficient factorisation algorithm which recovers the generating factors with high probability in n from a near optimal number $N = \Omega\left(\frac{n}{k}\log^2(n)\right)$ of vectors, where k, m and n scale proportionally. We demonstrate experimentally the efficacy of this algorithm in practice and highlight potential applications: in particular, instead of requiring full access to the encoder matrix, our results show that in the MMV-CCS setting sparse recovery is possible using only knowledge of the distribution from which the encoder matrix is sampled.

In Chapter 3 we continue our study of sparse representations, but consider them instead in the context of deep convolutional neural networks (DCNNs). In particular,

we investigate the impact of ReLU and related sparsifying activation functions on signal propagation by studying the representations and activation patterns computed by the forward pass at each layer. To this end we consider a variant of the approach proposed by Papyan et al [96], which interprets the forward pass of a DCNN as solving a sequence of sparse coding problems - we hence refer to this approach as Deep Convolutional Sparse Coding (DCSC). The benefit of this approach is that by assuming a particular data model based on the notion of reversible networks, and by omitting the learning of parameters, one can more transparently analyze the role of the activation function and the efficacy of the forward pass of a DCNN in recovering activation pathways. In the prior work [96], the authors proved that representations with an activation density proportional to the ambient dimension of the data are recoverable. We extend these uniform guarantees and prove with high probability that representations with a far greater density of activations per layer are recoverable.

Finally, in Chapter 4 we continue our investigation on the impact of the choice of activation function on signal propagation in deep, if not convolutional, neural networks. However, diverging markedly from Chapter 3 we forgo any assumptions on the data and instead consider random networks, which are of particular relevance for studying the properties of networks at initialisation. In this chapter we prove that certain problems at initialisation, which stop or hinder subsequent training, can be avoided by ensuring that the activation function deployed has a sufficiently large linear region around the origin. We further demonstrate empirically that using such activation functions leads to tangible benefits in practice, both in terms of test and training accuracy as well as training time. In addition, these results also allow us to successfully train networks in new hyperparameter regimes, with much larger bias variances than have been used before.

Contents

Chapter 1

Introduction

Transforming data into a different representation can be beneficial for a number of reasons; notably 1) to alleviate computational overheads involved in storing, transmitting and processing and 2) in order to highlight or remove different factors of variation, thereby separating the signal from the noise. Many data processing tasks can be performed by computing a suitable representation of the data: for example, the frequency representation of a radio signal can be used to perform filtering and denoising in communication systems while a linearly separable representation of objects in image data facilitates classification in computer vision. Transforms designed by hand, such as the Fast Fourier [29] or Wavelet [84] transforms, compute representations with well understood properties, are interpretable and have strong guarantees. However, this approach to algorithm design typically requires significant domain knowledge, analysis and effort.

A data-driven or machine learning approach to algorithm design reduces the burden involved with designing a data transform by hand by attempting to automatically select a suitable transform for the domain and task. One way of implementing this approach is via a parametric model, whose parameters are chosen in order to minimise a cost function, measuring some notion of performance, defined over a set of example data points called a training data set. The process of choosing appropriate parameters for the task at hand is colloquially referred to as learning or training. Matrix factorisation forms the basis of one such parametric model, in which each example in the training set is expressed as a linear combination of common vectors which act as the parameters of the model. Two specific examples of this are principal component analysis (PCA) [71], in which the common vectors are the principal components or directions of greatest variation in the training data, and dictionary learning, in which the common vectors correspond to atoms of an overcomplete dictionary used to obtain a sparse representation of the training data. These common

vectors are therefore customised to the domain in question, but can be computed, or learnt, generically using for example singular value decomposition (SVD) [53] in the case of PCA, or the K-SVD algorithm [2] in the context of dictionary learning [129]. Once acquired, these common vectors can then be used to compute meaningful or practically useful representations of new data points: for example, by deploying them as a linear transformation in combination with a thresholding operation then a low dimensional or sparse representation of the data can be acquired. Compared with methods designed by hand, which are typically domain specific, this approach is general and can be used to derive methods to compute desirable representations for a wide range of domains and tasks. On the other hand, it can be harder to provide guarantees concerning both the learnability of such transforms as well their ability to generalise to data points outside of the training set.

Increasingly, for many applications including image recognition [75], video captioning [62], natural language processing [19, 121] and audio processing [104], state of the art results are achieved using deep neural networks (DNNs) [54]. However, our ability to successfully deploy these models to solve challenging problems in general belies our understanding of them. Compared with the matrix factorisation based methods discussed previously, the training dynamics of neural networks and the representations that they generate are poorly understood. This lack of understanding can result in problems at initialisation as well as during training and deployment. First, at initialisation certain architecture and hyperparameter choices can scupper or hinder subsequent training. Second, given the richness of approximation of a given DNN architecture, the problem of overfitting [54, Chapter 5] may arise during training. Here, instead of learning representations which capture some common and task-relevant structure of the data, the learning algorithm focuses on the idiosyncrasies of the training data. To avoid this in practice adjustments to the hyperparameters or network architecture is required, which increases the time and cost to train these models. Third, in the context of deployment, concerns around the reliability of trained DNNs when it comes to processing new data points, i.e. their capacity to generalise, cannot be answered in confidence by assessing their empirical performance on a test set alone; this is highlighted by issues such as data set bias [120] and vulnerability to adversarial examples [126]. In many applications one might argue that this is not a cause for concern, in advertising for instance the only real risk is a lost sales opportunity. The same cannot be said for self driving cars, automated parole systems, medical screening tools and financial trading software. In order to avoid overfitting

and ensure generalisation we seek guarantees that the forward pass will compute representations with explanatory power, capturing task relevant structure present in all possible inputs rather than just those present in the training and test data sets.

To summarise, data driven approaches to algorithm design have many advantages and can produce state of the art results for many information processing tasks. However, training a model can be challenging and computationally costly, and the data transform produced often lacks guarantees when it comes to processing new data points, hindering deployment in higher risk applications. Here we adopt the hypothesis that these problems can be better understood and mitigated by analysing the representations generated by these learnt transforms. The unifying theme of this thesis then is the design and analysis of data-driven algorithms with guarantees that they compute representations which have explanatory power, or some other practical benefit. We consider this theme in the two contexts already discussed: firstly matrix factorisation, where we interpret one of the factor matrices as a target representation of the product matrix, and secondly the forward pass of a deep neural network, in which we analyse the impact of the choice of activation function on the evolution, with depth, of the representations of the data.

In order to study and potentially mitigate problems concerning overfitting and generalisation we derive certain recovery guarantees; this type of guarantee assumes that the observed data is generated by applying a particular function, referred to as an encoder, which can be either random or deterministic, to some vector. We interpret the input to the encoder as a representation of the observed data, which by construction has explanatory power or some other practical benefit. We then analyse the ability of an algorithm or inference technique, referred to as a decoder, to recover at least certain aspects of this input vector. In Chapters 2 and 3 we provide recovery guarantees in the situation where the encoder is a linear transform, and the decoder is a matrix factorisation algorithm and the forward pass of a neural network respectively. However, while in Chapter 2 the encoder is hidden from the decoder, in Chapter 3 we assume that the decoder has access to the encoder. In Chapter 4 we start by identifying certain failure modes at initialisation, caused by faulty propagation of signals in both the forward and backward pass. We then demonstrate how these failure modes can be avoided through careful selection and adaption of the activation function deployed. In the rest of this chapter we proceed to outline and motivate in more detail the problems studied as well as the contributions of this thesis.

1.1 A matrix factorisation problem inspired by compressed sensing

In many data science contexts, data is represented as a matrix and is often factorised into the product of two or more structured matrices so as to reveal important information. Perhaps the most famous of these factorizations is principle component analysis (PCA) [71], in which the unitary factors represent dominant correlations within the data. Dictionary learning [129] is another prominent matrix factorisation, in which the data matrix is viewed to lie, at least approximately, on a union of low dimensional subspaces. These subspaces are represented as the product of an overcomplete matrix, known as a dictionary, and a sparse matrix. More generally a wide variety of matrix factorizations have been studied to solve a broad range of problems, for example missing data in recommender systems [92], nonnegative matrix factorisation [80] and automatic separation of outliers from a low rank model via sparse PCA [23].

In Chapter 2 we study a new matrix factorisation problem inspired by multimeasurement vector combinatorial compressed sensing (MMV-CSS). Compressed sensing (CS) [20, 21, 22, 24, 25, 36], in its simplest setting, studies the design of algorithms and sensing matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ so as to enable the recovery of a sparse vector $\mathbf{x} \in \mathbb{R}^n$ from $m < n$ linear measurements $\mathbf{y} := \mathbf{A}\mathbf{x}$. A key application of CS is the recovery and reconstruction of sparse signals using a number of measurements well below that suggested by the Nyquist-Shannon sampling theorem: in particular, instead of sampling a signal at a high rate and then performing compression which can be wasteful, one can directly sample or sense the data in a compressed form, reducing the number of measurements required for reconstruction. Multimeasurement vector compressed sensing [27, 31, 90] studies the compressed sensing problem in the context of a set of $N > 1$ measurement vectors and can be expressed as the recovery of a sparse matrix $\mathbf{X}^{n \times N}$ from a measurement matrix $\mathbf{Y} := \mathbf{A}\mathbf{X}$. Typically $\mathbf{X}$ is structured such that sparse recovery of one column aids the recovery of others, for example, in the joint sparse setting the columns of $\mathbf{X}$ share a common support. Finally, combinatorial compressed sensing (CCS) [14] analyses the compressed sensing problem in the setting where the encoder matrix is sparse and binary. Not only are such matrices the natural sensing mechanism in some applications, but they also have computational benefits in terms of the cost to generate, store and apply them.

Our goal in Chapter 2 is to derive an algorithm with low sample and computational complexity, which can recover the matrix factors $\mathbf{A}$ and $\mathbf{X}$ up to column and row permutation respectively from $\mathbf{Y} := \mathbf{A}\mathbf{X}$. Here $\mathbf{A}$, which is referred to as the encoder

matrix, is an $m \times n$ sparse, binary matrix with a fixed number d nonzeros per column and $\mathbf{X}$, which is referred to as the sparse coding matrix, is an $n \times N$ sparse real matrix whose columns have at most k nonzeros and are dissociated. As our goal here is to use the different measurement vectors of $\mathbf{Y}$ to learn the encoder matrix $\mathbf{A}$, then unlike the joint sparse setting we do not impose a sparsity structure on the rows of $\mathbf{X}$. However, to facilitate efficient factorisation and avoid combinatorial searches, we impose the condition that the nonzero values in each column of $\mathbf{X}$ are dissociated [117]. This condition, borrowed from the field of additive combinatorics, ensures that the sum over any two different subsets of the nonzero values of a given column of $\mathbf{X}$ are distinct.

Matrices defined by the factorisation $\mathbf{Y} := \mathbf{AX}$ as defined above can be written as a sum of n rank one sparse matrices, whose nonzero entries, under the appropriate permutation, form dense rectangles or blocks with a constant value in each column. We therefore refer to them as Permuted Striped Block (PSB) matrices and the associated factorisation task as PSB factorisation. Inspired by certain random constructions of encoder matrices in the CCS literature, we study PSB factorisation under the assumption that A and X are particular random matrices, see Definition 2.1.3 in Section 2.1.2, whose product defines a distribution over PSB matrices. We refer to this distribution over the set of PSB matrices as the PSB model. In order to solve the PSB factorisation problem, in Chapter 2 we present the Decoder-Expander Based Factorisation (D-EBF) algorithm. For matrices distributed according to the PSB model, we prove that D-EBF recovers the encoder and sparse coding matrices up to permutation with high probability from as few as $N = \Omega(\frac{n}{k} \log^2(n))$ measurement vectors, which is optimal up to logarithmic factors. By imposing an additional and unique ordering on the columns of the encoder, then this result demonstrates that multimeasurement vector sparse reconstruction is possible without access to the relevant encoder matrix. Finally, we demonstrate the efficacy of this algorithm in practice.

1.2 Signal propagation in deep neural networks

We start this section by providing a brief primer on neural networks, for a more in depth overview we refer the reader to [54, 109]. Artificial neural networks where originally conceived as a mathematical model for biological information processing systems [17]. Mimicking networks of biological neurons, an artificial neural network is a directed network of computational nodes or neurons. Each neuron consists of

a weight vector $\mathbf{w} \in \mathbb{R}^n$, a bias $b \in \mathbb{R}$ and an activation function $\phi : \mathbb{R} \to \mathbb{R}$. Letting $\mathbf{x} \in \mathbb{R}^n$ denote a vector containing the inputs to a neuron, then the output of the neuron is computed as $\phi(\mathbf{w}^T\mathbf{x} + b)$. As a result, each neuron can be thought of as computing a different feature of the data, as determined by its weight and bias parameters. In this thesis we only consider feedforward neural networks, in which the neurons are typically organised into a sequence of layers, with interlayer connections between adjacent layers and no intralayer connections. Such networks have a designated input and output layer, and any layer between these two is referred to as a hidden layer. Networks with one hidden layer are called single layer networks, while networks with more than one hidden layer are called multilayer or deep neural networks (DNNs). The forward pass of a neural network is the function mapping from the input to the output layer. In the case of feedforward neural networks this function can be expressed as a composition of affine transforms and elementwise activation functions. Denoting the forward pass of an L layer feedforward neural network as $f^{(L)} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$, then for an input $\mathbf{x} \in \mathbb{R}^{n_0}$

$$f^{(L)}(\mathbf{x}) := (\phi_L \circ \mathbf{A}_L \circ \phi_{L-1} \circ \mathbf{A}_{L-1} ... \circ \phi_1 \circ A_1)(\mathbf{x}). \tag{1.1}$$

Here $\phi_l : \mathbb{R} \to \mathbb{R}$ is the elementwise activation function deployed at the lth layer, and $A_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ denotes an affine transform

$$A_l(\mathbf{x}^{(l-1)}) := \mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}, \tag{1.2}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, $\mathbf{b} \in \mathbb{R}^{n_l}$ and n_l denotes the dimension or number of neurons in the lth layer of the network. The representation, or activation as per the analogy with a biological brain, of the input $\mathbf{x}$ at the lth layer is denoted $\mathbf{x}^{(l)} := f^{(l)}(\mathbf{x})$, can also be expressed in terms of a recurrence equation $\mathbf{x}^{(l)} = \phi_l(\mathbf{z}^{(l)})$, where $\mathbf{z}^{(l)} := \mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}$ is referred to as the preactivation at the lth layer. The weight matrix $\mathbf{W}^{(l)}$ describes the connectivity between the $l-1$th and lth layers of the network. Unless stated otherwise, the weight matrices of a DNN are assumed to be dense. This implies that each node in a given layer is connected to all the nodes in the adjacent layers. Enforcing different connectivity patterns, or alternatively structure on the weight matrices, gives rise to different subcategories of DNN: in particular in Chapter 3 we consider deep convolutional neural networks (DCNNs), in which the weights matrices are convolutional [97]. Ever since the arrival of AlexNet [76] in 2012, DCCNs have been the state of the art for many problems in computer vision. They have also achieved excellent results in a host of other applications, including Natural Language Processing [73] and Speech Recognition [127].

The parameters of a fixed neural network architecture can be configured to implement a wide variety of different functions. This is the first ingredient behind their success, allowing the same or similar neural network architectures to be applied to solve a wide variety of different problems. The universal approximation theorem [32, 68, 102] states that a single layer feedforward network with a finite number of neurons can approximate any continuous function on a compact subset of $\mathbb{R}^n$, with only mild constraints on the activation functions deployed. Naturally DNNs also satisfy the universal approximation theorem, however, they are thought to be more expressive than single layer networks as their structure allows for a hierarchy of features. Intuitively, nodes deep in the network are able to efficiently compute high level features by combining a shared set of lower level features, computed by nodes in layers closer to the input. A number of theoretical works support this intuitive argument, at least for certain classes of function, demonstrating that deep networks are exponentially more efficient than shallow ones in terms of the number of nodes required to approximate a given function [43, 89, 118].

The second ingredient behind the success of neural networks is that their modular, compositional structure enables easy computation of the gradients of the network output with respect to the model parameters. The computation of these gradients can efficiently be performed using the backpropagation algorithm [107]. Backpropagation, also referred to as the backward pass of the network, generates a sequence of vectors $(\delta^{(l)}(\mathbf{x}))_{l=1}^{L}$ at each layer with respect to an input $\mathbf{x} \in \mathbb{R}^N$, which measure the error associated with minimising the loss $\mathcal{L}(\theta, x)$. Here θ denotes the trainable network parameters, $\theta := \bigcup_{l=1}^{L}\{\mathbf{W}^{(l)}\} \cup \{\mathbf{b}^{(l)}\}$. For a suitably smooth loss function $\mathcal{L}(\theta, x)$, the backpropagation error vector $\delta^{(l)}(\mathbf{x})$, and the gradient of $\mathcal{L}(\theta, x)$ with respect to biases $b_j^{(l)}$ and weights $w_{j,i}^{(l)}$, at each layer $l \in [L]$ can be computed using the following recurrence equations,

$$\begin{aligned}
\delta^{(L)}(\mathbf{x}) &= \mathbf{D}^{(L)}(\mathbf{x})\nabla_{\mathbf{h}^{(L)}}\mathcal{L}(\theta, \mathbf{x}), \\
\delta^{(l)}(\mathbf{x}) &= \left(\mathbf{D}^{(l)}(\mathbf{x})\mathbf{W}^{(l+1)}\right)^T \delta^{(l+1)}(\mathbf{x}), \\
\frac{\partial \mathcal{L}(\theta, \mathbf{x})}{\partial b_j^{(l)}} &= \delta_j^{(l)}(\mathbf{x}), \\
\frac{\partial \mathcal{L}(\theta, \mathbf{x})}{\partial w_{j,i}^{(l)}} &= h_i^{(l-1)}(\mathbf{x})\delta_j^{(l)}(\mathbf{x}).
\end{aligned} \tag{1.3}$$

Here, for each $l \in [L]$, $\mathbf{D}^{(l)}(\mathbf{x})$ is a diagonal matrix with $D_{ii}^{(l)}(\mathbf{x}) := \phi'(z_i^{(l)}(\mathbf{x}))$ for all $i \in [N_l]$. Consequently, the weights and biases of the network can be iteratively updated via gradient descent type algorithms, or some variant thereof, in order that

the forward pass improves its performance, as measured typically by averaging $\mathcal{L}(\theta, \mathbf{x})$ over batches of the training data, at executing the task in hand. We emphasise that in practice, instead of updating the parameters with respect to a single data point, gradients are calculated and averaged over batches of the training data.

While the advent of deep neural networks dates as far back as the 1960s, it took many years for deep networks to realise their potential due to the challenges involved with training them. In particular the problems of vanishing and exploding gradients [64, 74] and the related problem of model degeneracy [41, 94, 108] were identified as key factors driving poor training outcomes. To address this challenge a number of practical architectural and algorithmic solutions have been proposed. On the architectural side innovations in this regard include the introduction of new activation functions, notably the now ubiquitous ReLU [50], as well as certain wiring techniques, such as LSTM cells [66] for recurrent neural networks, and residual connections [60] and other variants, e.g., highway networks [113], for feedforward architectures. On the algorithmic side, new initialisation schemes [52, 61] for network parameters and normalisation of hidden representations [8, 69] have proven effective in practice. Using such techniques it is now possible to train truly massive deep learning based models, such as the GPT3 language model [19]. However, the computational costs involved in training these massive models is becoming increasingly prohibitive. If we wish to address this issue and decouple performance from compute power, then a better understanding of the interplay between architecture, training algorithm and parameter initialisation is required.

Even if the network achieves a low error on the training data set there are still no guarantees that the network will generalise well to new data points. In practice the problem of overfitting [54, Chapter 5] can be alleviated by using a test data set, which is distinct and separate to the training set, to assess the ability of the network to generalise during training. If and when the training error and test error diverge then a number of strategies can be adopted to draw them back in line with one another. Examples of such strategies include adjusting the network hyperparameters, adding regularisation terms to the cost function and making changes to the network architecture, for instance reducing the width and depth of the network or adding dropout layers. However, these approaches are heuristic in nature and labour intensive, which increases the time and cost to train useful deep learning models. Additionally, even strong empirical performance on both training and test data sets may not provide sufficient confidence in a network to use it in high risk applications. In particular, data set bias [120], in which the collected train and test data sets do not accurately

represent the population, and vulnerability to adversarial examples [126], which are small perturbations applied to the input data in order to trick the network into making mistakes, can cause problems in deployment. To have confidence in deep learning based methods and deploy them in higher risk contexts, we need guarantees that the forward pass algorithm computes meaningful representations of the population data.

To address these issues we study signal propagation in DNNs, which refers to the study of the sequences of representations $\{\mathbf{x}^{(l)}\}_{l=1}^{L}$ and error vectors $\{\delta^{(l)}\}_{l=1}^{L}$ generated by the forward and backward pass of the network respectively. Analysing these sequences and their statistics will prove fruitful in deriving guarantees that the forward pass generates representations with explanatory power, as illustrated in Chapter 3, as well as in identifying and avoiding problems at initialisation, which is the subject of Chapter 4. In both cases we focus on the role and properties of the activation function in achieving these goals.

1.2.1 Sparse recovery guarantees for the forward pass

In Chapter 3 we turn our attention to analysing signal propagation in the forward pass of Deep Convolutional Neural Networks (DCNNs), seeking to better understand the role of the activation function in computing representations with explanatory power. To this end we build on the work of Papyan et al [96], who, inspired by the connections between convolutional weight matrices used in deep learning and the dictionaries used in convolutional sparse coding [95], as well as the fact that ReLU activation functions are sparsifying, interpreted the forward pass of a DCNN as solving a sequence of convolutional sparse coding problems. This interpretation enables the analysis of the sequence of representations $\{\mathbf{x}^{(l)}\}_{l=1}^{L}$ using tools and ideas from compressed sensing. To this end, we introduce and study the Deep Convolutional Sparse Coding (DCSC) model, defined in Definition 3.2.1. Similar to Chapter 2, this model assumes that the data is generated, at least approximately, by a matrix product between a dictionary, referred to as the global dictionary, and a sparse latent representation. Of key importance is the factorised form of this global dictionary and the sparse intermediary representations generated at different levels or layers of this factorisation. This encoder can therefore be viewed as a linear network, with the activation pathway of a data point being the set of neurons at each layer which fire, i.e., are nonzero, as the signal propagates from the latent space to the observed data space. The activation pathway of a data point therefore indicates the key features present in the data and hence has strong explanatory power. In the DCSC model the forward pass of a DCNN is interpreted as the decoder associated with this linear

encoder. The learning of the weights of the decoder is omitted and the parameters of the encoder and decoder are shared: this allows us to more transparently analyse the role of the activation function in enabling the forward pass to recover the activation pathway of a data point.

Papyan et al [96] conducted an analysis of a similar model, demonstrating its connection to DCNNs and proved conditions under which the forward pass is guaranteed to recover activation pathways. A technical innovation of their work highlights that one can measure the efficacy of a sparsifying activation function through a new, local measure of sparsity particular to the convolutional structure present, referred to as stripe-sparsity. Using this measure the authors proved that representations with an activation density proportional to the ambient dimension of the data are recoverable. However, the upper bounds derived in [96] on the stripe-sparsity of a recoverable activation at a given layer depend on the inverse of the mutual coherence of the weight matrix at that layer, which is typically quite small. This limits the applicability of these results as only data points with a very sparse activation near the input of the encoder are recoverable by the forward pass decoder. In Chapter 3 we extend these uniform guarantees to the modified DCSC model and prove that activation pathways with a greater density of activations per layer are recoverable with high probability. To prove this result we leverage techniques based on one step thresholding developed by Schnass and Vandergheynst [110].

1.2.2 Designing activations for a better initialisation

Tools and principles have emerged in recent years to help understand the impact of different parameter initialisation schemes on training [58, 59, 100, 101, 103, 111]. In particular, [59, 103, 111] analysed signal propagation in the forward pass of a neural network under a wide layer width approximation. These works are based on the observation that the collection of outputs of a single hidden layer neural network converges in distribution to a centred Gaussian process as the width of the hidden layer goes to infinity [91]. This result was also recently extended to the output at each layer of a multilayer network [33, 79]. The convenience of using this model as an approximation for studying wide but finite width networks lies in the fact that the behaviour of a centred Gaussian process is fully described by its covariance matrix. Furthermore, the entries of the covariance matrices at different layers are related to one another via recurrence relationships in the form of integral equations [79, 103]. These integral equations depend on the input covariance at the previous layer, the activation function deployed and the probability distribution of the weights and

biases. The entries of these covariance matrices capture key statistics and properties of the activations of a data point at each layer: in particular, the expected length of an input vector as well as the expected inner product and correlation between pairs of inputs. Analysing the evolution with depth of these pairwise input correlations highlights the importance of choosing $(\phi, \sigma_w^2, \sigma_b^2)$ appropriately in order to avoid an overly sensitive or insensitive network at initialisation.

Another issue that can be encountered at initialisation, and which is encountered more broadly throughout training, is that of vanishing and exploding gradients [65], which can lead to stalled or unstable training respectively. One particular manifestation of vanishing and exploding gradients is model degeneracy [100, 101, 108]. Considering Equation (1.3), model degeneracy arises during the backward pass when the error vectors $\delta^{(l)}$ are projected onto subspaces of decreasing dimension as the depth l goes from L to 1. Model degeneracy can also be viewed as a poorly conditioned input-output Jacobian, with the mass of the distribution of its spectrum becoming increasingly concentrated with depth. As a result, in layers close to the input the directions in which the weights and biases can be updated becomes limited, harming the expressivity of the network. To avoid this Saxe et al [108] proposed that the input-output Jacobian should act as a near isometry, with each singular value being close to one. This property is referred to as *dynamical isometry*. Using tools from free probability and random matrix theory Pennington et al [100, 101] computed the limiting distribution of singular values of the input-output Jacobian and analysed the impact of various initialisation schemes and choices of activation function on it. An interesting finding from [100, 101] is that asymptotically with depth Sigmoid activated networks with an orthogonal initialisation can achieve dynamical isometry, whereas Gaussian initialised ReLU networks cannot.

Despite the contributions of these prior works, many questions concerning the role of the activation function and its interplay with the weight and bias hyperparameters remain unanswered. In particular, the advantages and drawbacks of various properties of the activation function, such as smooth vs nonsmooth, antisymmetric vs symmetric vs nonsymmetric or bounded vs unbounded, in regard to the aforementioned problems at initialisation are not clear. The guiding question in Chapter 4 is therefore as follows: what properties of the activation function are sufficient and or necessary for healthy signal propagation in DNNs at initialisation? To this end we prove that these problems at initialisation, which can stop or hinder subsequent training, can be avoided by ensuring that the activation function deployed has a sufficiently large linear region around the origin. We further demonstrate empirically that using such

activation functions leads to tangible benefits in practice, both in terms of test and training accuracy as well as training time. In addition, these results also allow us to successfully train networks in new hyperparameter regimes with much larger bias variances than has been possible before. Finally, our results suggest a new training protocol, in which a scaled activation function is deployed with the size of its linear region being reduced slowly over the first few epochs. We provide some preliminary experimental results investigating this direction.

1.3 General points on notation

Before proceeding we review certain conventions of notation typically adopted throughout this thesis. Bold upper case characters will typically be used to denote matrices, such as $\mathbf{A}$, $\mathbf{X}$, $\mathbf{Y}$. Column vectors will be denoted using a bold lower case letter, for example $\mathbf{a}$, $\mathbf{x}$, $\mathbf{y}$, and row vectors as $\tilde{\mathbf{a}}$, $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}$. Correspondingly, in terms of the columns, rows and entries of a matrix $\mathbf{A}$, the ith column is denoted $\mathbf{a}_i$, the jth row as $\tilde{\mathbf{a}}_j$, and the entry in the jth row of the ith column as $a_{j,i}$. Non-bold upper case letters will typically be used to refer to random variables. If A is a random matrix then the ith column is denoted A_i, the jth row as $\tilde{A}_j$, and the entry in the jth row of the ith column as $A_{j,i}$. Chapter specific notation is introduced and defined in situ.

Chapter 2

Matrix factorisation under the PSB model

In this chapter we study the matrix factorisation problem inspired by multimeasurement vector combinatorial compressed sensing, described in Section 1.1. In Section 2.1 we define in detail the problem studied as well as providing the necessary background to understand this work and its context in the wider literature. In Section 2.2 we present and analyse the Decoder-Expander Based Factorisation (D-EBF) algorithm, we then prove in Section 2.3 that this algorithm recovers the factor matrices up to permutation with high probability and with near optimal sample complexity. Finally, in Section 2.4 we demonstrate the efficacy of D-EBF in practice.

2.1 Problem definition, motivation and related work

This section is structured as follows: in Section 2.1.1 we provide the relevant background information on combinatorial compressed sensing and expander graphs, in Section 2.1.2 we define and motivate the PSB model, in Section 2.1.3 we summarise our contributions and highlight potential applications, finally, in Section 2.1.4 we discuss related work.

2.1.1 Combinatorial compressed sensing and expander graphs

Compressed sensing (CS) [20, 21, 22, 24, 25, 36] in its simplest form studies the problem of reconstructing a sparse, finite dimensional signal from a small number of linear measurements. A key result from this field of study is that a sparse vector $\mathbf{x} \in \mathbb{R}^n$, which is k sparse meaning that it has at most $k < n$ nonzeros, can be recovered via efficient decoding algorithms from $\mathbf{y} := \mathbf{A}\mathbf{x}$ as long as $m < n$ is sufficiently large relative to k, and the decoder has access to the encoder matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. We refer

the reader to [42, 47] for an overview and survey of the topic. Many theorems guaranteeing the recovery of $\mathbf{x}$ rely on the encoder matrix satisfying one or more properties, in particular the nullspace property [28, 35], a condition on the mutual coherence [35, 37] or the restricted isometry property (RIP) [11, 18, 22]. Although constructing matrices which satisfy a condition on the mutual coherence can be accomplished in a computationally efficient manner, it is NP-hard to compute both the nullspace constant (NSC) and restricted isometry constant (RIC) of a matrix [119]. On the other hand, it is recognised that stronger conditions can be proved via the nullspace property and RIP [47]. A perhaps surprising result is that a number of random matrix constructions, notably Gaussian, Bernoulli and partial Fourier, satisfy RIP with probability approaching one exponentially fast in the problem size [47]. Furthermore, not only are these random constructions popular from the perspective of deriving improved recovery guarantees, but also play a key role in applications: for instance Gaussian matrices achieve the optimal measurement rate $m = \mathcal{O}(k \log(k/n))$ with high probability while partial Fourier matrices are the natural sensing mechanism in Tomography [124].

Combinatorial compressed sensing (CCS) [14] analyses the compressed sensing problem in the setting where the encoder matrix is sparse and binary. Not only are these types of encoder matrix the relevant sensing mechanism in certain applications, e.g., the single pixel camera [40], but they also have benefits in terms of reduced computation and memory overheads compared with dense alternatives. Encoder matrices of this form are often interpreted as the adjacency matrices of an unbalanced bipartite graph. Furthermore, in [14] it was shown that when the encoder matrix is the adjacency matrix of a (k, ϵ, d)-expander graph then the optimal measurement rate $m = \mathcal{O}(k \log(k/n))$ can be achieved.

Definition 2.1.1 ((k, ϵ, d)-expander graph). *Consider a left d-regular bipartite graph $G = ([n], [m], E)$, for any $\mathcal{S} \subseteq [n]$ let $\mathcal{N}(\mathcal{S}) := \{j \in [m] : \exists l \in \mathcal{S} \text{ s.t. } (l, j) \in E\}$ be the subset of nodes in $[m]$ connected to a node in $\mathcal{S}$. With $\epsilon \in (0, 1)$, then G is a (k, ϵ, d)-expander graph iff*

$$|\mathcal{N}(\mathcal{S})| > (1 - \epsilon) d |\mathcal{S}| \quad \forall \quad \mathcal{S} \in [n]^{(\leq k)}. \tag{2.1}$$

Here $[n]^{(\leq k)}$ denotes the set of subsets of $[n]$ with cardinality at most k and ϵ is the expansion parameter, which plays a role analogous to that of the RIC of a matrix in proving recovery guarantees [14]. In this paper we define the adjacency matrix of a (k, ϵ, d)-expander graph $G([n], [m], E)$ as an $m \times n$ binary matrix $\mathbf{A}$ where $a_{j,l} = 1$

iff there is an edge between node $j \in [m]$ and $l \in [n]$, and is 0 otherwise[1]. In all that follows we will use $\mathcal{E}_{k,\epsilon,d}^{m\times n} \subset \{0,1\}^{m\times n}$ to denote the set of (k,ϵ,d)-expander graph adjacency matrices of dimension $m \times n$. The existence of optimal expanders, in the sense of achieving the optimal measurement rate $m = \mathcal{O}(k\log(k/m))$, was proved in [12, 26]. With regard to constructing such encoder matrices however, the current best deterministic constructions only achieve a measurement rate of $m = \mathcal{O}(k^{1+\gamma})$ for some constant $\gamma > 0$ [55]. As a result it is common to resort to random constructions, sampling A from a distribution so that with high probability $A \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$. A popular and natural construction in this regard is to sample A so that its columns are mutually independent and identically distributed, with the support of each column being chosen uniform at random from all possible supports of size d. Current state of the art bounds on the probability that this particular random matrix is a (k,ϵ,d)-expander which satisfies the optimal measurement rate can be found in [9]. In particular, we highlight the following asymptotic result.

Lemma 2.1.1 (**[9, Lemma 3.1]**). *Let A be an $m \times n$ random, binary matrix with mutually independent, identically distributed columns, whose supports are drawn uniform at random across all possible supports of cardinality d. For any $\epsilon \in (0, 1/2)$, suppose as $(k,m,n) \to \infty$ that $k/n \to \alpha_1$ and $m/n \to \alpha_2$, where $\alpha_1, \alpha_2 \in (0,1)$ are constants. As long as $\alpha_1 < (1-\gamma)\rho_{BT}(\alpha_2,\epsilon,d)$ for some constant $\gamma \in (0,1)$ then the probability that $A \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ approaches one exponentially fast in n.*

Here $\rho_{BT}(\alpha_2,\epsilon,d)$ denotes the phase transition curve, which is computed numerically, see [9, Equation 29]. Following [14], a series of iterative, greedy CCS algorithms were proposed: Sparse Matching Pursuit (SMP) [15], Sequential Sparse Matching Pursuit (SSMP) [16], Left Degree Dependent Signal Recovery (LDDSR) [123], Expander Recovery (ER) [70] and ℓ_0-decode [88, Algorithms 1 & 2]. These algorithms are specifically designed for the CCS setting and utilise certain key properties of expander graphs, namely the unique neighbour property.

Theorem 2.1.2 (**Unique neighbour property [26, Lemma 1.1]**). *Suppose that $G([n],[m],E)$ is an unbalanced, left d-regular bipartite graph. Let $\mathcal{S} \in [n]^{(\leq k)}$ and define*

$$\mathcal{N}_1(\mathcal{S}) = \{j \in \mathcal{N}(\mathcal{S}) : |\mathcal{N}(j) \cap \mathcal{S}| = 1\},$$

[1]We note that the adjacency matrices of graphs are often defined to describe the connectivity between all nodes in the graph, however, as we only consider bipartite graphs, in the definition adopted here the edges between nodes in the same group are not set to zero but rather are omitted entirely.

where $\mathcal{N}(j)$ is the subset of nodes of $[n]$ connected to the node $j \in [m]$. If $G([n],[m],E)$ is a (k,ϵ,d)-expander then

$$|\mathcal{N}_1(\mathcal{S})| > (1-2\epsilon)d|\mathcal{S}| \quad \forall \quad \mathcal{S} \in [n]^{(\leq k)}. \tag{2.2}$$

A proof of Theorem 2.1.2 in the notation used here is available in [88, Appendix A]. Critically, if $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ then the unique neighbour property ensures that certain entries in $\mathbf{x}$ appear repeatedly in $\mathbf{y} := \mathbf{A}\mathbf{x}$. This fact is exploited by CCS algorithms such as LDDSR, ER and ℓ_0-decode to guarantee that at each iteration a contraction in $||\mathbf{y}-\mathbf{A}\mathbf{x}||_0$ occurs. Moreover, the unique neighbour property is also used to prove that expander encoders satisfy RIP in the ℓ_1 norm [14].

The greedy, iterative algorithms for CCS that we have highlighted are highly efficient in practice. In particular, adopting the additional assumption that the sparse vector is dissociated, then [88] demonstrated that sparse recovery even for massive problems, i.e., $n = 2^{26}$, $m/n = 10^{-3}$ and $k/n = 0.3$, can be achieved in a matter of seconds using fairly standard computing infrastructure.

Definition 2.1.2 (**Dissociated vector, see Definition 4.32 of [117]**). *A vector $\mathbf{x} \in \mathbb{R}^N$ is said to be dissociated, which we will denote as $x \in \chi_k^n$, iff for any pair of subsets $\mathcal{T}_1, \mathcal{T}_2 \subseteq supp(\mathbf{x})$ it holds that $\sum_{j\in\mathcal{T}_1} x_j \neq \sum_{i\in\mathcal{T}_2} x_i$.*

Although at first glance this condition appears restrictive, it is fulfilled almost surely for isotropic vectors and more generally for any random vector whose nonzeros are drawn from a continuous distribution.

2.1.2 The Permuted Striped Block (PSB) model

If $\mathbf{x}$ is dissociated and $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$, then leveraging the unique neighbour property ℓ_0-decode [88, Algorithms 1 & 2] is able to read off entries of $\mathbf{x}$ directly from $\mathbf{y}$ based on the frequency with which they appear. The ease and manner with which these algorithms can recover a sparse code given access to the encoder matrix inspires the following question: is it possible to efficiently perform sparse recovery in the context of combinatorial compressed sensing without having access to the encoder matrix? Aside from trivial instances, given just a single measurement vector this task seems very challenging: although one might be able to identify certain entries in $\mathbf{y}$ as entries in $\mathbf{x}$, the overlap between columns in $\mathbf{A}$ will typically render the task impossible. Heuristically it seems possible that combining the information extracted from multiple measurement vectors may help in overcoming this issue. This leads us

to the following multi-measurement vector combinatorial compressed sensing (MMV-CSS) inspired matrix factorisation problem, in which we try to recover both the dissociated sparse codes $\mathbf{X} \in \chi_k^{n \times N}$, and the encoder matrix $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m \times n}$, from N measurement vectors, $\mathbf{Y} = \mathbf{AX}$. We will refer to such matrices $\mathbf{Y}$ as permuted striped block (PSB) matrices: in order to explain the origin of this name and to provide further intuition, observe that PSB matrices can be expressed as the sum of n rank one matrices of the form $\mathbf{a}_i \tilde{\mathbf{x}}_i$. Here $\tilde{\mathbf{x}}_i$ denotes the ith row of $\mathbf{X}$. Examples of matrices in this class are those that are a sum of n rank one matrices whose supports are a single dense block of size $d \times k$, which may or may not overlap one another. Other matrices in this set can then be generated by permuting the rows and columns of these rank one block matrices. Finally, these permuted blocks are striped in the sense that the entries in any given column of a block have the same value. We provide a visualisation in Figure 2.1.

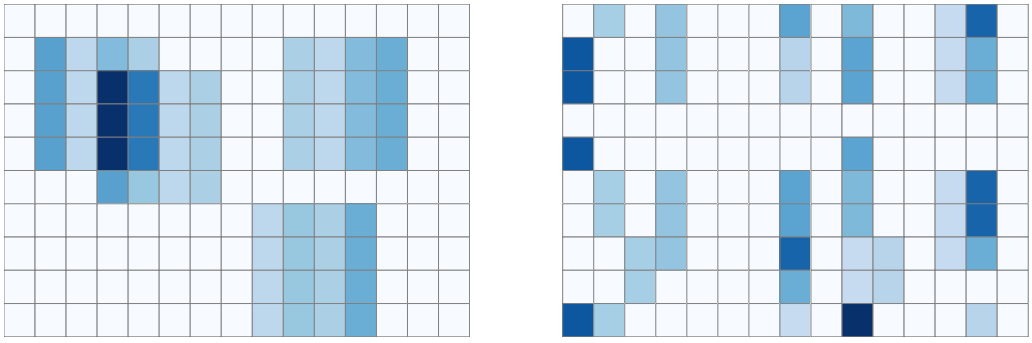


Figure 2.1: Visualisation of two examples of PSB matrices. In both cases the matrix consists of a sum of four rank 1 matrices, each with a support cardinality of 16. The left hand plot corresponds to the case where the support of each rank one matrix is arranged into a block. The right hand plot takes the same rank one matrices as in the left hand plot, but before summing them applies an independent, random row and column permutation to each. The white squares indicate a zero entry and nonzero entries in the same column have the same coefficient value. As a result, aside from where there are overlaps, each column of a block is a single stripe of colour.

An immediate issue presents itself in regard to recovering the original generating factors of a PSB matrix $\mathbf{Y}$. Observe, as is true for any matrix factorisation, that $\mathbf{Y} = \mathbf{AX} = \mathbf{AP}^T\mathbf{PX}$ for any permutation matrix $\mathbf{P} \in \mathcal{P}^{n \times n}$ [2] and therefore, without further information, it is not possible to distinguish the original generating factors from their permuted versions. For certain CS applications this is problematic as we will not be able to recover the locations of the nonzeros in the original sparse codes,

[2]Here and from now on we will use $\mathcal{P}^{n \times n}$ to denote the set of all $n \times n$ permutation matrices.

only their values. However, and as discussed in more detail in Section 2.1.3, with a small amount of additional information this problem can be resolved. Furthermore, in certain applications what is of interest are the statistics of the nonzero values rather than the full sparse code. In summary, the goal of this paper then is to derive and analyse algorithms which can efficiently recover $\mathbf{A}$ and $\mathbf{X}$ from $\mathbf{Y} = \mathbf{AX}$ up to permutation. To be clear, given some factorisation algorithm $F : \mathbb{R}^{m\times N} \to \mathcal{E}_{k,\epsilon,d}^{m\times n} \times \chi^{n\times N}$ and a PSB matrix $\mathbf{Y} := \mathbf{AX}$, with $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ and $\mathbf{X} \in \chi_k^{n\times N}$, then F successfully recovers $\mathbf{A}$ and $\mathbf{X}$ up to permutation from $\mathbf{Y}$ iff there exists a $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that $\hat{\mathbf{A}}, \hat{\mathbf{X}} := F(\mathbf{A}, \mathbf{X})$ satisfy $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$ and $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$.

Algorithm 1 GENERATE-ENCODER(m, n, d)

```
1:  A ← zeros(m, n)
2:  α ← ⌈m/d⌉
3:  β ← ⌈n/α⌉
4:  l ← 1
5:  for i = 1 : β do
6:      Q ∼ U(π([m]))
7:      for j = 1 : α do
8:          c ← (j − 1)d
9:          for r = 1 : d do
10:             A_{Q_{c+r},l} ← 1
11:         end for
12:         l ← l + 1
13:     end for
14: end for
15: Return A
```

In order to derive theoretical guarantees, and inspired by the random constructions used to generate optimal sensing expander graphs with high probability discussed in [9], we consider generating the encoder matrix using Algorithm 1. This algorithm takes as inputs the dimensions of the desired encoder m and n, the desired column sparsity d, and returns a binary matrix. The variable α stores the number of columns whose supports can be assigned using a single permutation of $[n]$, β is the number of permutations that are needed to be drawn in order to assign a support of cardinality d to each column of A and l is used as the column index of A. On line 6 a permutation of the set $[n]$ is drawn uniformly at random, note here that $Q \in [n]^n$ is a random vector holding this permutation and Q_i is the ith element of this vector. Note also that the draws of different permutations are considered to be mutually independent. The for loop running on lines 7-13 then uses the permutation Q to assign a support

to columns l through to $l + \alpha - 1$ of A. This is then repeated by drawing another permutation until all columns of A have been assigned a support. As a result, encoder matrices generated using Algorithm 1 have a fixed number d nonzeros per column and a maximum of $\lceil nd/m \rceil$ nonzeros per row. We note here that this upper bound on the number of nonzeros per row of A will be crucial for proving our recovery results in Section 2.3. Note also that the support of each column is dependant on the supports of at most $\lceil m/d \rceil - 1$ other columns, and that the intersection of these column supports is empty by construction. We are now ready to present the PSB model: this model places a distribution over a subset of PSB matrices in order to ensure with high probability that the encoder A, sampled using Algorithm 1, satisfies $A \in \mathcal{E}_{k,\epsilon,d}^{m \times n}$, and also that each column of A is equally likely to be represented in the measurement matrix.

Definition 2.1.3 (PSB model). *Let $d, k, m, n, N \in \mathbb{N}$ with $k = \alpha_1 n + 1$ and $m = \alpha_2 n$, where $\alpha_1, \alpha_2 \in (0, 1)$ are constants. Furthermore, suppose that there exists a constant $\gamma \in (0, 1)$ such that $\alpha_1 < (1 - \gamma)\rho_{BT}(\alpha_2, d, 1/6)$. Consider now the following random matrices.*

- *$A := [A_1 \ A_2 ... \ A_n]$ is a random binary matrix of size $m \times n$, sampled using Algorithm 1. The columns of A have exactly d ones while the rows have at most $\lceil nd/m \rceil$ ones by construction.*
- *$X := [X_1 \ X_2 ... \ X_N]$ is a random real matrix of size $n \times N$, whose columns are mutually independent, have exactly k nonzero entries, are dissociated and whose supports are chosen uniformly at random across all possible supports with cardinality k.*

A random, real $m \times N$ matrix Y is sampled from the PSB model, which we will denote as $Y \sim PSB(d, k, m, n, N)$, iff $Y := AX$.

We note here that in the parameter regime adopted in Definition 2.1.3 then Lemma 2.1.1 applies without modification to encoder matrices generated using Algorithm 1. This result follows in exactly the same manner as proved in [9] by observing that any pair of columns of A generated using Algorithm 1 either have independently drawn supports, as considered in the original case in [9], or, if they are dependent, then they are disjoint by construction. For brevity we do not replicate the original proof in detail here, instead referring the reader to [9] for further details.

2.1.3 Summary of contributions and potential applications

The contributions of this chapter are twofold: first, we provide a novel algorithm, Decoder-Expander Based Factorization (D-EBF), detailed in Algorithm 6 in Section 2.2.4, which is designed to recover $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ and $\mathbf{X} \in \chi_k^{n\times N}$ up to permutation from $\mathbf{Y} := \mathbf{AX}$. Second, we analyse the performance of D-EBF in the context of the PSB model, proving that it recovers the matrix factors up to permutation with high probability. The theoretical guarantees for D-EBF are summarised in Theorem 2.1.3.

Theorem 2.1.3. *Let $Y \sim PSB(d,k,m,n,N)$ as per Definition 2.1.3. Under the assumption that $A \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$, consider the reconstructions of the matrix factors of Y returned by D-EBF,*

$$\hat{A}, \hat{X} \leftarrow \text{D-EBF}(Y, m, n, N, \epsilon, d).$$

Then the following statements are true.

1. ***The reconstructions are accurate up to permutation:*** *there exists a random permutation $P \in \mathcal{P}^{n\times n}$ such that $\text{supp}(\hat{A}P^T) \subseteq \text{supp}(A)$, $\text{supp}(P\hat{X}) \subseteq \text{supp}(X)$ and $\hat{x}_{j,i} = x_{j,i}$ for all $(j,i) \in \text{supp}(\hat{X})$.*

2. ***On the uniqueness of the factorisation:*** *if $Y = \hat{A}\hat{X}$ then this factorisation is unique up to permutation, i.e., there exists a random permutation $P \in \mathcal{P}^{n\times n}$ such that $\hat{A}P^T = A$ and $P\hat{X} = X$.*

3. ***D-EBF is successful with high probability in n:*** *Suppose in addition to the assumptions of the PSB model that $\alpha_1 \leq 1 - \frac{d}{m}$, $N \geq \nu \frac{4d}{3\tau(n)} \frac{n}{k}(\ln^2(n) + \ln(n))$, where $\tau(n) = \mathcal{O}(1)$ [3] and $\nu > 1$ is a constant. Then the probability that there exists a permutation $P \in \mathcal{P}^{n\times n}$ such that $\hat{A}P^T = A$ and $P\hat{X} = X$ is greater than $1 - \mathcal{O}\left(n^{-\sqrt{\nu}+1}\log^2(n)\right)$*

A few remarks regarding the D-EBF algorithm and Theorem 4.1.1 are in order.

- **Probability that A is an expander:** recall from the discussion in Section 2.1.2 that, with A defined as in the PSB model, then the probability that $A \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ goes to one exponentially as $n \to \infty$. Furthermore, the probability of this event can be lower bounded using the bounds derived in [9]. As a result, from the definition of conditional probability, upper bounds on the probability of all three statements of Theorem 4.1.1 can easily be derived.

[3] The definition of $\tau(n)$ can be found in Lemma 2.3.5

It therefore also follows that the three statements of Theorem 4.1.1 still hold with high probability even without conditioning on the event $A \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$. We have opted not to present Theorem 4.1.1 in this form for two reasons: first in order to more clearly highlight the contributions of this paper and second because the lower bound on the probability that $A \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ derived in [9] is loose and overly pessimistic.

- **Sample complexity of D-EBF:** perhaps the most remarkable aspect of Theorem 4.1.1 is the required sample complexity. Indeed, observing that ν and d are constants and that $\tau(n) = \mathcal{O}(1)$, $n/k = \mathcal{O}(1)$, then as long as $N = \Omega(\log^2(n))$ D-EBF succeeds in recovering A and X up to permutation with high probability. Note that as each column of $\mathbf{Y}$ is a weighted sum of k columns of $\mathbf{A}$, which itself has n columns, then $N \geq n/k = \mathcal{O}(1)$ is necessary in order for $\mathbf{Y}$ to have at least one contribution from each column of $\mathbf{A}$. This is equal to the stated sample complexity $N = \Omega(\log^2(n))$ up to logarithmic factors. We hypothesise that this asymptotic lower bound is likely to be optimal as one $\log(n)$ factor arises from a coupon collector argument, inherent to the way X is sampled, and the second $\log(n)$ factor is needed to achieve the stated rate of convergence in probability.

- **Computational complexity of D-EBF:** in Section 2.2.4 we will ascertain that the per while loop iteration complexity cost of D-EBF is $\mathcal{O}(k^2(n+N)N)$. Analysing and bounding the number of iterations of the while loop of D-EBF in a meaningful manner is challenging and beyond the scope of Theorem 2.1.3. Our preliminary experimental investigations on moderately sized problems indicate that the number of iterations required in practice is not onerous and, as expected, reduces as N increases. We leave a proper analysis to future work.

- **Input arguments required by D-EBF:** observe that D-EBF requires not only the product matrix Y and its dimensions, but also certain parameters of the encoder matrix, namely the number of columns n, the expansion parameter ϵ and the column sparsity d. As will be discussed in Section 2.4, in practice n is not required. Indeed, ND-EBF can be run in an online fashion, processing input vectors as and when they are received and dynamically updating a list of column vectors representing $\hat{\mathbf{A}}$. The column sparsity d is required to be known in advance in order to generate the encoder matrix, as a result the assumption that this can be passed to or known in advance by the decoder seems reasonable.

The same cannot be said of the expansion parameter ϵ, as even with access to $\mathbf{A}$ computing ϵ is an NP-complete problem [3]. However, as discussed and demonstrated empirically in section 2.4, in practice ND-EBF can still function effectively given access only to an upper bound on ϵ instead. In our experiments we use $1/6$ for simplicity but tighter upper bounds on the expansion parameter can be computed with relative ease, see e.g., [67].

As shall be discussed in more detail in Section 2.4, as long as $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ and N is sufficiently large, then in practice D-EBF requires only knowledge of the column sparsity d in advance to compute the factors $\mathbf{A}$ and $\mathbf{X}$ of $\mathbf{Y}$ up to permutation. In the context of multi-measurement vector combinatorial compressed sensing (MMV-CCS), the D-EBF algorithm is only able to recover the sparse codes up to permutation, meaning the location of the nonzeros is lost. However, if the encoder and decoder agree an ordering of the columns of the encoder matrix in advance then this issue can be avoided. One such protocol to this end is as follows.

1. The encoder generates an encoder matrix A as per the PSB model and then, interpreting each column as a rational number written in binary, reorders the columns by the size of their corresponding rational number, e.g., from largest to smallest.

2. The encoder then generates Y by multiplying X by the ordered encoder matrix and then sends Y and d to the decoder.

3. The decoder applies the D-EBF algorithm to recover the encoder matrix and sparse codes up to permutation. The decoder then interprets each of the columns of the reconstructed encoder matrix as a rational number written in binary, and reorders the columns of $\hat{A}$ and the corresponding rows of $\hat{X}$ by the size of their corresponding rational number.

This protocol allows for full sparse recovery in the (MMV-CCS) setting without knowledge of the encoder matrix beyond the column sparsity d and an ordering protocol for the columns.

Even without such a protocol, from the permuted sparse code one is still able to recover and compute key statistics concerning the original sparse code, for example the average value of the nonzero entries, the maximum value or a histogram of the values. In addition, removing the location information may even provide certain benefits in some applications, e.g., ensuring privacy of users. As will be discussed in

Section 2.4, D-EBF can be adapted to operate in an online fashion. In this setting one could consider the columns of $\mathbf{Y}$ as linear measurement vectors of a system's state vector, which evolves in time and is sparse. Examples of such systems might include a network of sensors or the actions of users in a social network. Assuming that this system is measured using a suitable encoder matrix, which is fixed across time, then Theorem 4.1.1 implies that given sufficient samples then the activation statistics of all samples to date, as well as all future ones, could be computed without ever having explicit access to the encoder matrix. This would be necessary in situations where the encoder matrix is unknown by or private from the would be observer of the system, and may be advantageous from an efficiency perspective if the encoder matrix is costly or difficult to transmit.

2.1.4 Related work

In regard to connections with other matrix factorisation problems and methods, the PSB model and associated factorisation task are most closely related to those in which sparsity is also a prominent feature, such as in dictionary learning and subspace clustering. Dictionary learning [2, 42, 78, 82, 93] is a prominent matrix factorisation technique in data science, in which the columns of the observed data matrix are assumed to lie, at least approximately, on a union of low dimensional subspaces. This structure can be expressed as the product of an overcomplete matrix, known as a dictionary, and a sparse matrix or code. To be clear, given a data matrix $\mathbf{Y}$ dictionary learning methods seek to compute a dictionary $\mathbf{A}$ and a sparse coding matrix $\mathbf{X}$ such that $\mathbf{Y} \approx \mathbf{AX}$. Compared with dictionary learning, subspace clustering [44, 98, 122] adopts the additional structural assumption that the data lies on a union of low dimensional subspaces which are independent of one another.

More specifically, the literature most relevant to this work is that on dictionary recovery, a field aiming to provide recovery guarantees for dictionary learning. Compared with dictionary learning, dictionary recovery [1, 4, 6, 10, 112, 115, 116] presupposes that $\mathbf{Y} := \mathbf{AX}$ with the goal being to recover $\mathbf{A}$ and $\mathbf{X}$ up to permutation. It is common in dictionary recovery to place further structural assumptions on the factor matrices so as to facilitate the development of strong guarantees, albeit at the expense of model expressiveness. Two popular structural assumptions are that the dictionary is complete [112, 115, 116] or incoherent [1, 6]. Similar to the PSB model, the sparse coding matrix is often assumed to have been drawn from a sparse distribution, e.g., Bernoulli-Gaussian [1, 6] or Bernoulli-uniform [112, 115, 116]. This work diverges from the majority of the literature on dictionary recovery in regard to

the construction of the dictionary: in particular, while prior work uses incoherence or completeness to design algorithms and derive guarantees we instead operate under the assumption that the encoder matrix is the adjacency matrix of a (k, ϵ, d)-expander graph. We therefore use a very different set of ideas and tools in our algorithmic approach and proof technique.

In the context of matrix factorisation and dictionary recovery, we are aware of only one other work in which A is constructed so as to leverage the properties of expander graphs [5]. In this work the authors, guided by the notion of reversible neural networks, considered the learning of a deep network as a layerwise nonlinear dictionary learning problem. This work differs substantially from [5] in a number of respects. First, in terms of the problem setup, the primary difference is that in [5] the problem of recovering A and X from $Y = \sigma(AX)$ is studied, where σ denotes the elementwise unit step function and both X and Y are binary. As a result, this data model is designed for binary data. Second, in regard to algorithmic approach, in this prior work A is recovered up to permutation using a non-iterative approach, involving first the computation of all row wise correlations of Y and then applying a clustering technique adapted from the graph square root problem. By contrast, and as described in detail in Section 2.2.4, our method iteratively recovers parts of A and X directly from the columns of the residual by simultaneously leveraging both the unique neighbour property of A and the fact that the columns of X are dissociated.

2.2 Decoder-Expander Based Factorisation (D-EBF)

In this section we present the Decoder-Expander Based Factorisation (D-EBF) algorithm as a method for solving the PSB matrix factorisation problem introduced and defined in Section 2.1.2. In Section 2.2.1 we provide a high level overview of the D-EBF algorithm, before describing in detail the key steps in Sections 2.2.2 and 2.2.3. Finally, we provide a full definition of D-EBF in Section 2.2.4.

2.2.1 Overview of the D-EBF algorithm

The D-EBF algorithm iteratively computes a reconstruction of $\mathbf{A}$ and $\mathbf{X}$ by recovering parts of them from the observed matrix $\mathbf{Y}$, removing the contributions of these parts to form a residual, and then repeating these steps on said residual. The D-EBF algorithm starts each iteration by searching, independently, the columns of the residual for singleton values - nonzeros which are the sum of one nonzero in the corresponding column of $\mathbf{X}$. In Section 2.2.2 we prove, under certain assumptions, that singleton

values can be identified by the number of times they appear in a given column of the residual. In addition, as the columns of $\mathbf{X}$ are dissociated, then the locations in which a singleton value appears indicate part of the support of a column of $\mathbf{A}$. Furthermore, if a singleton value appears in sufficiently many locations then the associated partial support is unique to one column of $\mathbf{A}$. After identifying these frequently occurring singleton values from the columns of the residual, the D-EBF algorithm clusters the associated partial supports by their unique column of $\mathbf{A}$. The union of the partial supports in each cluster is then used to reconstruct, at least partially, a column of $\mathbf{A}$. The completely recovered columns, i.e., those whose supports have cardinality d, are then used to recover further entries of $\mathbf{X}$ using a sparse decoding algorithm from the combinatorial compressed sensing literature [15, 16, 88, 123, 70]. Finally, the contributions from the recovered parts of $\mathbf{A}$ and $\mathbf{X}$ are removed from $\mathbf{Y}$ to form a new residual. This process is then repeated on the subsequent residuals until either no additional entries from $\mathbf{A}$ or $\mathbf{X}$ are recovered or $\mathbf{Y}$ is fully decoded. We summarise this high level approach in Algorithm 2.

In the following sections we proceed to define each of the high level steps of Algorithm 2 in detail. In Section 2.2.2 we study steps 3, 4 and 5, which concern the extraction and clustering of singleton values and partial supports. In section 2.2.3 we consider step 6, the decoding step, and review sparse decoding algorithms from the combinatorial compressed sensing literature. Finally, in section 2.2.4 we present a detailed version of Algorithm 2.

Algorithm 2 High level overview of the D-EBF algorithm

1: Initialise residual as observed data $\mathbf{Y}$
2: **while** not converged **do**
3: Extract singleton values and associated partial supports from the residual
4: Cluster the partial supports by their column of $\mathbf{A}$
5: Update the reconstructions of $\mathbf{A}$ and $\mathbf{X}$ using the clustered partial supports and the singleton values respectively
6: Run a decode algorithm using the fully recovered columns of $\mathbf{A}$ to recover more entries of $\mathbf{X}$.
7: Update residual by removing contributions from recovered parts of $\mathbf{A}$ and $\mathbf{X}$
8: **end while**

2.2.2 Singleton values and partial supports

As referenced to in Section 2.1.1, expander graphs play major key role both in the design of D-EBF as well as the proof of Theorem 2.1.3. In Lemma 2.2.1 we summarise

key properties of the adjacency matrices of (k, ϵ, d)-expander graphs.

Lemma 2.2.1 (Properties of the adjacency matrix of a (k, ϵ, d)-expander graph). *If $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ then any submatrix $\mathbf{A}_{\mathcal{S}}$ of $\mathbf{A}$, where $\mathcal{S} \in [n]^{(\leq k)}$, satisfies the following.*

1. *There are more than $(1-\epsilon)d|\mathcal{S}|$ rows in $\mathbf{A}_{\mathcal{S}}$ that have **at least one** non-zero.*
2. *There are more than $(1-2\epsilon)d|\mathcal{S}|$ rows in $\mathbf{A}_{\mathcal{S}}$ that have **only one** non-zero.*
3. *The overlap in support of the columns of $\mathbf{A}_{\mathcal{S}}$ is upper bounded as*

$$|\bigcap_{l\in\mathcal{S}} \text{supp}(\mathbf{a}_l)| < 2\epsilon d.$$

Proof. Property 1 follows directly from Definition 2.1.1 and property 2 from Theorem 2.1.2. To prove property 3, consider the pairwise overlap in support between any two distinct columns $\mathbf{a}_l$ and $\mathbf{a}_h$ of $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$. From Definition 2.1.1 it follows that

$$|supp(\mathbf{a}_l) \cup supp(\mathbf{a}_h)| > (1-\epsilon)2d.$$

Using the inclusion-exclusion principle then

$$|supp(\mathbf{a}_l)| + |supp(\mathbf{a}_h)| - |supp(\mathbf{a}_l) \cap supp(\mathbf{a}_h)| > (1-\epsilon)2d.$$

Given that $|supp(\mathbf{a}_l)| = |supp(\mathbf{a}_h)| = d$, then a simple rearrangement shows that

$$|supp(\mathbf{a}_l) \cap supp(\mathbf{a}_h)| < 2d - (1-\epsilon)2d = 2\epsilon d.$$

Therefore, for any $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| \geq 2$ and for any $l, h \in \mathcal{S}$ such that $l \neq h$, then

$$|\bigcap_{l\in\mathcal{S}} supp(\mathbf{a}_l)| \leq |supp(\mathbf{a}_l) \cap supp(\mathbf{a}_h)| < 2\epsilon d.$$

□

The inspiration for studying singleton values arises from property 2 of Lemma 2.2.1. Indeed, by conditioning on $\mathbf{A}$ being the adjacency matrix of a (k, ϵ, d)-expander graph, then the submatrix of $\mathbf{A}$ associated with the support of a column of $\mathbf{X}$ has potentially many rows with a single nonzero entry. This implies that there are potentially many entries in each column of $\mathbf{Y}$ which correspond directly to an entry in $\mathbf{X}$. Recalling that the jth row of $\mathbf{A}$ is denoted as $\tilde{\mathbf{a}}_j$, then we are now ready to provide the definition of a singleton value.

Definition 2.2.1 (**Singleton value**). *Consider a vector* $\mathbf{r} = \mathbf{A}\mathbf{z}$ *where* $\mathbf{A} \in \{0,1\}^{m\times n}$ *and* $\mathbf{z} \in \mathbb{R}^n$. *A singleton value of* $\mathbf{r}$ *is an entry* $r_j \neq 0$ *such that* $|\operatorname{supp}(\tilde{\mathbf{a}}_j) \cap \operatorname{supp}(\mathbf{z})| = 1$, *hence* $r_j = x_l$ *for some* $l \in \operatorname{supp}(\mathbf{z})$.

We call a binary vector whose nonzeros coincide with some subset of the nonzeros of a column of $\mathbf{A}$ a partial support of that column.

Definition 2.2.2 (**Partial support**). *A partial support* $\mathbf{w} \in \{0,1\}^m$ *of a column* $\mathbf{a}_l \in \{0,1\}^m$ *of* $\mathbf{A} \in \{0,1\}^{m\times n}$ *is a binary vector satisfying* $\operatorname{supp}(\mathbf{w}) \subseteq \operatorname{supp}(\mathbf{a}_l)$. *Furthermore* $\mathbf{w}$ *is said to originate from* $\mathbf{a}_l$ *iff* $\operatorname{supp}(\mathbf{w}) \subseteq \operatorname{supp}(\mathbf{a}_l)$ *and* $\operatorname{supp}(\mathbf{w}) \nsubseteq \operatorname{supp}(\mathbf{a}_h)$ *for all* $h \in [n]\backslash\{l\}$.

We now introduce some useful notation for what follows.

- Let the function $f : \mathbb{R} \times \mathbb{R}^m \to [m]$ count the number of times a real number $\alpha \in \mathbb{R}$ appears in some vector $\mathbf{r} \in \mathbb{R}^m$, $f(\alpha, \mathbf{r}) := |\{j \in [m] : r_j = \alpha\}|$.
- Let the function $g : \mathbb{R} \times \mathbb{R}^m \to \{0,1\}^m$ return a binary vector whose nonzeros correspond to the locations in which $\alpha \in \mathbb{R}$ appears in a vector $\mathbf{r} \in \mathbb{R}^m$, i.e., with $\mathbf{w} = g(\alpha, \mathbf{r})$ then $w_j = 1$ iff $r_j = \alpha$ and is 0 otherwise.

Observe that if $\mathbf{z}$ is dissociated then the locations in which a singleton value appears in $\mathbf{r}$ defines a partial support of a column of $\mathbf{A}$. Furthermore, under the assumption that $\mathbf{A}$ is the adjacency matrix of a (k, ϵ, d)-expander graph and that $\mathbf{z}$ is dissociated, it is possible to derive the following sufficient condition with which to identify singleton values.

Corollary 2.2.1.1 (**Sufficient condition for identifying singleton values (i)**). *Consider a vector* $\mathbf{r} = \mathbf{A}\mathbf{z}$ *where* $\mathbf{A} \in \mathcal{E}^{m\times n}_{k,\epsilon,d}$ *and* $\mathbf{z} \in \chi^n_k$. *Then for* $\alpha \in \mathbb{R}\backslash\{0\}$ *if* $f(\alpha, \mathbf{r}) \geq 2\epsilon d$ *then there exists an* $l \in \operatorname{supp}(\mathbf{x})$ *such that* $\alpha = x_l$.

Proof. By construction if $\alpha \in \mathbb{R}\backslash\{0\}$ and $f(\alpha, \mathbf{r}) > 0$ then there exists a $\mathcal{S} \subseteq \operatorname{supp}(\mathbf{z})$ such that $\alpha = \sum_{l\in\mathcal{S}} x_l$. Suppose that $|\mathcal{S}| \geq 2$. Then

$$\begin{aligned} f(\alpha, \mathbf{r}) &= |\{j \in [m] : \mathcal{S} = \operatorname{supp}(\tilde{\mathbf{a}}_j) \cap \operatorname{supp}(\mathbf{z})\}| \\ &\leq |\{j \in [m] : \mathcal{S} \subseteq \operatorname{supp}(\tilde{\mathbf{a}}_j) \cap \operatorname{supp}(\mathbf{z})\}| \\ &\leq |\{j \in [m] : \mathcal{S} \subseteq \operatorname{supp}(\tilde{\mathbf{a}}_j)\}| \\ &= |\bigcap_{l\in\mathcal{S}} \operatorname{supp}(a_l)| \\ &< 2\epsilon d, \end{aligned}$$

where the final inequality follows from Lemma 2.2.1. Therefore if $\alpha \in \mathbb{R}\backslash\{0\}$ satisfies $f(\alpha, \mathbf{r}) \geq 2\epsilon d$ it must follow that α is a singleton value, i.e., $|\mathcal{S}| = 1$, and therefore there exists an $l \in \operatorname{supp}(\mathbf{x})$ such that $\alpha = x_l$. □

Under the assumptions stated this corollary implies that any value which appears in $\mathbf{r}$ at least $2\epsilon d$ times must be a singleton value. However, this corollary does not provide insight into whether or not there exist singletons values in $\mathbf{r}$ appearing in at least $2\epsilon d$ locations. To resolve this matter, in Lemma 2.2.2, adapted from [88, Theorem 4.6], we show, under certain assumptions, that there always exist a positive number of singleton values which appear more than $(1-2\epsilon)d$ times in $\mathbf{r}$. This result is needed for the proof of Theorem 2.1.3.

Lemma 2.2.2 (**Existence of frequently occurring singleton values, adapted from [88, Theorem 4.6]**). *Consider a vector* $\mathbf{r} = \mathbf{A}\mathbf{z}$ *where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *and* $\mathbf{z} \in \chi_k^n$. *For* $l \in \operatorname{supp}(\mathbf{z})$, *let* $\Omega_l := \{j \in [m] : r_j = z_l\}$ *be the set of row indices* $j \in [m]$ *of* $\mathbf{r}$ *such that* $r_j = z_l$. *Defining* $\mathcal{T} := \{l \in \operatorname{supp}(\mathbf{z}) : |\Omega_l| > (1-2\epsilon)d\}$ *as the set of singleton values which appear more than* $(1-2\epsilon)d$ *times in* $\mathbf{r}$, *then*

$$|\mathcal{T}| \geq \frac{|\operatorname{supp}(\mathbf{z})|}{(1+2\epsilon)d}.$$

Proof. For typographical ease let $\gamma := \lceil (1-2\epsilon)d \rceil$ and define $U := \operatorname{supp}(\mathbf{z})\backslash\mathcal{T}$, which is the set of singleton values which appear at most $(1-2\epsilon)d$ times. Additionally, for this proof we adopt the following graph inspired notation.

- $\mathcal{N}_1(\mathbf{r}) := \bigcup_{l\in\operatorname{supp}(\mathbf{z})} \Omega_l$ is the set of row indices of $\mathbf{r}$ corresponding to singleton values, by Theorem 2.1.2 it holds that $|\mathcal{N}_1(\mathbf{r})| \geq \gamma|\operatorname{supp}(\mathbf{z})|$.
- $\mathcal{N}_1^{\mathcal{T}}(\mathbf{r}) := \bigcup_{l\in\mathcal{T}} \Omega_l$ is the set of row indices of $\mathbf{r}$ corresponding to singleton values that appear more than $(1-2\epsilon)d$ times. Therefore $\gamma|\mathcal{T}| \leq |\mathcal{N}_1^{\mathcal{T}}(\mathbf{r})| \leq d|\mathcal{T}|$.
- $\mathcal{N}_1^{\mathcal{U}}(\mathbf{r}) := \bigcup_{l\in\mathcal{U}} \Omega_l$ is the set of row indices of $\mathbf{r}$ corresponding to singleton values that appear at most $(1-2\epsilon)d$ times. Note that $|\mathcal{N}_1^{\mathcal{U}}(\mathbf{r})| \leq (\gamma-1)|\mathcal{U}|$.

Clearly $|\mathcal{N}_1(\mathbf{r})| = |\mathcal{N}_1^{\mathcal{T}}(\mathbf{r})| + |\mathcal{N}_1^{\mathcal{U}}(\mathbf{r})|$, it therefore follows that

$$\begin{aligned}
|\mathcal{N}_1(\mathbf{r})| &\geq \gamma|\operatorname{supp}(\mathbf{z})| \\
&= \gamma(|\mathcal{T}| + |\mathcal{U}|) \\
&= \gamma|\mathcal{T}| + |\mathcal{U}| + (\gamma-1)|\mathcal{U}| \\
&\geq \gamma|\mathcal{T}| + |\mathcal{U}| + |\mathcal{N}_1^{\mathcal{U}}(\mathbf{r})|.
\end{aligned}$$

Substituting $|\mathcal{N}_1(\mathbf{r})| = |\mathcal{N}_1^{\mathcal{T}}(\mathbf{r})| + |\mathcal{N}_1^{\mathcal{U}}(\mathbf{r})|$ then

$$\begin{aligned}|\mathcal{N}_1^{\mathcal{T}}(\mathbf{r})| &\geq \gamma|\mathcal{T}| + |\mathcal{U}| \\ &= |\mathcal{T}| + |\mathcal{U}| + (\gamma - 1)|\mathcal{T}| \\ &= |\operatorname{supp}(\mathbf{z})| + (\gamma - 1)|\mathcal{T}|.\end{aligned}$$

As $|\mathcal{N}_1^{\mathcal{T}}(\mathbf{r})| \leq d|\mathcal{T}|$ then $d|\mathcal{T}| \geq |\operatorname{supp}(\mathbf{z})| + (\gamma - 1)|\mathcal{T}|$, rearranging gives

$$|\mathcal{T}| \geq \frac{|\operatorname{supp}(\mathbf{z})|}{d - \gamma + 1}.$$

Finally, as $\gamma \geq d - 2\epsilon d$ then

$$|\mathcal{T}| \geq \frac{|\operatorname{supp}(\mathbf{z})|}{1 + 2\epsilon d}$$

as claimed. Note in addition that so long as $\mathbf{r}$ is nonzero, meaning $|\operatorname{supp}(\mathbf{z})| \geq 1$, then it is always possible to extract at least one partial support with cardinality greater than $(1 - 2\epsilon)d$. □

If $\epsilon \leq 1/4$ then $(1 - 2\epsilon)d \geq 2\epsilon d$. Therefore, under this and the other necessary assumptions, for any nonzero $\mathbf{r}$ Corollary 2.2.1.1 and Lemma 2.2.2 together guarantee that it is always possible to identify at least one singleton value which appears more than $2\epsilon d$ times in $\mathbf{r}$. The results presented so far concerning the extraction of singleton values and partial supports assume that the residual analysed is of the form $\mathbf{r} = \mathbf{A}\mathbf{z}$, where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ and $\mathbf{z} \in \chi_k^n$. Due to the iterative nature of D-EBF, typically the residual under consideration is instead of the form $\mathbf{r} = \mathbf{y} - \hat{\mathbf{A}}\hat{\mathbf{x}}$, where $\hat{\mathbf{A}}$ and $\hat{\mathbf{x}}$ are the estimates or reconstructions of $\mathbf{A}$ and $\mathbf{x}$ respectively. Under certain assumptions, the following result maintains that the sufficient condition given in Corollary 2.2.1.1 also hold for residuals of this form.

Lemma 2.2.3 (**Sufficient condition for identifying singleton values (ii)**). *Consider a vector* $\mathbf{y} = \mathbf{A}\mathbf{x}$ *where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *and* $\mathbf{x} \in \chi_k^n$. *Let* $\hat{\mathbf{A}} \in \{0,1\}^{m\times n}$ *and* $\hat{\mathbf{x}} \in \chi_k^n$ *be such that a column* $\hat{\mathbf{a}}_l$ *of* $\hat{\mathbf{A}}$ *is nonzero iff* $\hat{x}_l \neq 0$, *and there exists a permutation matrix* $\mathbf{P} \in \mathcal{P}^{n\times n}$ *such that* $\operatorname{supp}(\hat{\mathbf{A}}\mathbf{P}^T) \subseteq \operatorname{supp}(\mathbf{A})$, $\operatorname{supp}(\mathbf{P}\hat{\mathbf{x}}) \subseteq \operatorname{supp}(\mathbf{x})$ *and* $\hat{x}_{P(l)} = x_{P(l)}$ *for all* $l \in supp(\hat{\mathbf{x}})$, *where* $P : [n] \to [n]$ *denotes the row permutation caused by pre-multiplication with* $\mathbf{P}$. *Consider the residual* $\mathbf{r} = \mathbf{y} - \hat{\mathbf{A}}\hat{\mathbf{x}}$, *if for some* $\alpha \in \mathbb{R}\backslash\{0\}$ *it holds that* $f(\alpha, \mathbf{r}) \geq 2\epsilon d$ *is satisfied, then there exists an* $l \in [n]$ *such that* $\alpha = x_l$. *Furthermore, with* $\mathbf{w} = g(\alpha, \mathbf{r})$ *then* $\operatorname{supp}(\mathbf{w}) \subseteq \operatorname{supp}(\mathbf{a}_l)$ *is a partial support of* $\mathbf{a}_l$.

Proof. As in Corollary 2.2.1.1, then by construction if $\alpha \in \mathbb{R}\backslash\{0\}$ and $f(\alpha, \mathbf{r}) > 0$ then there exists a $\mathcal{S} \subseteq \text{supp}(\mathbf{z})$ such that $\alpha = \sum_{l\in\mathcal{S}} x_l$. In order to simplify our notation for what follows, let $\mathbf{V} := \hat{\mathbf{A}}\mathbf{P}^T$ and $\mathbf{u} := \mathbf{P}\hat{\mathbf{x}}$. By construction $\text{supp}(\mathbf{V}) \subseteq \text{supp}(\mathbf{A})$ and $\text{supp}(\mathbf{u}) \subseteq \text{supp}(\mathbf{x})$. In addition, with $\Omega_j := \text{supp}(\tilde{\mathbf{a}}_j) \cap \text{supp}(\mathbf{x})$ and $\Gamma_j := \text{supp}(\tilde{\mathbf{v}}_j) \cap \text{supp}(\mathbf{u})$, then $\Gamma_j \subseteq \Omega_j$ for all $j \in [m]$. As $u_l = x_l$ for all $l \in \text{supp}(\mathbf{u})$ and $\mathbf{r} = \mathbf{y} - \mathbf{V}\mathbf{u}$ then

$$r_j = \sum_{l\in\Omega_j} x_l - \sum_{h\in\Gamma_j} x_h = \sum_{l\in(\Omega_j\backslash\Gamma_j)} x_l.$$

It therefore follows that each entry r_j is a sum over a subset of the nonzero values of $\mathbf{x}$ used in the sum of y_j. As a result, we may write $\mathbf{r} := \mathbf{B}\mathbf{x}$ where $\mathbf{B} \in \{0,1\}^{m\times n}$ and $\text{supp}(\mathbf{B}) \subseteq \text{supp}(\mathbf{A})$. Therefore, for any $\mathcal{S} \subseteq \text{supp}(\mathbf{x})$ such that $|\mathcal{S}| \geq 2$, then from Lemma 2.2.1 it follows that

$$|\bigcap_{l\in\mathcal{S}} \text{supp}(\mathbf{b}_l)| \leq |\bigcap_{l\in\mathcal{S}} \text{supp}(\mathbf{a}_l)| < 2\epsilon d.$$

Therefore, if $\alpha = \sum_{l\in\mathcal{S}} x_l$ with $|\mathcal{S}| \geq 2$ then $f(\alpha, \mathbf{r}) < 2\epsilon d$. As a result, for some $\alpha \in \mathbb{R}$ if $f(\alpha, \mathbf{r}) \geq 2\epsilon d$ then α is a singleton and there exists an $l \in [n]$ such that $\alpha = x_l$. With $\mathbf{w} = g(\alpha, \mathbf{r})$, then as $\mathbf{x}$ is dissociated it follows that $\text{supp}(\mathbf{w}) \subseteq \text{supp}(\mathbf{b}_l) \subseteq \text{supp}(\mathbf{a}_l)$ and so $\mathbf{w}$ is a partial support of $\mathbf{a}_l$. □

To perform step 4 of Algorithm 2, it is necessary that the partial supports extracted across all columns of $\mathbf{Y}$, or the residual of $\mathbf{Y}$, can be clustered accurately and efficiently. To this end we present Corollary 2.2.4, which provides a necessary and sufficient condition with which to cluster sufficiently large partial supports.

Lemma 2.2.4 (**Clustering partial supports**). *Consider a pair of partial supports* $\mathbf{w}_1$ *and* $\mathbf{w}_2$*, extracted from* $\mathbf{r}_1 = \mathbf{A}\mathbf{z}_1$ *and* $\mathbf{r}_2 = \mathbf{A}\mathbf{z}_2$ *respectively, where* $\mathbf{A} \in \mathcal{E}^{m\times n}_{k,\epsilon,d}$ *and* $\mathbf{z}_1, \mathbf{z}_2 \in \chi^n_k$*. If* $\epsilon \leq 1/6$*,* $|\,\text{supp}(\mathbf{w}_1)| > (1-2\epsilon)d$ *and* $|\,\text{supp}(\mathbf{w}_2)| > (1-2\epsilon)d$*, then* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *originate from the same column of* $\mathbf{A}$ *iff* $\mathbf{w}_1^T\mathbf{w}_2 \geq 2\epsilon d$*.*

Proof. Suppose that $\mathbf{w}_1$ and $\mathbf{w}_2$ originate from ith and jth column of $\mathbf{A}$ respectively. From Lemma 2.2.1, and observing that $|supp(\mathbf{a}_i) \cap supp(\mathbf{a}_j)| = \mathbf{a}_i^T\mathbf{a}_j$ for binary vectors $\mathbf{a}_i$ and $\mathbf{a}_j$, then $i \neq j$ iff $\mathbf{a}_i^T\mathbf{a}_j < 2\epsilon d$. Observe also that as $\text{supp}(\mathbf{w}_1) \subseteq \text{supp}(\mathbf{a}_i)$ and $\text{supp}(\mathbf{w}_2) \subseteq \text{supp}(\mathbf{a}_j)$ then $\mathbf{w}_1^T\mathbf{w}_2 \leq \mathbf{a}_i^T\mathbf{a}_j$. Suppose that $\mathbf{w}_1^T\mathbf{w}_2 \geq 2\epsilon d$ and assume that $i \neq j$. This is a contradiction however as it must hold that $\mathbf{a}_i^T\mathbf{a}_j < 2\epsilon d$, but

$$\mathbf{a}_i^T\mathbf{a}_j \geq \mathbf{w}_1^T\mathbf{w}_2 \geq 2\epsilon d.$$

Therefore $\mathbf{w}_1^T\mathbf{w}_2 \geq 2\epsilon d$ implies that $i = j$, i.e., $\mathbf{w}_1$ and $\mathbf{w}_2$ originate from the same column of $\mathbf{A}$. Now assume as per the statement of the lemma that $|\text{supp}(\mathbf{w}_1)| > (1-2\epsilon)d$, $|\text{supp}(\mathbf{w}_2)| > (1-2\epsilon)d$ and $\epsilon \leq 1/6$, which implies that $(1-2\epsilon)d \geq 2\epsilon d$. Suppose $\mathbf{w}_1^T\mathbf{w}_2 < 2\epsilon d$, if $i = j$ then $\mathbf{w}_1$ and $\mathbf{w}_2$ must differ by less than $4\epsilon d$ nonzeros. This implies that

$$\mathbf{w}_1^T\mathbf{w}_2 > (1-4\epsilon)d \geq 2\epsilon d$$

which is a contradiction. Therefore, under the assumptions provided it follows that $\mathbf{w}_1^T\mathbf{w}_2 < 2\epsilon d$ implies $i \neq j$. Combining these results it follows that $\mathbf{w}_1$ and $\mathbf{w}_2$ originate from the same column of $\mathbf{A}$ iff $\mathbf{w}_1^T\mathbf{w}_2 \geq 2\epsilon d$ as claimed. □

As highlighted in the proof, $\mathbf{w}_1^T\mathbf{w}_2 \geq 2\epsilon d$ is sufficient to conclude that any two partial supports originate from the same column. However, without adding the additional assumptions on ϵ and on the size of the partial supports, we cannot conclude that $\mathbf{w}_1^T\mathbf{w}_2 < 2\epsilon d$ implies $\mathbf{w}_1$ and $\mathbf{w}_2$ do not originate from the same column. This limits us, for now, to clustering only fairly large partial supports, which have at least 2/3rds of the total nonzero entries of their respective columns of $\mathbf{A}$. Fortunately, under the same conditions as Lemma 2.2.4, Lemma 2.2.2 guarantees the existence of partial supports of this size. Note however, that if we have access to a complete column $\mathbf{a}_l$ of $\mathbf{A}$, then $\mathbf{w}_1^T\mathbf{a}_l \geq 2\epsilon d$ is sufficient to conclude that $\mathbf{w}_1$ originates from $\mathbf{a}_l$. This implies that it is possible to assign partial supports consisting of only around 1/3rd of the total nonzeros to a column of $\mathbf{A}$. For this reason we will later introduce the decode step, discussed in section 2.2.3, to try and match smaller partial supports to the columns of $\hat{\mathbf{A}}$ which have d nonzeros.

We are now in a position to define in detail the subroutines corresponding to steps 3, 4 and 5 of Algorithm 2. We emphasise before proceeding that the subroutines we will present are designed around the assumption that $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$. Consider the column vector $\mathbf{y} = \mathbf{A}\mathbf{x}$ and suppose that $\hat{\mathbf{A}}$ and $\hat{\mathbf{x}}$ are the current reconstructions of $\mathbf{A}$ and $\mathbf{x}$. Algorithm 3 processes the residual $\mathbf{r} = \mathbf{y} - \hat{\mathbf{A}}\hat{\mathbf{x}}$, taking as input arguments $\mathbf{r}$, the dimension m of $\mathbf{r}$, the expansion parameter ϵ, the column sparsity d, as well as the current reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{x}}$. The algorithm identifies singleton values and partial supports, either matching them with a nonzero column of $\hat{\mathbf{A}}$ or identifying them as belonging to a column of $\mathbf{A}$ not yet observed. To this end Algorithm 3 returns the unmatched singleton values and partial supports, stored as entries of the vector $\mathbf{q}$ and columns of the matrix $\mathbf{W}$ respectively, the number of unmatched partial supports p, the updated reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{x}}$ and a Boolean variable UPDATED, which indicates whether or not the reconstructions have in fact

been updated. The outer for loop and subsequent if statement, lines 5 and 6 of Algorithm 3 respectively, iterate through each nonzero entry of the input vector $\mathbf{r}$ to check for singleton values: this is performed by the inner for loop, starting on line 10, which identifies the locations in $\mathbf{r}$ where the current entry being checked appears. The variable c counts the number of appearances of the current entry; if $c > (1-2\epsilon)d$ then it is presumed that the current entry is a singleton value. Subsequently, on lines 18 and 19 the column $\hat{\mathbf{a}}_\kappa$ of $\hat{\mathbf{A}}$ whose support overlaps most with the associated candidate partial support is identified. Considering line 20, then if the overlap is sufficiently large, i.e., at least $2\epsilon d$ as per Lemma 2.2.4, then the partial support is assumed to originate from the identified column $\hat{\mathbf{a}}_\kappa$ and the reconstructions are updated accordingly, as per lines 21 and 22. Note here that $\sigma : \mathbb{R} \to \mathbb{R}$ denotes the unit threshold function, $\sigma(x) = 1$ for all $x \geq 1$ and is 0 otherwise. Note also, guided by Lemma 2.2.4, that Algorithm 3 only attempts to match partial supports with cardinality larger than $(1-2\epsilon)d$ to a column of $\hat{\mathbf{A}}$, as $\hat{\mathbf{A}}$ may contain incomplete columns. Indeed, we can only guarantee that a nonzero column of $\hat{\mathbf{A}}$ has more than $(1-2\epsilon)d$ nonzeros. If the overlap is not sufficient, then again guided by Lemma 2.2.4 we may conclude that the partial support under consideration does not match any existing nonzero column of the reconstruction. As a result, on lines 25 and 26 the current entry of the residual and its associated candidate partial support are added to $\mathbf{q}$ and $\mathbf{W}$ respectively. On line 27 the variable p, used to count the number of unmatched singleton values extracted from $\mathbf{r}$, is updated accordingly. If $c \leq (1-2\epsilon)d$ then the for loop starting on line 30 checks if the entry under consideration corresponds to a previously identified singleton value, now stored in $\hat{\mathbf{x}}$. If so, then on line 32 the partial support associated with the current entry is used to update the appropriate column of $\hat{\mathbf{A}}$, regardless of its size.

In regard to the computational complexity of Algorithm 3, if the sparsity of $\mathbf{r}$, $\hat{\mathbf{A}}$ and $\hat{\mathbf{x}}$ is not taken advantage of then the for loops starting on lines 5 and 10 iterate through $\mathcal{O}(m)$ entries, and the for loop starting on line 30 through $\mathcal{O}(n)$ entries. As the latter two for loops are nested inside the first, and $m = \mathcal{O}(n)$, then this corresponds to $\mathcal{O}(mn)$ for loop iterations. Based on the matrix vector product on line 18, each iteration has a cost of $\mathcal{O}(mn)$, therefore the total cost of Algorithm 3 is $\mathcal{O}(m^2n^2)$. If the sparsity of $\mathbf{r}$, $\hat{\mathbf{A}}$ and $\hat{\mathbf{x}}$ is taken advantage of, then as $\mathbf{r}$ has at most kd nonzero entries and d is constant, the for loops starting on lines 5, 10 and 30 each iterate through $\mathcal{O}(k)$ nonzero entries. Again, as the latter two for loops are nested inside the first this corresponds to $\mathcal{O}(k^2)$ for loop iterations. Based on the sparse

Algorithm 3 EXTRACT&MATCH($\mathbf{r}, m, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{x}}$)

$\mathbf{q} \leftarrow [\,]$
$\mathbf{W} \leftarrow [\,]$
$p \leftarrow 0$
UPDATED$\leftarrow$FALSE
for $i = 1 : m$ **do**
  if $r_i \neq 0$ **then**
    $c \leftarrow 1$
    $\mathbf{w} \leftarrow \text{zeros}(m)$
    $w_i \leftarrow 1$
    for $j = (i+1) : m$ **do**
      if $r_i = r_j$ **then**
        $w_j \leftarrow 1$
        $r_j \leftarrow 0$
        $c \leftarrow c + 1$
      end if
    end for
    if $c > (1 - 2\epsilon)d$ **then**
      $\mathbf{t} \leftarrow \hat{\mathbf{A}}^T \mathbf{w}$
      $\kappa \leftarrow \text{argmax}_{k \in [n]} \{t_k\}$
      if $t_\kappa \geq 2\epsilon d$ **then**
        $\hat{x}_\kappa \leftarrow r_i$
        $\hat{\mathbf{a}}_\kappa \leftarrow \sigma(\hat{\mathbf{a}}_\kappa + \mathbf{w})$
        UPDATED$\leftarrow$TRUE
      else
        $\mathbf{q} \leftarrow [\mathbf{q}^T; r_i]$
        $\mathbf{W} \leftarrow [\mathbf{W}; \mathbf{w}]$
        $p \leftarrow p + 1$
      end if
    else
      for $l \in \text{supp}(\hat{\mathbf{x}})$ **do**
        if $r_i = \hat{x}_l$ **then**
          $\hat{\mathbf{a}}_l \leftarrow \sigma(\hat{\mathbf{a}}_l + \mathbf{w})$
          UPDATED$\leftarrow$TRUE
        end if
      end for
    end if
  end if
end for
Return $\mathbf{q}, \mathbf{W}, p, \hat{\mathbf{A}}, \hat{\mathbf{x}}$, UPDATED

matrix vector product on line 18, each iteration has a cost of $\mathcal{O}(n)$, therefore when sparsity is leveraged the total cost of Algorithm 3 is $\mathcal{O}(k^2 n)$.

Lemma 2.2.5 (**Accuracy guarantees for Algorithm 3**). *Consider a vector* $\mathbf{y} = \mathbf{A}\mathbf{x}$*, where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *with* $\epsilon \leq 1/6$ *and* $\mathbf{x} \in \chi_k^n$*. Let* $\hat{\mathbf{A}}^{(1)} \in \{0,1\}^{m\times n}$ *and* $\hat{\mathbf{x}}^{(1)} \in \chi_k^n$ *be such that a column* $\hat{\mathbf{a}}_l^{(1)}$ *of* $\hat{\mathbf{A}}^{(1)}$ *is nonzero iff* $\hat{x}_l^{(1)} \neq 0$*, and if it is nonzero then* $|\operatorname{supp}(\hat{\mathbf{a}}_l^{(1)})| > (1-2\epsilon)d$*. In addition, assume that there exists a permutation matrix* $\mathbf{P} \in \mathcal{P}^{n\times n}$ *such that* $\operatorname{supp}(\hat{\mathbf{A}}^{(1)}\mathbf{P}^T) \subseteq \operatorname{supp}(\mathbf{A})$*,* $\operatorname{supp}(\mathbf{P}\hat{\mathbf{x}}^{(1)}) \subseteq \operatorname{supp}(\mathbf{x})$ *and* $\hat{x}_{P(l)}^{(1)} = x_{P(l)}$ *for all* $l \in \operatorname{supp}(\hat{\mathbf{x}}^{(1)})$*, where* $P : [n] \to [n]$ *denotes the row permutation caused by pre-multiplication with* $\mathbf{P}$*. Define the residual* $\mathbf{r} := \mathbf{y} - \hat{\mathbf{A}}^{(1)}\hat{\mathbf{x}}^{(1)}$ *and consider the variables returned by Algorithm 3,*

$$\mathbf{q}, \mathbf{W}, p, \hat{\mathbf{A}}^{(2)}, \hat{\mathbf{x}}^{(2)} \leftarrow \text{EXTRACT\&MATCH}(\mathbf{r}, m, d, \epsilon, \hat{\mathbf{A}}^{(1)}, \hat{\mathbf{x}}^{(1)}).$$

Then the following hold.

1. *If* $p \geq 1$ *then for each* $i \in [p]$ *there exists a unique* $l \in \operatorname{supp}(\mathbf{x})$ *such that* $q_i = x_l$ *and* $\operatorname{supp}(\mathbf{w}_i) \subseteq \operatorname{supp}(\mathbf{a}_l)$ *with* $|\operatorname{supp}(\mathbf{w}_i)| > (1-2\epsilon)d$*. In addition,* $\mathbf{a}_l^T \hat{\mathbf{a}}_h^{(1)} < 2\epsilon d$ *for all* $h \in [n]$*.*

2. *The updated reconstructions satisfy the following: a column* $\hat{\mathbf{a}}_l^{(2)}$ *of* $\hat{\mathbf{A}}^{(2)}$ *is nonzero iff* $\hat{x}_l^{(2)} \neq 0$*,* $\operatorname{supp}(\hat{\mathbf{A}}^{(1)}\mathbf{P}^T) \subseteq \operatorname{supp}(\hat{\mathbf{A}}^{(2)}\mathbf{P}^T) \subseteq \operatorname{supp}(\mathbf{A})$*,* $\operatorname{supp}(\mathbf{P}\hat{\mathbf{x}}^{(1)}) \subseteq \operatorname{supp}(\mathbf{P}\hat{\mathbf{x}}^{(2)}) \subseteq \operatorname{supp}(\mathbf{x})$ *and* $\hat{x}_{P(l)}^{(2)} = x_{P(l)}$ *for all* $l \in \operatorname{supp}(\hat{\mathbf{x}}^{(2)})$*.*

Proof. We will prove both statements by analysing the updates to the reconstructions, which take place on lines 21, 22, 25, 26 and 32. To this end, we first prove that the variables r_i and $\mathbf{w}$ in Algorithm 2.2.5 are a singleton value and partial support respectively if $c > (1-2\epsilon)d$. We then show that when $\mathbf{w}$ and a nonzero column of $\hat{\mathbf{A}}^{(1)}$ are partial supports originating from the same column in $\mathbf{A}$, then $\mathbf{w}$ is correctly used to update this column of the reconstruction and r_i the corresponding row of $\hat{\mathbf{x}}^{(1)}$. In the case where $\mathbf{w}$ originates from a different column of $\mathbf{A}$ to any of the nonzero columns of $\hat{\mathbf{A}}^{(1)}$, we show that $\mathbf{w}$ and r_i are correctly added to $\mathbf{W}$ and $\mathbf{q}$ respectively.

By the conditions of the lemma the residual $\mathbf{R}$ satisfies the conditions of Lemma 2.2.3. Therefore, and as $\epsilon \leq 1/6$, if on line 17 $c = f(r_i, \mathbf{r}) > (1-2\epsilon)d \geq 2\epsilon d$ then there exists an $l \in \operatorname{supp}(\mathbf{x})$ such that $r_i = x_l$. Furthermore, with $\mathbf{w} = g(r_i, \mathbf{r})$ then $\operatorname{supp}(\mathbf{w}) \subseteq \operatorname{supp}(\mathbf{a}_l)$ and $|\operatorname{supp}(\mathbf{w})| = f(r_i, \mathbf{r}) > (1-2\epsilon)d$. We conclude then that if $c > (1-2\epsilon)d$ then r_i and $\mathbf{w}$ are a singleton and partial support respectively. All that remains to be shown is that the partial support $\mathbf{w}$ is either correctly matched to an existing nonzero column of $\hat{\mathbf{A}}^{(1)}$, or it is correctly identified as originating from a

column of $\mathbf{A}$ for which there is not currently a partial or full reconstruction. We now assume that $\mathbf{w}$ is a partial support of $\mathbf{a}_l$ and examine each of these two cases in turn.

Suppose there exists a nonzero column $\hat{\mathbf{a}}_h^{(1)}$ of $\hat{\mathbf{A}}^{(1)}$ which is a partial support originating from $\mathbf{a}_l$, i.e., $\text{supp}(\hat{\mathbf{a}}_h^{(1)}) \subseteq \text{supp}(\mathbf{a}_l)$. As $|\,\text{supp}(\hat{\mathbf{a}}_h^{(1)})| > (1-2\epsilon)d \geq 2\epsilon d$ then by Lemma 2.2.1 it follows that $\text{supp}(\hat{\mathbf{a}}_h^{(1)}) \nsubseteq \text{supp}(\mathbf{a}_k)$ for all $k \neq h$. From Lemma 2.2.4 then $\mathbf{w}^T\hat{\mathbf{a}}_h^{(1)} \geq 2\epsilon d$ and $\mathbf{w}^T\hat{\mathbf{a}}_k^{(1)} < 2\epsilon d$ for all $k \neq l$. As a result, on line 19 it must follow that $\kappa = h$ and on line 20 that $t_\kappa \geq 2\epsilon d$. Therefore, the partial support $\mathbf{w}$ is identified as matching with the reconstruction of $\mathbf{a}_l$ in $\hat{\mathbf{A}}^{(1)}$, and $\hat{x}_l$ and $\hat{\mathbf{a}}_h$ are then correctly updated on lines 21 and 22 accordingly. Consider now the other case, in which there does not exist a nonzero column $\hat{\mathbf{a}}_h^{(1)}$ of $\hat{\mathbf{A}}^{(1)}$ which is a partial support originating from $\mathbf{a}_l$. Therefore, by Lemma 2.2.4, for any $h \in [n]$ then $\mathbf{w}^T\hat{\mathbf{a}}_h^{(1)} < 2\epsilon d$ and so the condition on line 20 is not satisfied. As a result, the singleton value r_i and partial support $\mathbf{w}$ are correctly used to update $\mathbf{q}$ and $\mathbf{W}$ on lines 25 and 26, and as a consequence statement 1 of the lemma immediately follows. For Statement 2, in addition observe that line 32 introduces no erroneous nonzeros as $\mathbf{x}$ is dissociated. Note also that as updates only occur to the existing nonzero columns of $\hat{\mathbf{A}}^{(1)}$, the sets of column indices corresponding to the nonzero columns of $\hat{\mathbf{A}}^{(1)}$ and $\hat{\mathbf{A}}^{(2)}$ are equal. Given these observation Statement 2 is must also be true. □

We now turn our attention to clustering the unmatched partial supports and singleton values extracted across all columns of the residual $\mathbf{R} = \mathbf{Y} - \hat{\mathbf{A}}\hat{\mathbf{X}}$. This step of D-EBF is performed by Algorithm 4, which takes as inputs the unmatched singleton values, stored in the entries of the vector $\mathbf{q} \in \mathbb{R}^p$, the unmatched partial supports stored in the columns of the matrix $\mathbf{W} \in \{0,1\}^{m\times p}$, ordered so that $\mathbf{w}_i$ is the partial support associated with the singleton q_i, the number of unmatched singleton values p, the smallest zero column index $\eta := \min_{l\in[n]}\{\hat{\mathbf{a}}_l = \mathbf{0}_m\}$, the vector $\mathbf{h} \in [N]^p$ which stores the indices of the columns of $\mathbf{R}$ from which each singular value was extracted, i.e., h_i is the column index of $\mathbf{R}$ from which q_i was extracted, the expansion parameter ϵ and column sparsity d of $\mathbf{A}$, and the current reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$. The p dimensional binary vector $\mathbf{b}$, initialised as the zero vector on line 1 of Algorithm 4, is used to indicate whether or not a partial support has been assigned to a column of $\hat{\mathbf{A}}$, one indicating true and zero false. The entries of the gram matrix $\mathbf{G}$, calculated on line 2, are used to cluster the unmatched partial supports. The outer for loop, starting on line 3, first checks on line 4 if the current partial support under consideration has been used to update a column of the reconstruction already. If it hasn't, then on line 5 it is used as the basis for a new nonzero column $\hat{\mathbf{a}}_\eta$ of $\hat{\mathbf{A}}$. On line 6 the corresponding singleton value is then used to update the appropriate

entry in $\hat{\mathbf{X}}$. The inner for loop, starting on line 8, checks the inner product of the partial support in question with all other partial supports that have not already been processed, as per the if statement on line 9. On line 10, any partial support that overlaps the new nonzero column $\hat{\mathbf{a}}_\eta$ sufficiently, as per Lemma 2.2.4, and which has not yet been assigned to a column, is then used to update $\hat{\mathbf{a}}_\eta$. On line 11 the corresponding singleton values are used to update the appropriate entries in $\hat{\mathbf{X}}$. Once this process is complete. then on line 15 η is updated accordingly, with new nonzero columns being introduced to $\hat{\mathbf{A}}$ from left to right.

Algorithm 4 CLUSTER&ADD($\mathbf{q}, \mathbf{W}, p, \eta, \mathbf{h}, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{X}}$)

1: $\mathbf{b} \leftarrow \text{zeros}(p)$
2: $\mathbf{G} \leftarrow \mathbf{W}^T\mathbf{W}$
3: **for** $i = 1 : p$ **do**
4: **if** $b_i = 0$ **then**
5: $\hat{\mathbf{a}}_\eta \leftarrow \mathbf{w}_i$
6: $\hat{x}_{\eta,h_i} \leftarrow q_i$
7: $b_i \leftarrow 1$
8: **for** $j = (i+1) : p$ **do**
9: **if** $g_{i,j} \geq 2\epsilon d$ **and** $b_j = 0$ **then**
10: $\hat{\mathbf{a}}_\eta \leftarrow \sigma(\hat{\mathbf{a}}_\eta + \mathbf{w}_j)$
11: $\hat{x}_{\eta,h_j} \leftarrow q_j$
12: $b_j \leftarrow 1$
13: **end if**
14: **end for**
15: $\eta \leftarrow \eta + 1$
16: **end if**
17: **end for**
18: Return $\hat{\mathbf{A}}, \hat{\mathbf{X}}, \eta$

The main expense of Algorithm 4 is the computation of the inner products between different partial supports, equivalent to the matrix product on line 2. As $p = \mathcal{O}(kN)$ then if the sparsity of $\mathbf{W}$ is not leveraged this matrix product costs $\mathcal{O}(k^2mN^2)$. If $\mathbf{W}$ is stored using a sparse format, as would be natural, then as each column of $\mathbf{W}$ is d sparse with d fixed then the cost can be reduced to $\mathcal{O}(k^2N^2)$.

2.2.3 Using a decoder algorithm to improve recovery

While the nonzeros of $\mathbf{A}$ can be recovered from potentially many of the columns of $\mathbf{R}$, each nonzero of $\mathbf{X}$ can be recovered only from one column. This makes the recovery up to row permutation of $\mathbf{X}$ more challenging than than the recovery up

Algorithm 5 DECODE($\mathbf{y}, m, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{x}}$)

$\mathcal{S} \leftarrow \{l \in [n] : |\operatorname{supp}(\mathbf{a}_l)| = d\}$
UPDATED←FALSE
RUN←TRUE
while RUN = TRUE **do**
  RUN←FALSE
  $\mathbf{r} \leftarrow \mathbf{y} - \hat{\mathbf{A}}\hat{\mathbf{x}}$
  if $\mathbf{r} \neq \mathbf{0}_m$ **then**
    for $i = 1 : m$ **do**
      if $r_i \neq 0$ **then**
        $c \leftarrow 1$
        $\mathbf{w} \leftarrow \text{zeros}(m)$
        $w_i \leftarrow 1$
        for $j = (i+1) : m$ **do**
          if $r_i = r_j$ **then**
            $w_j \leftarrow 1$
            $r_j \leftarrow 0$
            $c \leftarrow c + 1$
          end if
        end for
        if $c \geq 2\epsilon d$ **then**
          $\mathbf{t} \leftarrow \hat{\mathbf{A}}_{\mathcal{S}}^T \mathbf{w}$
          $\kappa \leftarrow \text{argmax}_{k \in \mathcal{S}}\{t_k\}$
          if $t_\kappa \geq 2\epsilon d$ **then**
            $\hat{x}_\kappa \leftarrow r_i$
            UPDATED←TRUE
            RUN←TRUE
          end if
        end if
      end if
    end for
  end if
end while
Return $\hat{\mathbf{x}}$, UPDATED

to column permutation of $\mathbf{A}$. As highlighted in the discussion of Algorithm 3 in Section 2.2.2, due to Lemma 2.2.4 partial supports which are at most $(1-2\epsilon)d$ are not clustered or matched with a column of $\hat{\mathbf{A}}$. This is because the updates of prior iterations only guarantee that each nonzero column of the reconstruction has more than $(1-2\epsilon)d$ nonzeros. However, if we know that $\hat{\mathbf{a}}_l$ is complete, i.e. there exists a $h \in [n]$ such that $\hat{\mathbf{a}}_l = \mathbf{a}_h$, then for any partial support $\mathbf{w}$ it suffices that $\hat{\mathbf{a}}_l^T \mathbf{w} \geq 2\epsilon d$ to conclude that $\mathbf{w}$ originates from $\hat{\mathbf{a}}_l$. Therefore it is possible to match smaller partial supports, i.e., those of size around $d/3$ rather than $2d/3$, to columns of the reconstruction which are complete. To this end, while Algorithm 3 focuses on clustering partial supports with cardinality larger than $(1-2\epsilon)d$ and matching them to a nonzero column of $\hat{\mathbf{A}}$, the decode step, detailed in Algorithm 5, focuses on matching any partial support whose cardinality is at least $2\epsilon d$ to a complete column of $\hat{\mathbf{A}}$. More generally, the decode step should be considered a placeholder for one of the many algorithms from the combinatorial compressed sensing literature, aiming to at least partially decode $\hat{\mathbf{X}}$ given $\mathbf{R}$ and the complete columns of the encoder reconstruction $\hat{\mathbf{A}}_{\mathcal{S}}$. Examples of such algorithms include Sparse Matching Pursuit (SMP) [15], Sequential Sparse Matching Pursuit (SSMP) [16], Left Degree Dependent Signal Recovery (LDDSR) [123], Expander Recovery (ER) [70] and ℓ_0-decode [88, Algorithms 1 & 2]. In particular, [88, Algorithms 1 & 2] are particularly appropriate for deployment in this context as they are based on a model in which the columns of $\mathbf{X}$ also satisfy the dissociated property, and, as can be observed in [88], are able to recover $\mathbf{X}$ from $\mathbf{Y}$ given $\mathbf{A}$ even when k is large, i.e., $k \approx m/3$.

For clarity, and to allow for the reuse of certain theoretical tools, in our analysis we will consider the decoder subroutine detailed in Algorithm 5, which is inspired by Expander Recovery (ER) [70]. We again emphasise that other decoding algorithms could be deployed instead, depending on the specific objectives and circumstances, see in particular [88, Table 2]. Algorithm 5 attempts to decode a column vector $\mathbf{y} = \mathbf{A}\mathbf{x}$, taking as inputs $\mathbf{y}$ and its dimension m, the expansion parameter ϵ and column sparsity d of $\mathbf{A}$, and the corresponding reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{x}}$ of $\mathbf{A}$ and $\mathbf{x}$. The algorithm returns an updated reconstruction of the sparse code $\hat{\mathbf{x}}$ and a Boolean variable UPDATED, which indicates whether or not an update to $\hat{\mathbf{x}}$ has actually occurred. Algorithm 5 operates in a manner similar to Algorithm 3, the key difference being that the partial supports are compared with and matched to only the complete columns of $\hat{\mathbf{A}}$, which are identified on line 1. As in the case of Algorithm 3, the role of the nested for loops starting on lines 8 and 13 is to identify and extract singleton values and partial supports. However, as the columns of $\hat{\mathbf{A}}_{\mathcal{S}}$

are complete, then the required overlap in support as per line 20 is only $2\epsilon d$: this follows from the fact that any partial support $\mathbf{w}$, irrespective of its size, originates from a column $\mathbf{a}_l$ of $\mathbf{A}$ if $\mathbf{w}^T\mathbf{a}_l \geq 2\epsilon d$. The Boolean variable UPDATED is used to track whether or not at any point an update to $\hat{\mathbf{x}}$ has occurred. The Boolean variable RUN is used to determine whether the while loop running from lines 4 to 32 should iterate: this occurs as long as a new nonzero is added to $\hat{\mathbf{x}}$ on line 24. As a result, at each iterate prior to the termination iterate, then on line 6 the contribution of at least one column of $\hat{\mathbf{A}}_{\mathcal{S}}$ is removed from $\mathbf{y}$. This update to the residual potentially reveals new singleton values, which can be matched with another column of $\hat{\mathbf{A}}_{\mathcal{S}}$, allowing for further updates to $\hat{\mathbf{x}}$. If $\hat{\mathbf{x}}$ is not updated during the previous iterate, then the residual at the current and previous iterates will be the same. Therefore no progress can be made and so the algorithm terminates.

Without leveraging sparsity, the outer for loop starting on line 8 iterates through $\mathcal{O}(m)$ elements. Each iteration involves an inner for loop, starting on line 13, which also iterates through $\mathcal{O}(m)$ elements, and a matrix vector product on line 21. The cost of each of these is $\mathcal{O}(m)$ and $\mathcal{O}(mn)$ respectively, therefore overall the cost per iteration of the while loop is $\mathcal{O}(m^2n)$. As $\hat{\mathbf{x}}$ has $\mathcal{O}(k)$ nonzeros then the while loop starting on line 4 iterates $\mathcal{O}(k)$ times. Therefore, without taking advantage of sparsity, the total computational cost of Algorithm 5 is $\mathcal{O}(km^2n)$. However, if $\mathbf{r}$, $\hat{\mathbf{A}}_{\mathcal{S}}$ and $\hat{\mathbf{x}}$ are stored using a sparse format, then the per iteration cost of the while loop is $\mathcal{O}(kn)$, resulting in a reduced total cost of $\mathcal{O}(k^2n)$. Finally, under similar assumptions to Lemma 2.2.5, we provide the following accuracy guarantees for Algorithm 5.

Lemma 2.2.6 (**Accuracy of Algorithm 5**). *Consider a vector* $\mathbf{y} = \mathbf{A}\mathbf{x}$*, where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *with* $\epsilon \leq 1/6$ *and* $\mathbf{x} \in \chi_k^n$*. Let* $\hat{\mathbf{A}} \in \{0,1\}^{m\times n}$ *and* $\hat{\mathbf{x}}^{(1)} \in \chi_k^n$ *be such that a column* $\hat{\mathbf{a}}_l$ *of* $\hat{\mathbf{A}}$ *is nonzero iff* $\hat{x}_l^{(1)} \neq 0$*, and if it is nonzero then* $|\operatorname{supp}(\hat{\mathbf{a}}_l)| > (1-2\epsilon)d$*. In addition, assume there exists a permutation matrix* $\mathbf{P} \in \mathcal{P}^{n\times n}$ *such that* $\operatorname{supp}(\hat{\mathbf{A}}\mathbf{P}^T) \subseteq \operatorname{supp}(\mathbf{A})$*,* $\operatorname{supp}(\mathbf{P}\hat{\mathbf{x}}^{(1)}) \subseteq \operatorname{supp}(\mathbf{x})$ *and* $\hat{x}^{(1)}_{P(l)} = x_{P(l)}$ *for all* $l \in$ *supp*$(\hat{\mathbf{x}}^{(1)})$*, where* $P : [n] \to [n]$ *denotes the row permutation caused by pre-multiplication with* $\mathbf{P}$*. Consider the variables computed during and returned by DECODE, Algorithm 5,*

$$\hat{\mathbf{x}}^{(2)}, \text{UPDATED} \leftarrow \text{DECODE}(\mathbf{y}, m, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{x}}^{(1)}).$$

Let $\mathcal{Q}$ *denote the event that* $\operatorname{supp}(\mathbf{P}\hat{\mathbf{x}}) \subseteq \operatorname{supp}(\mathbf{x})$ *and* $\hat{x}_{P(l)} = x_{P(l)}$ *for all* $l \in \operatorname{supp}(\hat{\mathbf{x}})$*. Then the following hold.*

1. *At any point during the run-time of DECODE* $\mathcal{Q}$ *is true.*

2. *The sparse code reconstruction returned by DECODE satisfies* $\text{supp}(\mathbf{P}\hat{\mathbf{x}}^{(1)}) \subseteq \text{supp}(\mathbf{P}\hat{\mathbf{x}}^{(2)})$. *Furthermore, recalling that* $\mathcal{S} = \{l \in [n] : \hat{\mathbf{a}}_l = d\}$ *and defining* $P(\mathcal{S}) := \{h \in [n] : \exists l \in \mathcal{S} \text{ s.t. } P(l) = h\}$, *if* $\text{supp}(\mathbf{x}) \subseteq P(\mathcal{S})$ *then* $\mathbf{P}\hat{\mathbf{x}}^{(2)} = \mathbf{x}$.

3. *The while loop, lines 5-31, exits after at most* k *iterations.*

Proof. First we prove Statement 1 by induction. Clearly by the assumptions on $\hat{\mathbf{x}}^{(1)}$ then $\mathcal{Q}$ is true at the start of iteration 1. In addition, if $\mathcal{Q}$ is true at the end of an iteration it must be true at the beginning of the next iteration. To prove Statement 1 it therefore suffices to prove that if $\mathcal{Q}$ is true at the start, i.e., line 4, of iteration t, then it is also true at the end, i.e., line 31, of iteration t. Suppose then that $\mathcal{Q}$ is true at the start of iteration t, then the residual at the start of iteration t satisfies the conditions required for Lemma 2.2.3. As a result, if on line 20 it holds that $c = f(r_i, \mathbf{r}) \geq 2\epsilon d$, then there exists an $l \in \text{supp}(\mathbf{x})$ such that $r_i = x_l$, meaning r_i is a singleton value. Furthermore, with $\mathbf{w} = g(r_i, \mathbf{r})$ then $|\text{supp}(\mathbf{w})| \geq 2\epsilon d$ and there exists a unique $l \in [n]$ such that $\text{supp}(\mathbf{w}) \subseteq \text{supp}(\mathbf{a}_l)$. Similar to Lemma 2.2.5, we now explore the two possible cases: either there exists a unique $h \in \mathcal{S}$ such that $\hat{\mathbf{a}}_h = \mathbf{a}_l$ or there does not. Suppose then that there exists a column $\hat{\mathbf{a}}_h$ such that $\hat{\mathbf{a}}_h = \mathbf{a}_l$. Therefore $\mathbf{w}^T\hat{\mathbf{a}}_h \geq 2\epsilon d$ and from Lemma 2.2.4 it follows that $\mathbf{w}^T\hat{\mathbf{a}}_k < 2\epsilon d$ for all $k \neq n$. Therefore on line 22 it must be true that $\kappa = h$. As a result the condition on line 23 is satisfied and so the partial support $\mathbf{w}$ is identified as matching with the complete reconstruction of $\mathbf{a}_l$ in $\hat{\mathbf{A}}$. Subsequently the corresponding entry in $\hat{\mathbf{x}}$ is correctly updated as per line 24. Suppose now for all $h \in \mathcal{S}$ it holds that $\hat{\mathbf{a}}_h \neq \mathbf{a}_l$. From Lemma 2.2.1 it therefore follows that $t_h = \mathbf{w}^T\hat{\mathbf{a}}_h \leq \mathbf{a}_l^T\mathbf{a}_{P(h)} < 2\epsilon d$ for all $h \in \mathcal{S}$. Therefore $t_\kappa < 2\epsilon d$ and hence the condition on line 23 is not satisfied. As a result, under this assumption $\mathbf{w}$ correctly does not match with any complete column of the reconstruction and so no update occurs to $\hat{\mathbf{x}}$. As $\hat{\mathbf{A}}$ is unchanged and only the row entries of $\hat{\mathbf{x}}$ with index in $\mathcal{S}$ are updated, then this implies that $\mathcal{Q}$ is true at the end of iteration t. This concludes the proof by induction of Statement 1.

The first part of Statement 2 follows immediately from Statement 1. For the second part of Statement 2, let $\mathbf{V} := \hat{\mathbf{A}}\mathbf{P}^T$ and $\mathbf{u} := \mathbf{P}\hat{\mathbf{x}}^{(1)}$. By construction $\text{supp}(\mathbf{V}) \subseteq \text{supp}(\mathbf{A})$ and $\text{supp}(\mathbf{u}) \subseteq \text{supp}(\mathbf{x})$. Therefore at any iteration of the while loop the residual calculated on line 6 can be expressed as

$$\mathbf{r} = \mathbf{y} - \hat{\mathbf{A}}\hat{\mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{V}\mathbf{u}.$$

For typographical ease let $\mathcal{C} := \text{supp}(\mathbf{x})$ and suppose that $\mathcal{C} \subseteq P(\mathcal{S})$, we proceed by contradiction and assume that Statement 2 is false. If Statement 2 is false then

this implies that the while loop of DECODE exits at some iteration with $\mathbf{r} \neq \mathbf{0}_m$. Therefore, and as Statement 1 is true, it must follow that $f(x_l, \mathbf{r}) < 2\epsilon d$ for all $l \in \mathcal{C}$. Due to our assumptions and the proof of Statement 1 then $\mathbf{V}_{P(\mathcal{S})} = \mathbf{A}_{P(\mathcal{S})}$, $\text{supp}(\mathbf{u}) \subseteq \mathcal{C}$ and $u_l = x_l$ for all $l \in \text{supp}(\mathbf{u})$. Therefore

$$\mathbf{r} = \mathbf{A}_{\mathcal{C}}\mathbf{x}_{\mathcal{C}} - \mathbf{V}_{\mathcal{C}}\mathbf{u}_{\mathcal{C}} = \mathbf{A}_{\mathcal{C}}\mathbf{z}$$

where $\mathbf{A}_{\mathcal{C}} \in \mathcal{E}_{k,\epsilon,d}^{m\times k}$ with $\epsilon \leq 1/6$ and $\mathbf{z} := \mathbf{x}_{\mathcal{C}} - \mathbf{u}_{\mathcal{C}} \in \chi_k^k$. As $\mathbf{A}_{\mathcal{C}} \in \mathcal{E}_{k,\epsilon,d}^{m\times k}$ with $\epsilon \leq 1/6$ and $\mathbf{z} \in \chi_k^k$, $\mathbf{z} \neq \mathbf{0}_k$, then Lemma 2.2.2 ensures the existence an $l \in \text{supp}(\mathbf{x}) \backslash \text{supp}(\mathbf{u})$ such that $f(x_l, \mathbf{r}) > (1 - 2\epsilon)d \geq 2\epsilon d$, which is a contradiction. Therefore Statement 2 must also be true.

Finally, to prove Statement 3, observe that at each iteration prior to the exit iteration the contribution of at least one column of $\hat{\mathbf{A}}_{\mathcal{S}}$ is removed from the residual $\mathbf{r}$. As $\mathbf{x}$ is k sparse then there are at most k columns contributing to $\mathbf{r}$, therefore the maximum number of possible iterates is k. □

2.2.4 Detailed summary of the D-EBF algorithm

Algorithm 6 provides a detailed version of Algorithm 2. The algorithm takes as inputs the product matrix $\mathbf{Y} \in \mathbb{R}^{m\times N}$, the dimensions m, n and N of $\mathbf{A}$ and $\mathbf{X}$, and the expansion parameter ϵ and column sparsity d of $\mathbf{A}$. The algorithm returns the estimates $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ of $\mathbf{A}$ and $\mathbf{X}$ respectively, aiming to recover them up to permutation. We emphasise again that the D-EBF algorithm implicitly assumes that $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$.

The while loop starting on line 6 runs until either $\hat{\mathbf{A}}\hat{\mathbf{X}} = \mathbf{Y}$, or no changes to the residual $\mathbf{R}$ of $\mathbf{Y}$ occur. The for loop starting on line 11 extracts partial supports and singleton values from each column of $\mathbf{R}$ as per Algorithm 3, storing the unmatched singleton values and partial supports in $\mathbf{q}$ and $\mathbf{W}$ respectively. On line 15 the vector $\mathbf{h}$ is used to keep track of each singleton value's column of origin, while on line 16 p is used to count the number of singleton values extracted across all columns of $\mathbf{R}$ at the current iteration. So long as at least one unmatched singleton value is found, then on line 19 the associated unmatched partial supports are clustered and used to construct new nonzero columns in $\hat{\mathbf{A}}$. Note that the nonzero columns of $\hat{\mathbf{A}}$ are added from left to right, with the variable η tracking the zero column of $\hat{\mathbf{A}}$ with the smallest index. To further grow the support of $\hat{\mathbf{X}}$, the for loop starting on line 22 applies Algorithm 5 to each of the columns of $\mathbf{Y}$. Alternatively, and as discussed in section 2.2.3, some other decoding algorithm from the combinatorial compressed sensing literature could be used instead. Finally, given the updates to $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$, the

residual is then recomputed on line 25 in preparation for singleton value and partial support extraction and clustering at the next iteration.

Algorithm 6 D-EBF($\mathbf{Y}, m, n, N, \epsilon, d$)

1: $\hat{\mathbf{A}} \leftarrow \text{zeros}(m, n)$
2: $\hat{\mathbf{X}} \leftarrow \text{zeros}(n, N)$
3: $\mathbf{R} \leftarrow \mathbf{Y}$
4: UPDATED $\leftarrow$ TRUE
5: $\eta \leftarrow 1$
6: **while** $\mathbf{R} \neq \mathbf{0}_{m \times N}$ **and** UPDATED=TRUE **do**
7: $\quad \mathbf{q} \leftarrow [\,]$
8: $\quad \mathbf{W} \leftarrow [\,]$
9: $\quad \mathbf{h} \leftarrow [\,]$
10: $\quad p \leftarrow 0$
11: $\quad$ **for** $i = 1 : N$ **do**
12: $\quad\quad \mathbf{u}, \mathbf{V}, c, \hat{\mathbf{A}}, \hat{\mathbf{x}}_i, \text{UPDATED} \leftarrow \text{EXTRACT\&MATCH}(\mathbf{r}_i, m, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{x}}_i)$
13: $\quad\quad \mathbf{q} \leftarrow [\mathbf{q}^T; \mathbf{u}^T]$
14: $\quad\quad \mathbf{W} \leftarrow [\mathbf{W}; \mathbf{V}]$
15: $\quad\quad \mathbf{h} \leftarrow [\mathbf{h}, i \times \text{zeros}(c)]$
16: $\quad\quad p \leftarrow p + c$
17: $\quad$ **end for**
18: $\quad$ **if** $p > 0$ **then**
19: $\quad\quad \hat{\mathbf{A}}, \hat{\mathbf{X}}, \eta \leftarrow \text{CLUSTER\&ADD}(\mathbf{q}, \mathbf{W}, p, \eta, \mathbf{h}, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{X}})$
20: $\quad\quad$ UPDATED$\leftarrow$TRUE
21: $\quad$ **end if**
22: $\quad$ **for** i=1:N **do**
23: $\quad\quad \hat{\mathbf{x}}_i, \text{UPDATED} \leftarrow \text{DECODE}(\mathbf{y}_i, m, N, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{x}}_i)$
24: $\quad$ **end for**
25: $\quad \mathbf{R} \leftarrow \mathbf{Y} - \hat{\mathbf{A}}\hat{\mathbf{X}}$
26: **end while**
27: Return $\hat{\mathbf{A}}, \hat{\mathbf{X}}$

In terms of computational cost the key contributors are lines 12, 19, 23 and 25. We will present the computational cost first without and then with taking advantage of sparsity. As per the discussion in section 2.2.2, the for loop running from line 11 to 17 has a cost of $\mathcal{O}(m^2n^2N)$ or $\mathcal{O}(k^2nN)$ respectively. Observe also that by transferring the matching process, lines 17-24 of Algorithm 3, to Algorithm 4, then singleton values and partial supports can be extracted from each column of $\mathbf{R}$ in parallel, using N processors with a computational cost of $\mathcal{O}(k^2n)$ per processor. Indeed, this step is trivially parallelizable across the columns of $\mathbf{R}$. Likewise, from the discussion in Section 2.2.2, line 19 has a cost of $\mathcal{O}(k^2mN^2)$ or $\mathcal{O}(k^2N^2)$ respectively. In comparison this step cannot be parallelized. Turning our attention to the decoding step of the

algorithm, then, as per the discussion in Section 2.2.3, the for loop running from line 22 to 24 has a cost of $\mathcal{O}(km^2nN)$ or $\mathcal{O}(k^2nN)$ respectively. This step is again trivially parallelizable across the columns of $\mathbf{Y}$. Finally, the matrix product on line 25 has a cost of $\mathcal{O}(mnN)$ or $\mathcal{O}(knN)$ respectively. Therefore, the while loop running from lines 6 to 26 of Algorithm 6 has a per iteration cost of $\mathcal{O}(m(mn^2+k^2N)N)$ if sparsity is not leveraged, or $\mathcal{O}(k^2(n+N)N)$ if it is. Under certain assumptions we are also able to provide accuracy guarantees for Algorithm 6, as detailed in Lemma 2.2.7.

Lemma 2.2.7 (**Accuracy of D-EBF**). *Let $\mathbf{Y} = \mathbf{AX}$, where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ and $\mathbf{X} \in \chi_k^{n\times N}$. Consider Algorithm 6, D-EBF: in regard to the reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ computed at any point during the run-time of D-EBF, we will say $\mathcal{Q}$ is true iff the following hold.*

(a) A column $\hat{\mathbf{a}}_l$ of $\hat{\mathbf{A}}$ is nonzero iff the lth row of $\hat{\mathbf{X}}$ is nonzero.

(b) For all $l \in [n]$ if $\hat{\mathbf{a}}_l$ is nonzero then $\operatorname{supp}(\hat{\mathbf{a}}_l) > (1-2\epsilon)d$.

(c) There exists a permutation matrix $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that $\operatorname{supp}(\hat{\mathbf{A}}\mathbf{P}^T) \subseteq \operatorname{supp}(\mathbf{A})$, $\operatorname{supp}(\mathbf{P}\hat{\mathbf{X}}) \subseteq \operatorname{supp}(\mathbf{X})$ and $\hat{x}_{P(l),h} = x_{P(l),h}$ for all $h \in [N]$ and $l \in supp(\hat{\mathbf{x}}_l)$, where $P : [n] \to [n]$ denotes the row permutation caused by pre-multiplication with $\mathbf{P}$.

Suppose that D-EBF exits after the completion of iteration $t_f \in \mathbb{N}$ of the while loop, then the following statements are true.

1. *At any point during the run time of D-EBF $\mathcal{Q}$ is true. Therefore the reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ returned by D-EBF also also satisfy the conditions for $\mathcal{Q}$ to be true.*

2. *For any iteration of the while loop $t < t_f$, let $\hat{\mathbf{A}}^{(1)}$ and $\hat{\mathbf{X}}^{(1)}$ denote the reconstructions at the start of the while loop, line 6, and $\hat{\mathbf{A}}^{(2)}$ and $\hat{\mathbf{X}}^{(2)}$ denote the reconstructions at the end of the while loop, line 26. Then either $\operatorname{supp}(\hat{\mathbf{A}}^{(1)}) \subset \operatorname{supp}(\hat{\mathbf{A}}^{(2)})$ and $\operatorname{supp}(\hat{\mathbf{X}}^{(1)}) \subseteq \operatorname{supp}(\hat{\mathbf{X}}^{(2)})$ or $\operatorname{supp}(\hat{\mathbf{A}}^{(1)}) \subseteq \operatorname{supp}(\hat{\mathbf{A}}^{(2)})$ and $\operatorname{supp}(\hat{\mathbf{X}}^{(1)}) \subset \operatorname{supp}(\hat{\mathbf{X}}^{(2)})$.*

3. *Consider the reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ returned by D-EBF and the associated residual $\mathbf{R} = \mathbf{Y} - \hat{\mathbf{A}}\hat{\mathbf{X}}$: there exists a permutation matrix $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$ and $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$ iff $\mathbf{R} = \boldsymbol{0}_{m\times N}$. As a result, $\mathbf{R} = \boldsymbol{0}_{m\times N}$ is a sufficient condition to ensure that $\mathbf{Y}$ has a unique, up to permutation, factorisation of the form $\mathbf{Y} = \mathbf{AX}$ where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ and $\mathbf{X} \in \chi_k^{n\times N}$.*

Proof. We first prove statement 1 by induction, showing that if at the start of an iteration of the while loop $\mathcal{Q}$ is true, then $\mathcal{Q}$ will also be true throughout and up to the end of this iteration. This implies that $\mathcal{Q}$ will also be true at the start of the next iteration, and therefore by induction we will be able to conclude that Statement 1 is true. At the start of the first iterate $\mathcal{Q}$ is trivially true as this corresponds to the initialisation $\hat{\mathbf{A}} = \mathbf{0}_{m\times n}$, $\hat{\mathbf{X}} = \mathbf{0}_{n\times N}$. Assume then at any iterate $t \in [t_f]$ that $\mathcal{Q}$ is true on line 6, this implies $\mathcal{Q}$ is also true at the start of EXTRACT&MATCH. This ensures that the conditions on $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ required by Lemma 2.2.5 are satisfied and therefore the reconstructions returned by EXTRACT&MATCH, which are passed to CLUSTER&ADD on line 20, are also such that $\mathcal{Q}$ is true. In addition, by Statement 1 of Lemma 2.2.5, then for all $i \in [p]$ it follows that q_i and $\mathbf{w}_i$ are a singleton value and partial support respectively, with $|\operatorname{supp}(\mathbf{w}_i)| > (1-2\epsilon)d$. Furthermore, these partial supports originate from columns of $\mathbf{A}$ for which there is not yet a corresponding nonzero partial or complete column in $\hat{\mathbf{A}}$. As a result, the only manner in which the reconstructions returned by CLUSTER&ADD on line 20 can result in $\mathcal{Q}$ being false is if there is a clustering error, i.e., there exists a pair of partial supports whose inner product is larger than $2\epsilon d$ despite originating from different columns. This however would contradict Lemma 2.2.4, therefore the reconstructions returned by CLUSTER&ADD, which are inputted into DECODE on line 23, ensure that $\mathcal{Q}$ is true. This in turn implies that the conditions of Lemma 2.2.6 are satisfied, and therefore, from Statement 1 of Lemma 2.2.6, it follows that the reconstructions returned after the completion of the for loop running on lines 22-24, are also such that $\mathcal{Q}$ is true. Therefore, if $\mathcal{Q}$ is true at the beginning of an iteration it is also true throughout said iteration and at its end. As this implies that $\mathcal{Q}$ is true at the beginning of the next iteration, and given that $\mathcal{Q}$ is true at the start of the first iteration, then by induction Statement 1 must be true.

Given that Statement 1 is true, and as by inspection at no point during the run-time of D-EBF are nonzeros removed from the reconstructions, then clearly $\operatorname{supp}(\hat{\mathbf{A}}^{(1)}) \subseteq \operatorname{supp}(\hat{\mathbf{A}}^{(2)})$ and $\operatorname{supp}(\hat{\mathbf{X}}^{(1)}) \subseteq \operatorname{supp}(\hat{\mathbf{X}}^{(2)})$. In addition, as it is assumed that $t < t_f$, then by the construction of D-EBF an update must occur to either $\hat{\mathbf{A}}$ or $\hat{\mathbf{X}}$ during the iteration, therefore Statement 2 must also be true.

Finally, to prove Statement 3, clearly if there exists a $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that $\mathbf{A} = \hat{\mathbf{A}}\mathbf{P}^T$ and $\mathbf{X} = \mathbf{P}\hat{\mathbf{X}}$ then $\mathbf{Y} = \mathbf{A}\mathbf{X} = \hat{\mathbf{A}}\mathbf{P}^T\mathbf{P}\hat{\mathbf{X}}$ and so $\mathbf{R} = \mathbf{0}_{m\times N}$. For the converse, as Statement 1 is true, then there exists a $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that $\mathbf{V} := \hat{\mathbf{A}}\mathbf{P}^T$ satisfies $\operatorname{supp}(\mathbf{V}) \subseteq \operatorname{supp}(\mathbf{A})$, $\mathbf{U} := \mathbf{P}\hat{\mathbf{X}}$ satisfies $\operatorname{supp}(\mathbf{U}) \subseteq \operatorname{supp}(\mathbf{X})$ and $u_{i,j} = x_{i,j}$ for all $(i,j) \in \mathit{supp}(\mathbf{U})$. If $\mathbf{R} = \mathbf{0}_{m\times N}$ then $\mathbf{Y} = \hat{\mathbf{A}}\hat{\mathbf{X}} = \hat{\mathbf{A}}\mathbf{P}^T\mathbf{P}\hat{\mathbf{X}} = \mathbf{V}\mathbf{U}$. As $\mathbf{Y} =$

$\mathbf{AX} = \mathbf{VU}$ and $\text{supp}(\mathbf{V}) \subseteq \text{supp}(\mathbf{A})$, $\text{supp}(\mathbf{U}) \subseteq \text{supp}(\mathbf{X})$ and $u_{i,j} = x_{i,j}$ for all $(i,j) \in supp(\mathbf{U})$, then $\mathbf{A} = \mathbf{V}$ and $\mathbf{X} = \mathbf{U}$. □

In short, the implication of Lemma 2.2.7 is that as long as $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$, then the reconstructions returned by D-EBF are accurate up to permutation, and if they are in addition complete up to permutation then the factorisation of this form is unique. Without further assumptions on $\mathbf{X}$ it is not possible to derive guarantees in respect to fully recovering the matrix factors up to permutation. For instance, if a row of $\mathbf{X}$ is a zero row vector then the corresponding column of $\mathbf{A}$ contributes to no columns of $\mathbf{Y}$ and therefore cannot be hoped to be recovered. In Section 4.2 we will derive such guarantees by assuming that Y is drawn from the PSB model, defined in Definition 2.1.3.

2.3 Theoretical guarantees for D-EBF under the PSB model

In this section we will derive Theorem 4.1.1. Analysing D-EBF directly is challenging, therefore our approach is instead to study a simpler, surrogate algorithm, which we use to lower bound the performance of D-EBF. To this end we introduce the Naive Decoder-Expander Based Factorisation Algorithm (ND-EBF). Although highly suboptimal from a computational perspective, and therefore clearly not recommended for deployment in practice, this algorithm allows us to analyse the parameter regime in which D-EBF is successful with high probability. The structure of this section is as follows: in Section 2.3.1 we present and describe the ND-EBF algorithm, provide accuracy guarantees analogous to Lemma 2.2.7 and connect ND-EBF with D-EBF. Then in Section 2.3.2 we use these results to prove Theorem 4.1.1.

2.3.1 Naive Decoder-Expander Based Factorisation (ND-EBF)

In order to define the ND-EBF algorithm we must introduce the subroutine detailed in Algorithm 7. This subroutine plays a role analogous to that of Algorithm 4 in D-EBF. The key difference between these two subroutines is that while Algorithm 4 attempts to cluster all partial supports in $\mathbf{W}$, and then use each of these clusters to reconstruct a column of $\mathbf{A}$, Algorithm 7 attempts to identify only the largest cluster of partial supports in $\mathbf{W}$, and then use this one cluster to reconstruct a single column of $\mathbf{A}$. Algorithm 7 takes as inputs the following: the unmatched singleton values, stored in the entries of the vector $\mathbf{q} \in \mathbb{R}^p$, the unmatched partial supports stored

in the columns of the matrix $\mathbf{W} \in \{0,1\}^{m \times p}$ and ordered so that $\mathbf{w}_i$ is the partial support associated with the singleton q_i, the number of unmatched singleton values p, the index $\eta := \min\{l \in [n] : \hat{\mathbf{a}}_l = \mathbf{0}_m\}$, the vector $\mathbf{h} \in [N]^p$ which stores the indices of the columns of $\mathbf{R}$ from which each singleton value was extracted, i.e., h_i is the column index of $\mathbf{R}$ from which q_i was extracted, the expansion parameter ϵ and column sparsity d of $\mathbf{A}$, and the current reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$. The algorithm returns updated reconstructions $\hat{\mathbf{X}}$ and $\hat{\mathbf{X}}$: as can be observed at most one new nonzero column of $\hat{\mathbf{A}}$ and row of $\hat{\mathbf{X}}$ are updated. The p dimensional binary vector

Algorithm 7 MAXCLUSTER&ADD($\mathbf{q}, \mathbf{W}, p, \eta, \mathbf{h}, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{X}}$)

```
1:  b ← zeros(p)
2:  G ← W^T W
3:  C* ← φ
4:  for i = 1 : p do
5:     if b_i = 0 then
6:        C ← {i}
7:        for j = (i + 1) : p do
8:           if g_{i,j} ≥ 2εd and b_j = 0 then
9:              C ← C ∪ {i}
10:          end if
11:       end for
12:       if |C| > |C*| then
13:          C* ← C
14:       end if
15:    end if
16: end for
17: for i ∈ C* do
18:    â_η ← σ(â_η + w_i)
19:    x̂_{η,h_i} ← q_i
20: end for
21: Return Â, X̂
```

b, initialised as the zero vector on line 1, is used to indicate whether or not a partial support has been assigned to a column of $\hat{\mathbf{A}}$, one indicating true and zero false. The entries of the gram matrix **G**, calculated on line 2, are used to cluster the unmatched partial supports. On line 3 the index set $\mathcal{C}^* \subseteq [p]$ is initialised as empty and is used to denote the cluster of partial supports in **W** with the largest cardinality identified so far. The outer for loop, starting on line 4, first checks, as per the if statement on line 5, if the partial support $\mathbf{w}_i$ currently under consideration has been used to update a column of the reconstruction already. If this is not the case, then on line 6

the index set $\mathcal{C} \subseteq [p]$, which is used to denote the set of partial supports in $\mathbf{W}$ which originate from the same column of $\mathbf{A}$ as $\mathbf{w}_i$, is initialised. The inner for loop, lines 7-11, then identifies and finds the indices of any such partial supports. On lines 12 and 13, if $|\mathcal{C}| > |\mathcal{C}^*|$ then a larger cluster than $\mathcal{C}^*$ has been identified and so $\mathcal{C}^*$ is updated accordingly. Finally, after the completion of the outer for loop then on lines 18 and 19, the partial supports and singleton values belonging to the largest cluster $\mathcal{C}^*$ are used to update just the ηth column of $\hat{\mathbf{A}}$ and row of $\hat{\mathbf{X}}$ respectively.

Algorithm 8 ND-EBF($\mathbf{Y}, m, n, N, d, \epsilon$)

1: $\hat{\mathbf{A}} \leftarrow \text{zeros}(m, n)$
2: $\hat{\mathbf{X}} \leftarrow \text{zeros}(n, N)$
3: $\mathbf{R} \leftarrow \mathbf{Y}$
4: $\eta \leftarrow 1$
5: RUN←TRUE
6: **while** $\eta \leq n$ **and** RUN=TRUE **do**
7: $\quad \mathbf{q} \leftarrow []$
8: $\quad \mathbf{W} \leftarrow []$
9: $\quad \mathbf{h} \leftarrow []$
10: $\quad p \leftarrow 0$
11: $\quad$ **for** $i = 1 : N$ **do**
12: $\quad\quad \mathbf{u}, \mathbf{V}, c, \_, \hat{\mathbf{x}}_i, \_ \leftarrow$ EXTRACT&MATCH$(\mathbf{r}_i, m, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{x}}_i)$
13: $\quad\quad \mathbf{q} \leftarrow [\mathbf{q}^T; \mathbf{u}^T]$
14: $\quad\quad \mathbf{W} \leftarrow [\mathbf{W}; \mathbf{V}]$
15: $\quad\quad \mathbf{h} \leftarrow [\mathbf{h}, i \times \text{zeros}(c)]$
16: $\quad\quad p \leftarrow p + c$
17: $\quad$ **end for**
18: $\quad$ **if** $p > 0$ **then**
19: $\quad\quad \hat{\mathbf{A}}, \hat{\mathbf{X}} \leftarrow$ MAXCLUSTER&ADD$(\mathbf{q}, \mathbf{W}, p, \eta, \mathbf{h}, \epsilon, d, \hat{\mathbf{A}}, \hat{\mathbf{X}})$
20: $\quad\quad$ **if** $|\text{supp}(\hat{\mathbf{a}}_\eta)| = d$ **then**
21: $\quad\quad\quad$ **for** $i = 1 : N$ **do**
22: $\quad\quad\quad\quad \hat{\mathbf{x}}_i, \_ \leftarrow$ DECODE$(\mathbf{y}_i, m, N, \epsilon, d, \hat{\mathbf{A}}_{\mathcal{S}}, \hat{\mathbf{x}}_i)$
23: $\quad\quad\quad$ **end for**
24: $\quad\quad\quad \mathbf{R} \leftarrow \mathbf{Y} - \hat{\mathbf{A}}\hat{\mathbf{X}}$
25: $\quad\quad\quad \eta \leftarrow \eta + 1$
26: $\quad\quad$ **else**
27: $\quad\quad\quad$ RUN←FALSE
28: $\quad\quad$ **end if**
29: $\quad$ **else**
30: $\quad\quad$ RUN←FALSE
31: $\quad$ **end if**
32: **end while**
33: Return $\hat{\mathbf{A}}$, $\hat{\mathbf{X}}$

The ND-EBF algorithm is presented and defined in Algorithm 6. As referenced to earlier, the way this algorithm operates is very similar to that of D-EBF, the key difference being that while D-EBF maximally utilises all partial supports and singleton values extracted at each iteration, ND-EBF seeks only to utilise those associated with the largest cluster. The extraction of singleton values and partial supports, lines 11-17, is done in exactly the same fashion as in D-EBF. If at least one singleton value and partial support are extracted, i.e., $p > 0$, then on line 19 Algorithm 7 is deployed in order to recover a single column of $\mathbf{A}$ from the largest cluster of partial supports in $\mathbf{W}$. If this reconstruction is not complete, meaning the condition on line 20 is not satisfied, then the algorithm terminates. In order to recover further entries of $\mathbf{X}$ and grow the support of $\hat{\mathbf{X}}$, then on line 22 Algorithm 5, or some other combinatorial decoding algorithm, is deployed. Given the updates to both $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$, the residual is then recomputed on line 24 in preparation for singleton value and partial support extraction and clustering at the next iteration. Analogous to Lemma 2.2.7 we provide the following accuracy guarantees for ND-EBF.

Lemma 2.3.1 (**Accuracy of ND-EBF**). *Let* $\mathbf{Y} = \mathbf{AX}$*, where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *with* $\epsilon \leq 1/6$ *and* $\mathbf{X} \in \chi_k^{n\times N}$*. Consider Algorithm 8, ND-EBF: in regard to the reconstructions* $\hat{\mathbf{A}}$ *and* $\hat{\mathbf{X}}$ *computed at any point during the run-time of ND-EBF, we will say, for* $\eta \in [n]$*, that* $\mathcal{Q}(\eta)$ *is true iff the following hold.*

(a) The column vector $\hat{\mathbf{a}}_l$ *and row vector* $\tilde{\hat{\mathbf{x}}}_l$ *are nonzero iff* $l < \eta$*.*

(b) If $l < \eta$ *then* $|\operatorname{supp}(\hat{\mathbf{a}}_l)| = d$*.*

(c) There exists a permutation matrix $\mathbf{P} \in \mathcal{P}^{n\times n}$ *such that* $\operatorname{supp}(\hat{\mathbf{A}}\mathbf{P}^T) \subseteq \operatorname{supp}(\mathbf{A})$*,* $\operatorname{supp}(\mathbf{P}\hat{\mathbf{X}}) \subseteq \operatorname{supp}(\mathbf{X})$ *and* $\hat{x}_{P(l),i} = x_{P(l),i}$ *for all* $i \in [N]$ *and* $l \in supp(\hat{\mathbf{x}}_l)$*, where* $P : [n] \to [n]$ *denotes the row permutation caused by pre-multiplication with* $\mathbf{P}$*.*

Suppose that ND-EBF exits after the completion of iteration $\eta_f \in [n]$ *of the while loop, then the following statements are true.*

1. *For all* $\eta \in [\eta_f]$*, then at the start of the* η*th iteration of the while loop* $\mathcal{Q}(\eta)$ *is true.*

2. *Consider the reconstructions* $\hat{\mathbf{A}}$ *and* $\hat{\mathbf{X}}$ *returned by ND-EBF:* $\hat{\mathbf{A}}$ *and* $\hat{\mathbf{X}}$ *always satisfy (c) and there exists a permutation matrix* $\mathbf{P} \in \mathcal{P}^{n\times n}$ *such that* $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$ *and* $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$ *iff* $\eta_f = n$ *and* $\mathcal{Q}(n+1)$ *is true.*

3. *Consider the reconstructions* $\hat{\mathbf{A}}$ *and* $\hat{\mathbf{X}}$ *returned by ND-EBF and the associated residual* $\mathbf{R} = \mathbf{Y} - \hat{\mathbf{A}}\hat{\mathbf{X}}$ *: there exists a permutation matrix* $\mathbf{P} \in \mathcal{P}^{n\times n}$ *such that* $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$ *and* $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$ *iff* $\mathbf{R} = \boldsymbol{0}_{m\times N}$*. As a result,* $\mathbf{R} = \boldsymbol{0}_{m\times N}$ *is a sufficient condition to ensure that* $\mathbf{Y}$ *has a unique, up to permutation, factorisation of the form* $\mathbf{Y} = \mathbf{AX}$*, where* $\mathbf{A} \in \mathcal{E}^{m\times n}_{k,\epsilon,d}$ *with* $\epsilon \leq 1/6$ *and* $\mathbf{X} \in \chi^{n\times N}_k$*.*

Proof. We will prove Statement 1 by induction: the base case $\mathcal{Q}(1)$ is trivially true at the start of the first iteration as this corresponds to the initialisation $\hat{\mathbf{A}}^{(0)} = \mathbf{0}_{m\times n}$ and $\hat{\mathbf{X}}^{(0)} = \mathbf{0}_{n\times N}$. It therefore suffices to prove for any $\eta \in [n_f - 1]$, that if $\mathcal{Q}(\eta)$ is true at the start of iteration η, then $\mathcal{Q}(\eta + 1)$ is true at the end of iteration η, as this in turn implies $\mathcal{Q}(\eta + 1)$ is true at the start of iteration $\eta + 1$. Assume then for arbitrary $\eta \in [\eta_f - 1]$ that $\mathcal{Q}(\eta)$ is true at the start of iteration η, it follows that the reconstructions $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ inputted into EXTRACT&MATCH satisfy the conditions required in Lemma 2.2.5. Therefore, by Lemma 2.2.5, for all $i \in [p]$ it follows that q_i and $\mathbf{w}_i$ are a singleton value and partial support respectively, $|\operatorname{supp}(\mathbf{w}_i)| > (1 - 2\epsilon)d$ and $\mathbf{w}_i$ originates from a column $\mathbf{a}_l \neq \hat{\mathbf{a}}_h$ for all $h \in [n]$. Note also that as $\operatorname{supp}(\hat{\mathbf{a}}_l) = d$ for any $l < \eta$ and $\hat{\mathbf{a}}_l = \mathbf{0}_m$ otherwise, updates to $\hat{\mathbf{X}}$ during EXTRACT&MATCH occur only on rows with a row index less than η. Furthermore, as per line 12 ND-EBF ignores updates to $\hat{\mathbf{A}}$ performed by EXTRACT&MATCH. Therefore $\mathcal{Q}(\eta)$ must still be true after the completion of EXTRACT&MATCH. Consider now the updates to the reconstructions performed by the MAXCLUSTER&ADD subroutine on line 19 of ND-EBF. This subroutine updates only the ηth column and row of $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ respectively. As $\eta < \eta_f$ then $p > 0$ and $|\operatorname{supp}(\hat{\mathbf{a}}_\eta)| = d$. Furthermore, the corresponding row of $\hat{\mathbf{X}}$ is also updated by MAXCLUSTER&ADD with $\hat{x}_{\eta,i} = q_i$ for all $i \in \mathcal{C}^*$. Therefore conditions (a) and (b) of $\mathcal{Q}(\eta + 1)$ must be true at the completion of MAXCLUSTER&ADD. Assume now that condition (c) is not satisfied by the reconstructions returned by MAXCLUSTER&ADD: then $\hat{\mathbf{a}}_\eta \neq \mathbf{a}_h$ for all $h \in [n]$ and as a result there must have occurred a clustering error. This implies in turn that there exists a pair of columns of $\mathbf{W}$ whose inner product is at least $2\epsilon d$ despite both columns being partial supports originating from different columns of $\mathbf{A}$. This however would contradict Lemma 2.2.4, therefore by contradiction condition (c) is satisfied and as a result we may conclude that at the exit of MAXCLUSTER&ADD $\mathcal{Q}(\eta + 1)$ is true. As a result, the reconstructions inputted to DECODE on line 22 are such that the conditions required for Lemma 2.2.6 are satisfied, hence by Lemma 2.2.6 it follows that $\operatorname{supp}(\mathbf{PX}) \subseteq \operatorname{supp}(\mathbf{X})$ and $\hat{x}_{P(l),i} = x_{P(l),i}$ for all $(l, i) \in \operatorname{supp}(\hat{\mathbf{X}})$. As DECODE only updates the rows of $\hat{\mathbf{X}}$ corresponding to the complete columns of

$\hat{\mathbf{A}}$, then at the end of iteration η of the while loop of ND-EBF it follows that $\mathcal{Q}(\eta+1)$ is true. This concludes the proof by induction of Statement 1.

Turning our attention to Statement 2 then to prove the reconstructions returned by ND-EBF always satisfy condition (c), observe from Statement 1 that $\mathcal{Q}(\eta_f)$ is true at the start of iteration η_f. Therefore, by the same arguments used in the proof of Statement 1, the reconstructions returned by MAXCLUSTER&ADD at the η_fth iteration satisfy condition (c). As η_f is the final iteration of the while loop of ND-EBF then by construction there are two possibilities: (i) $|\operatorname{supp}(\mathbf{a}_\eta)| < d$ or (ii) $|\operatorname{supp}(\mathbf{a}_\eta)| = d$ and $\eta_f = n$. In the case of (i), ND-EBF exits with no further updates to the reconstructions, therefore the reconstructions returned by ND-EBF must satisfy (c). In the case of (ii), then as the reconstructions satisfy condition (c) and $|\operatorname{supp}(\hat{\mathbf{a}_h})| = d$ for all $h \in [n]$, then there must also exist a permutation $\mathbf{P} \in \mathcal{P}^{n\times n}$ such $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$. It follows that the inputs to DECODE during the nth iteration of the while loop satisfy the conditions required for Lemma 2.2.6. As $\mathcal{S} = \{l \in [N] : |\operatorname{supp}(\mathbf{a}_l)| = d\} = [n]$, then $\operatorname{supp}(\mathbf{x}_i) \subseteq P(\mathcal{S})$ for all $i \in [N]$ and, due to Statement 2 of Lemma 2.2.6, it therefore follows that $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$. In summary, in either case the reconstructions returned by ND-EBF satisfy condition (c).

Concerning the second part of Statement 2, observe that if $\mathcal{Q}(n+1)$ is true then $\eta_f = n$. Otherwise, if $\eta_f < n$ then ND-EBF exits with $|\operatorname{supp}(\hat{\mathbf{a}}_h)| < d$ for all $\eta_f < h \leq n$, and so $\mathcal{Q}(n+1)$ cannot be true. Suppose there exists a permutation matrix $\mathbf{P} \in \mathcal{P}^{n\times n}$ such $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$ and $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$. By inspection it is clear that $\mathcal{Q}(n+1)$ must be true which in turn implies $\eta_f = n$. The converse follows by observing that $\eta_f = n$ and $\mathcal{Q}(n+1)$ being true imply that case (ii) must occur, which we have already shown implies the existence of a permutation $\mathbf{P} \in \mathcal{P}^{n\times n}$ such $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$ and $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$.

The proof of Statement 3 follows in the same manner as the proof of Statement 3 in Lemma 2.2.7. Clearly, if there exists a $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that $\mathbf{A} = \hat{\mathbf{A}}\mathbf{P}^T$ and $\mathbf{X} = \mathbf{P}\hat{\mathbf{X}}$ then $\mathbf{Y} = \hat{\mathbf{A}}\mathbf{P}^T\mathbf{P}\hat{\mathbf{X}} = \hat{\mathbf{A}}\hat{\mathbf{X}}$ and therefore $\mathbf{R} = \mathbf{Y} - \hat{\mathbf{A}}\hat{\mathbf{X}} = \mathbf{0}_{m\times N}$. For the converse, as Statement 2 is true then there exists a $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that $\mathbf{V} := \hat{\mathbf{A}}\mathbf{P}^T$ satisfies $\operatorname{supp}(\mathbf{V}) \subseteq \operatorname{supp}(\mathbf{A})$, $\mathbf{U} := \mathbf{P}\hat{\mathbf{X}}$ satisfies $\operatorname{supp}(\mathbf{U}) \subseteq \operatorname{supp}(\mathbf{X})$ and $u_{i,j} = x_{i,j}$ for all $(i,j) \in \mathit{supp}(\mathbf{U})$. As $\mathbf{AX} = \hat{\mathbf{A}}\mathbf{P}^T\mathbf{P}\hat{\mathbf{X}} = \mathbf{VU}$ and $\operatorname{supp}(\mathbf{V}) \subseteq \operatorname{supp}(\mathbf{A})$, $\operatorname{supp}(\mathbf{U}) \subseteq \operatorname{supp}(\mathbf{X})$ and $u_{i,j} = x_{i,j}$ for all $(i,j) \in \mathit{supp}(\mathbf{U})$, then $\mathbf{A} = \mathbf{V}$ and $\mathbf{X} = \mathbf{U}$. □

In short, the implication of Lemma 2.3.1 is that as long as $\mathbf{A} \in \mathcal{E}^{m\times n}_{k,\epsilon,d}$ with $\epsilon \leq 1/6$, then the reconstructions returned by ND-EBF are accurate up to permutation, and, if they are in addition complete up to permutation, then the factorisation of this form

is unique. The following corollary will allow us to focus our analysis on the recovery of $\mathbf{A}$.

Corollary 2.3.1.1 (Recovery of A up to permutation is sufficient to recover both factors up to permutation). *Let $\mathbf{Y} = \mathbf{AX}$, where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ and $\mathbf{X} \in \chi_k^{n\times N}$. Suppose that $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ are the reconstructions of $\mathbf{A}$ and $\mathbf{X}$ returned by either ND-EBF or D-EBF. If there there exists a $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$ then $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$.*

Proof. In the case of ND-EBF this result follows directly from Statement 2 of Lemma 2.3.1. For D-EBF, observe that the last update to $\hat{\mathbf{A}}$ occurs on line 19 during the final iteration of the while loop prior to the algorithm terminating. Therefore, if there exists a $\mathbf{P} \in \mathcal{P}^{n\times n}$ such that the returned reconstruction of $\mathbf{A}$ satisfies $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$, then from line 20 and onward it must likewise hold that $\hat{\mathbf{A}}\mathbf{P}^T = \mathbf{A}$. Therefore, at the input to DECODE on line 23 it follows that $\mathcal{S} = \{l \in [n] : \hat{\mathbf{a}}_l = d\} = [n]$ and as a result $\text{supp}(\mathbf{x}_i) \subset P(\mathcal{S})$ for all $i \in [N]$. In addition, due to Statement 1 of Lemma 2.2.7, it follows that the input reconstructions to DECODE on line 23 satisfy the conditions of Lemma 2.2.6. As a result, by Statement 2 of Lemma 2.2.6, the sparse code $\hat{\mathbf{X}}$ returned by DECODE satisfies $\mathbf{P}\hat{\mathbf{X}} = \mathbf{X}$. This concludes the proof of the corollary. □

To complete this section we provide Lemma 2.3.3: the key takeaway of this lemma is that if ND-EBF successfully recovers the matrix factors of $\mathbf{Y}$ up to permutation, then so to will D-EBF. To prove this result however we require the following result.

Lemma 2.3.2. *Let $\mathbf{y} = \mathbf{Ax}$ where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ and $\mathbf{x} \in \chi_k^n$. For $b \in \{1, 2\}$, let $\hat{\mathbf{A}}^{(b)}$ and $\hat{\mathbf{x}}^{(b)}$ be such that there exists a $\mathbf{P}_b \in \mathcal{P}^{n\times n}$ $\text{supp}(\hat{\mathbf{A}}^{(b)}\mathbf{P}_b^T) \subseteq \text{supp}(\mathbf{A})$ and $\hat{x}^{(b)}_{P_b(l)} = x_{P_b(l)}$ for all $l \in \text{supp}(\hat{\mathbf{x}}^{(b)})$, where $P_b : [n] \to [n]$ is the row permutation corresponding to pre-multiplication by $\mathbf{P}_b$. In addition, assume that $\text{supp}(\hat{\mathbf{A}}^{(1)}\mathbf{P}_1^T) \subseteq \text{supp}(\hat{\mathbf{A}}^{(2)}\mathbf{P}_2^T)$ and $\text{supp}(\mathbf{P}_1\hat{\mathbf{x}}^{(1)}) \subseteq \text{supp}(\mathbf{P}_2\hat{\mathbf{x}}^{(2)})$. Considering the sparse codes returned by DECODE,*

$$\hat{\mathbf{x}}^{(b)}, \_ \leftarrow \text{DECODE}(\mathbf{y}, m, \hat{\mathbf{A}}^{(b)}, \hat{\mathbf{x}}^{(b)}, \epsilon, d)$$

then $\text{supp}(\hat{\mathbf{x}}^{(1)}) \subseteq \text{supp}(\hat{\mathbf{x}}^{(2)})$.

Proof. For typographical ease let $\mathbf{V}^{(1)} = \hat{\mathbf{A}}^{(1)}\mathbf{P}_1^T$, $\mathbf{u}^{(1)} = \mathbf{P}_1\hat{\mathbf{x}}^{(1)}$, $\mathbf{V}^{(2)} = \hat{\mathbf{A}}^{(2)}\mathbf{P}_2^T$ and $\mathbf{u}^{(2)} = \mathbf{P}_2\hat{\mathbf{x}}^{(2)}$. Observe for $b \in \{1, 2\}$ that if

$$\mathbf{u}^{(b)}, \_ \leftarrow \text{DECODE}(\mathbf{y}, m, \mathbf{V}^{(b)}, \mathbf{u}^{(b)}, \epsilon, d) \tag{2.3}$$

then $\mathbf{u}^{(b)} = \mathbf{P}_b\hat{\mathbf{x}}^{(b)}$. Given equation 2.3 it therefore suffices to prove $\text{supp}(\mathbf{u}^{(1)}) \subseteq \text{supp}(\mathbf{u}^{(2)})$. Let $\mathcal{Q}(t)$ be true iff at the end of iteration t of the while loop, starting on line 4 of DECODE, it holds that $\text{supp}(\mathbf{u}^{(1)}) \subseteq \text{supp}(\mathbf{u}^{(2)})$. We proceed by induction: observe that $\mathcal{Q}(0)$, which we may interpret as the event that at the start of the first iteration $\text{supp}(\mathbf{u}^{(1)}) \subseteq \text{supp}(\mathbf{u}^{(2)})$, is trivially true by the assumptions of the lemma. Therefore it suffices to prove that if $\mathcal{Q}(t-1)$ is true then $\mathcal{Q}(t)$ is true.

Assume $\mathcal{Q}(t-1)$ is true, analysing the residual computed on line 6 of DECODE it follows that

$$\mathbf{r}^{(1)} = \mathbf{A}\mathbf{x} - \mathbf{V}^{(1)}\mathbf{u}^{(1)},$$
$$\mathbf{r}^{(2)} = \mathbf{A}\mathbf{x} - \mathbf{V}^{(2)}\mathbf{u}^{(2)}.$$

Define, as in the proof of Lemma 2.2.1, for $j \in [m]$ the sets $\Omega_j := \text{supp}(\tilde{\mathbf{a}}_j) \cap \text{supp}(\mathbf{x})$, $\Gamma_j^{(1)} := \text{supp}(\tilde{\mathbf{v}}_j^{(1)}) \cap \text{supp}(\mathbf{u}^{(1)})$ and $\Gamma_j^{(2)} := \text{supp}(\tilde{\mathbf{v}}_j^{(2)}) \cap \text{supp}(\mathbf{u}^{(2)})$. It follows for any entry $j \in [m]$ that

$$r_j^{(1)} = \sum_{l\in\Omega_j} x_l - \sum_{h\in\Gamma_j^{(1)}} x_h = \sum_{l\in\Omega_j\backslash\Gamma_j^{(1)}} x_l$$
$$r_j^{(2)} = \sum_{l\in\Omega_j} x_l - \sum_{h\in\Gamma_j^{(2)}} x_h = \sum_{l\in\Omega_j\backslash\Gamma_j^{(2)}} x_l$$

As $\Gamma_j^{(1)} \subseteq \Gamma_j^{(2)}$, then $\Omega_j\backslash\Gamma_j^{(2)} \subseteq \Omega_j\backslash\Gamma_j^{(1)}$. Therefore any entry $r_j^{(2)}$ is a sum over a subset of the nonzeros in $\mathbf{x}_i$ used in the sum of $r_{j,i}^{(1)}$. Consider any row index $\kappa \in [n]$ such that $u_\kappa^{(1)} = 0$ at the start of iteration t and $u_\kappa^{(1)} = x_\kappa$ at the end of iteration t. This implies that $f(u_\kappa, \mathbf{r}^{(1)}) \geq 2\epsilon d$ and by the construction of DECODE that $\mathbf{v}_\kappa^{(1)} = \mathbf{v}_\kappa^{(2)} = \mathbf{a}_\kappa$. If $u_\kappa^{(2)} = x_\kappa$ at the start of iteration t then then clearly $u_\kappa^{(2)} = x_\kappa$ at the end of iteration t. If $u_\kappa^{(2)} = 0$ at the start of iteration t, then $\kappa \in \Omega_j\backslash\Gamma^{(2)}$ for all $j \in \text{supp}(\mathbf{a}_\kappa)$. In addition, as $\Omega_j\backslash\Gamma^{(2)} \subseteq \Omega_j\backslash\Gamma^{(1)}$ then $f(u_\kappa, \mathbf{r}^{(2)}) \geq f(u_\kappa, \mathbf{r}^{(1)}) \geq 2\epsilon d$ and therefore line 24 of DECODE ensures $u_\kappa^{(2)} = x_\kappa$ at the end of iteration t. Therefore we have proved, assuming $\mathcal{Q}(t-1)$ is true, that if $u_\kappa^{(1)} = 0$ at the start of iteration t and $u_\kappa^{(1)} = x_\kappa$ at the end of iteration t, then $u_\kappa^{(2)} = x_\kappa$ at the end of iteration t. Therefore $\text{supp}(\mathbf{u}^{(1)}) \subseteq \text{supp}(\mathbf{u}^{(2)})$ at the end of iteration t and so $\mathcal{Q}(t)$ is true. Given that we have proved both the base case and the inductive step the proof is complete. □

We are now ready to present and prove Lemma 2.3.3

Lemma 2.3.3 (**If ND-EBF successfully computes the matrix factors up to permutation then so will D-EBF**). *Let* $\mathbf{Y} = \mathbf{A}\mathbf{X}$ *where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *with* $\epsilon \leq 1/6$ *and* $\mathbf{X} \in \chi_k^{n\times N}$. *Let* $\hat{\mathbf{A}}^{(1)}$ *and* $\hat{\mathbf{X}}^{(1)}$ *denote the reconstructions returned by ND-EBF and* $\hat{\mathbf{A}}^{(2)}$ *and* $\hat{\mathbf{X}}^{(2)}$ *the reconstructions returned by D-EBF. If there exists a* $\mathbf{P}_1 \in \mathcal{P}^{n\times n}$

such that $\hat{\mathbf{A}}^{(1)}\mathbf{P}_1^T = \mathbf{A}$ *and* $\mathbf{P}_1\hat{\mathbf{X}}^{(1)} = \mathbf{X}$*, then there exists a* $\mathbf{P}_2 \in \mathcal{P}^{n\times n}$ *such that* $\hat{\mathbf{A}}^{(2)}\mathbf{P}_2^T = \mathbf{A}$ *and* $\mathbf{P}_2\hat{\mathbf{X}}^{(2)} = \mathbf{X}$*.*

Proof. Before proceeding we emphasise certain points of concerning the notation used in this proof. In particular, in what follows we use $\hat{\mathbf{A}}^{(1)}$ and $\hat{\mathbf{X}}^{(1)}$ to denote the reconstructions computed any point during the run-time of ND-EBF. Likewise, we use $\hat{\mathbf{A}}^{(2)}$ and $\hat{\mathbf{X}}^{(2)}$ to denote the reconstructions computed at any point during the run-time of D-EBF. In addition, let $\ell(\eta)$, where $\ell : [n] \to [n]$, denote the column of $\mathbf{A}$ recovered by ND-EBF during iteration η, i.e., $\hat{\mathbf{a}}_\eta^{(1)} = \mathbf{a}_{\ell(\eta)}$. Furthermore, let $\tau(\eta)$, where $\tau : [n] \to [n]$, denote the index of the column reconstruction of $\mathbf{a}_{\ell(\eta)}$ in $\hat{\mathbf{A}}^{(2)}$. Note that as a result of Lemma 2.2.7, at any point during the run-time of D-EBF either $\hat{\mathbf{a}}_{\tau(\eta)}^{(2)} = \mathbf{0}_m$ or $\mathbf{a}_{\ell(\eta)}^T\hat{\mathbf{a}}_{\tau(\eta)}^{(2)} > (1-2\epsilon)d$. Finally, let $\mathcal{Q}(\eta)$ be true iff at the end of iteration η of the while loop of D-EBF $\hat{\mathbf{a}}_{\tau(l)}^{(2)} = \hat{\mathbf{a}}_l^{(1)}$ for all $l \in [\eta]$, and for any $l \in [\eta]$, $\kappa \in [N]$ such that $\hat{x}_{l,\kappa}^{(1)} \neq 0$, then $\hat{x}_{\tau(l),\kappa}^{(2)} = \hat{x}_{l,\kappa}^{(1)}$.

It suffices to prove that if ND-EBF recovers $\mathbf{a}_{\ell(\eta)}$ by the end of ηth iteration of its while loop, then D-EBF also recovers $\mathbf{a}_{\ell(\eta)}$ by the end of ηth iteration of its while loop. Indeed, if this is true then as ND-EBF recovers $\mathbf{A}$ up to permutation by assumption, then so to will D-EBF. The result claimed then follows from Corollary 2.3.1.1. Our task therefore is to prove for all $\eta \in [n]$ that $\mathcal{Q}(\eta)$ is true. We proceed by induction. Observe that $\mathcal{Q}(0)$ is trivially true as both algorithms are initialised with $\hat{\mathbf{A}}^{(1)} = \hat{\mathbf{A}}^{(2)} = \mathbf{0}_{m\times n}$ and $\hat{\mathbf{X}}^{(1)} = \hat{\mathbf{X}}^{(2)} = \mathbf{0}_{n\times N}$. It suffices then to show for $\eta \in [n]$ that $\mathcal{Q}(\eta-1)$ true implies $\mathcal{Q}(\eta)$ true. Under the assumption that $\mathcal{Q}(\eta-1)$ is true, we now analyse the residuals computed at the beginning of iteration η of the respective while loops of ND-EBF and D-EBF. From Statement 1 of Lemma 2.3.1, then there exists a $\mathbf{P}_1 \in \mathcal{P}^{n\times n}$ such that $\mathbf{V}^{(1)} := \hat{\mathbf{A}}^{(1)}\mathbf{P}_1^T$ satisfies $\operatorname{supp}(\mathbf{V}^{(1)}) \subseteq \operatorname{supp}(\mathbf{A})$, $\mathbf{U}^{(1)} := \mathbf{P}_1\hat{\mathbf{X}}^{(1)}$ satisfies $\operatorname{supp}(\mathbf{U}^{(1)}) \subseteq \operatorname{supp}(\mathbf{X})$ and $u_{i,j}^{(1)} = x_{i,j}$ for all $(i,j) \in \operatorname{supp}(\mathbf{U}^{(1)})$. Likewise, from Statement 1 of Lemma 2.2.7, there exists a $\mathbf{P}_2 \in \mathcal{P}^{n\times n}$ such that $\mathbf{V}^{(2)} := \hat{\mathbf{A}}^{(2)}\mathbf{P}_2^T$ satisfies $\operatorname{supp}(\mathbf{V}^{(2)}) \subseteq \operatorname{supp}(\mathbf{A})$, $\mathbf{U}^{(2)} := \mathbf{P}_2\hat{\mathbf{X}}^{(2)}$ satisfies $\operatorname{supp}(\mathbf{U}^{(2)}) \subseteq \operatorname{supp}(\mathbf{X})$ and $u_{i,j}^{(2)} = x_{i,j}$ for all $(i,j) \in \operatorname{supp}(\mathbf{U}^{(2)})$. From Lemma 2.2.6, for any $l \geq \eta$ then $\mathbf{v}_{P_1(l)}^1$ and $\tilde{\mathbf{u}}_{P_1(l)}^1$ are zero vectors, therefore as $\mathcal{Q}(\eta-1)$ is true $\operatorname{supp}(\mathbf{V}^{(1)}) \subseteq \operatorname{supp}(\mathbf{V}^{(2)})$ and $\operatorname{supp}(\mathbf{U}^{(1)}) \subseteq \operatorname{supp}(\mathbf{U}^{(2)})$. The respective residuals inputted to EXTRACT&MATCH for each algorithm can be written as

$$\mathbf{R}^{(1)} = \mathbf{A}\mathbf{X} - \mathbf{V}^{(1)}\mathbf{U}^{(1)},$$
$$\mathbf{R}^{(2)} = \mathbf{A}\mathbf{X} - \mathbf{V}^{(2)}\mathbf{U}^{(2)}.$$

Considering an arbitrary column index $i \in [N]$ of the residuals, define, as in the proof of Lemma 2.2.1, $\Omega_j := \operatorname{supp}(\tilde{\mathbf{a}}_j) \cap \operatorname{supp}(\mathbf{x}_i)$, $\Gamma_j^{(1)} := \operatorname{supp}(\tilde{\mathbf{v}}_j^{(1)}) \cap \operatorname{supp}(\mathbf{u}_i^{(1)})$ and

$\Gamma_j^{(2)} := \text{supp}(\tilde{\mathbf{v}}_j^{(2)}) \cap \text{supp}(\mathbf{u}_i^{(2)})$, observing that $\Gamma_j^{(1)} \subseteq \Gamma_j^{(2)}$. The entry $(j,i) \in [m] \times [N]$ in each of these residuals can therefore be written as

$$r_{j,i}^{(1)} = \sum_{l \in \Omega_j} x_l - \sum_{h \in \Gamma_j^{(1)}} x_h = \sum_{l \in \Omega_j \backslash \Gamma_j^{(1)}} x_l,$$
$$r_{j,i}^{(2)} = \sum_{l \in \Omega_j} x_l - \sum_{h \in \Gamma_j^{(2)}} x_h = \sum_{l \in \Omega_j \backslash \Gamma_j^{(2)}} x_l$$

As $\Gamma_j^{(1)} \subseteq \Gamma_j^{(2)}$, then $\Omega_j \backslash \Gamma_j^{(2)} \subseteq \Omega_j \backslash \Gamma_j^{(1)}$. Therefore, any entry $r_{j,i}^{(2)}$ is a sum over a subset of the nonzeros in $\mathbf{x}_i$ used in the sum of $r_{j,i}^{(1)}$.

Suppose that at the end of iteration η of the while loop of D-EBF it holds that $\hat{\mathbf{a}}_{\tau(\eta)}^{(2)} = \mathbf{a}_{\ell(\eta)}$. Letting $\mathcal{S}_1 := \{l \in [n] : \text{supp}(\hat{\mathbf{a}}_l^{(1)}) = d\}$ and $\mathcal{S}_2 := \{l \in [n] : \text{supp}(\hat{\mathbf{a}}_l^{(2)}) = d\}$, then, as we are also assuming that $\mathcal{Q}(\eta - 1)$ is true, it must follow that $\mathcal{S}_1 \subseteq \mathcal{S}_2$. As a result, by Lemma 2.3.2, if for some column index $\kappa \in [N]$ it holds that $\hat{x}_{\eta,\kappa}^{(1)} = x_{\ell(\eta),\kappa}$, then $\hat{x}_{\tau(\eta),\kappa}^{(2)} = x_{\ell(\eta),\kappa}$ and as a result $\mathcal{Q}(\eta)$ must be true. Therefore, to prove the inductive step it suffices to show that $\hat{\mathbf{a}}_{\tau(\eta)}^{(2)} = \mathbf{a}_{\ell(\eta)}$ at the end of iteration η.

To this end, consider then an arbitrary $\gamma \in \text{supp}(\mathbf{a}_{\ell(\eta)})$. By the construction of ND-EBF and the assumption that ND-EBF recovers $\mathbf{A}$ up to permutation, then $\hat{a}_{\gamma,\eta}^{(1)} = 0$ at the start of iteration η and $\hat{a}_{\gamma,\eta}^{(1)} = 1$ at the end of iteration η. Therefore, considering the residual at the start of iteration η, there must exist a $\kappa \in [N]$ such that $f(x_{\ell(\eta),\kappa}, \mathbf{r}_\kappa^{(1)}) > (1 - 2\epsilon)d$ and $r_{\gamma,\kappa}^{(1)} = x_{\ell(\eta),\kappa}$. Recall now that $r_{j,\kappa}^{(2)}$ is a sum over a subset of the nonzeros in $\mathbf{x}_\kappa$ used in the sum of $r_{j,\kappa}^{(1)}$, therefore either $r_{j,\kappa}^{(2)} = 0$ or $r_{j,\kappa}^{(2)} = x_{\ell(\eta),\kappa}$. If $r_{j,\kappa}^{(2)} = 0$ then it must already be true at the beginning of iteration η that $\hat{\mathbf{a}}_{\gamma,\tau(\eta)}^{(2)} = 1$. If at the beginning of iteration η it holds that $\hat{x}_{\tau(\eta),\kappa}^{(2)} = x_{\ell(\eta),\kappa}$ and $r_{j,\kappa}^{(2)} = x_{\ell(\eta),\kappa}$, then the for loop on lines 30-35 of EXTRACT&MATCH ensures by the end of iteration η that $\hat{\mathbf{a}}_{\gamma,\tau(\eta)}^{(2)} = 1$. If at the beginning of iteration η it holds instead that $\hat{x}_{\tau(\eta),\kappa}^{(2)} = 0$ and $r_{j,\kappa}^{(2)} = x_{\ell(\eta),\kappa}$, then $\ell(\eta) \in \Omega_p \backslash \Gamma_p$ for all $p \in \text{supp}(\mathbf{a}_{\ell(\eta)})$. Therefore, as $f(x_{\ell(\eta),\kappa}, \mathbf{r}_\kappa^{(1)}) > (1 - 2\epsilon)d$ then $f(x_{\ell(\eta),\kappa}, \mathbf{r}_\kappa^{(2)}) > (1 - 2\epsilon)d$. Therefore during the subroutine EXTRACT&MATCH on line 13 of D-EBF, the partial support $\mathbf{w} = g(x_{\ell(\eta),\kappa}, \mathbf{r}_\kappa^{(2)})$, which satisfies $\gamma \in \text{supp}(\mathbf{w})$ and $|\text{supp}(\mathbf{w})| > (1 - 2\epsilon)d$, is extracted. During the CLUSTER&ADD subroutine on line 20 then by Lemma 2.2.7 it follows that $\hat{\mathbf{a}}_{\tau(\eta)}^{(2)} \leftarrow \sigma(\hat{\mathbf{a}}_{\tau(\eta)}^{(2)} + \mathbf{w})$. It follows that by the end of iteration η $\hat{a}_{\gamma,\tau(\eta)}^{(2)} = 1$. As $\gamma \in \text{supp}(\mathbf{a}_{\ell(\kappa)})$ was arbitrary, then it follows that $\hat{\mathbf{a}}_{\tau(\eta)}^{(2)} = \mathbf{a}_{\ell(\kappa)}$ at the end of iteration η. Therefore, if $\mathcal{Q}(\eta - 1)$ is true then $\mathcal{Q}(\eta)$ is true. This concludes the proof of the lemma. $\square$

2.3.2 Proof of Theorem 2.1.3

Together, Corollary 2.3.1.1 and Lemma 2.3.3 allow us to lower bound the performance D-EBF by analysing the conditions under which ND-EBF recovers a column of $\mathbf{A}$ at each iteration of the while loop, lines 6-31 of Algorithm 8. To this end, we first lower bound N in order to lower bound with high probability the number of nonzeros per row of X.

Lemma 2.3.4 (**Nonzeros per row in** X). *For some* $\beta \in \mathbb{Z}_{\geq 0}$ *and* $\mu > 1$, *if* $N \geq \beta\left(\mu \frac{n}{k} \ln(n) + 1\right)$ *then the probability that the random matrix* X, *as defined in the PSB model Definition 2.1.3, has at least* β *nonzeros per row is more than* $\left(1 - n^{-(\mu-1)}\right)^{\beta}$.

Proof. Let $X(k, \beta, r^c)$ denote a random binary matrix constructed by concatenating i.i.d. random, binary column vectors whose supports are drawn uniformly at random across all possible supports with a fixed cardinality k, until there are at least β non-zeros per row. Here r^c denotes *without replacement* in regard to how the support of an individual column is chosen. Let $X(k, \beta, r)$ denote a random binary matrix constructed by concatenating i.i.d. random, binary column vectors *with replacement*, by which it is meant that the support of each column is generated by taking k i.i.d samples uniform from $[n]$ with replacement, until there are at least β non-zeros per row. Let $\eta[X(k, \beta, r^c)]$ be the random variable which counts the number of columns of $X(k, \beta, r^c)$: here we adopt square brackets purely for typographical clarity. Note that the supports of the columns in the random matrix X defined in Definition 2.1.3 are drawn in the same manner as those of the random matrix $X(k, \beta, r^c)$. The difference between these random matrices is that the number of columns N of X is fixed in advance instead of, as is the case for $X(k, \beta, r^c)$, drawing enough columns until there are at least β nonzeros per row. Therefore

$$\mathbb{P}\left(\text{“X has at least } \beta \text{ non-zeros per row”}\right) = \mathbb{P}\left(\eta[X(k, \beta, r^c)] \leq N\right).$$

Or goal then is equivalent to lower bound $\mathbb{P}\left(\eta[X(k, \beta, r^c)] \leq N\right)$. To this end we first note that this problem can be interpreted as a generalisation of the coupon collector and dixie cup problems [45, 39, 114], which correspond to analysing the expectations of $\eta[X(1, 1, r^c)]$ and $\eta[X(\beta, 1, r^c)]$ respectively. For context, in the classic coupon collector problem, objects are drawn one at a time from a set of size n, typically uniformly at random with replacement until each object has been seen once. We again emphasise that in the notation adopted here r^c refers to sampling the

entries in the support of a given column without replacement, not the supports of the columns themselves which are mutually independent. The generalisation studied here is equivalent to analysing the number of draws of k objects from $[n]$, uniform at random with replacement, until each object has been picked at least β times with high probability. As the distribution of $\eta[X(k,\beta,r^c)]$ is challenging to analyse directly, we instead bound the quantity of interest using the classic coupon collector distribution, for which Chernoff style bounds can more easily be derived.

Using the notation described above, let $\left(X^{(i)}(k,1,r^c)\right)_{i=1}^{\beta}$ be mutually independent and identically distributed random matrices and define

$$\tilde{X}(k,\beta,r^c) := [X^{(1)}(k,1,r^c),\ X^{(2)}(k,1,r^c)...\ X^{(\beta)}(k,1,r^c)].$$

As each of the submatrices $X^{(i)}(k,1,r^c)$ may contain rows with more than 1 nonzero in them then clearly it holds that $\eta[X(k,\beta,r^c)] \leq \eta[\tilde{X}(k,\beta,r^c)]$, and therefore

$$\mathbb{P}(\eta[X(k,\beta,r^c)] \leq N) \geq \mathbb{P}(\eta[\tilde{X}(k,\beta,r^c)] \leq N).$$

Observe that

$$\bigcap_{i=1}^{\beta}\{\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\} \implies \{\eta[\tilde{X}(k,\beta,r^c)] \leq N\},$$

$$\{\eta[\tilde{X}(k,\beta,r^c)] \leq N\} \not\implies \bigcap_{i=1}^{\beta}\{\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\},$$

therefore $\bigcap_{i=1}^{\beta}\{\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\} \subseteq \{\eta[\tilde{X}(k,\beta,r^c)] \leq N\}$. As a result

$$\begin{aligned}
\mathbb{P}(\eta[X(k,\beta,r^c)] \leq N) &\geq \mathbb{P}(\eta[\tilde{X}(k,\beta,r^c)] \leq N) \\
&\geq \mathbb{P}\left(\bigcap_{i=1}^{\beta}\{\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\}\right) \\
&= \prod_{i=1}^{\beta}\mathbb{P}\left(\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\right) \\
&= \left[\mathbb{P}\left(\eta[X(k,1,r^c)] \leq \frac{N}{\beta}\right)\right]^{\beta}
\end{aligned}$$

where the equalities on the third and fourth lines follow from the assumptions of mutual independence and identical distribution respectively. If the support of a column is sampled with replacement then this will potentially decrease its cardinality relative to if it were sampled without replacement. It follows that

$$\mathbb{P}\left(\eta[X(k,1,r^c)] \leq \frac{N}{\beta}\right) \geq \mathbb{P}\left(\eta[X(k,1,r)] \leq \frac{N}{\beta}\right).$$

Observe that sampling k elements of the support of a column with replacement is equivalent to unioning the supports of k mutually independent, identically distributed column vectors, each with only one element in their support, drawn uniformly at random across all n possible locations. To this end, consider the random matrix $X(1,1)$, where the r argument has been dropped as sampling with or without replacement when there is only one nonzero per column are equivalent. The columns of this matrix are i.i.d. random vectors of dimension n with a single nonzero, whose location is drawn uniformly at random. Furthermore, sufficiently many of these columns are drawn and concatenated so as to ensure that $X(1,1)$ has at least one nonzero per row. Denote the lth column of $X(1,1)$ as Z_l. Let X' be a random binary matrix which is a deterministic function of $X(1,1)$ with $\text{supp}(X'_l) := \bigcup_{i=(l-1)k+1}^{lk} \text{supp}(Z_i)$ for all $l \in [\lceil \eta[X(1,1)]/k \rceil]$ [4]. Therefore, given that for any $x \in \mathbb{R}$ it holds that $\lceil x \rceil \leq x+1$, then

$$\begin{aligned}
\mathbb{P}\left(\eta[X(k,1,r^c)] \leq \frac{N}{\beta}\right) &\geq \mathbb{P}\left(\eta[X(k,1,r)] \leq \frac{N}{\beta}\right) \\
&= \mathbb{P}\left(\eta[X'] \leq \frac{N}{\beta}\right) \\
&= \mathbb{P}\left(\lceil \eta[X(1,1)]/k \rceil \leq \frac{N}{\beta}\right) \\
&\geq \mathbb{P}\left(\eta[X(1,1)] \leq k\left(\frac{N}{\beta} - 1\right)\right).
\end{aligned}$$

By inspection $\eta[X(1,1)]$ is equivalent to the number of draws required to see each coupon at least once in the classic coupon collector problem, for which the following bound with $t > 0$ is well known (see e.g., [34]),

$$\mathbb{P}(\eta[X(1,1)] > n\ln(n) + tn) < e^{-t}.$$

Letting $k\left(\frac{N}{\beta} - 1\right) = n\ln(n) + tn$, then rearranging gives $t = \frac{k(N-\beta) - \beta n \ln(n)}{\beta n}$. Assuming $n > 1$, then for some $\mu > 1$ if $N = \beta\left(\mu \frac{n}{k} \ln(n) + 1\right)$, then $t = (\mu - 1)\ln(n)$ and therefore

$$\begin{aligned}
\mathbb{P}\left(\eta[X(1,1)] \leq k\left(\frac{N}{\beta} - 1\right)\right) &= 1 - \mathbb{P}\left(\eta[X(1,1)] > k\left(\frac{N}{\beta} - 1\right)\right) \\
&> 1 - e^{-(\mu-1)\ln(n)} \\
&= 1 - n^{-(\mu-1)}.
\end{aligned}$$

[4] Note that if $\eta[X(1,1)]$ is not a multiple of k, then additional columns can be drawn so that the support of the last column of X' is still the union of the supports of k columns.

Therefore, if X has $N \geq \beta\left(\mu\frac{n}{k}\ln(n)+1\right)$ columns whose supports are sampled as described in Definition 2.1.3, then

$$\mathbb{P}\left(\text{“X has at least } \beta \text{ non-zeros per row”}\right) > \left(1-n^{-(\mu-1)}\right)^{\beta}$$

as claimed. □

Assuming a lower bound on the number of nonzeros per row of X and that Y is drawn from the PSB model, Lemma 2.3.5 provides a lower bound on the probability that a column of A is recovered at iteration $\eta \in [n]$ of the while loop, lines 6-31, of ND-EBF.

Lemma 2.3.5 (Probability that ND-EBF recovers a column at iteration η). *Let $Y = AX$ as per the PSB model, detailed in Definition 2.1.3, with $k = \alpha_1 n + 1$ and $m = \alpha_2 n$, where $\alpha_1, \alpha_2 \in (0,1)$ are fixed constants and $\alpha_1 \leq 1 - \frac{d}{m}$. Assume that each row of X has at least $\beta(n) = (1+2\epsilon d)L(n)$ nonzeros where $L : \mathbb{N} \to \mathbb{N}$. Suppose that ND-EBF, Algorithm 8, is deployed to try and compute the factors of Y and that the algorithm reaches iteration $\eta \in [n]$ of the while loop. Then there is a unique column $A_{\ell(\eta)}$ of A, satisfying $A_{\ell(\eta)} \neq \hat{A}_l$ for all $l \in [\eta - 1]$, such that*

$$\mathbb{P}(\hat{A}_\eta = A_{\ell(\eta)}) > 1 - de^{-\tau(n)L(n)},$$

where $\tau(n) := -\ln\left(1-\left(1-\frac{b}{\alpha_1 n}\right)^{\alpha_1 n}\right)$ is $\mathcal{O}(1)$.

Proof. Consider the start of iteration η of the while loop of ND-EBF, then from Lemma 2.3.1 the following must hold: $\hat{A}_l$ and $\tilde{\hat{X}}_l$ are nonzero iff $l < \eta$, if $l < \eta$ then $|\operatorname{supp}(\hat{A}_l)| = d$, there exists a random permutation matrix $P \in \mathcal{P}^{n\times n}$ such that $\operatorname{supp}(\hat{A}P^T) \subseteq \operatorname{supp}(A)$, $\operatorname{supp}(P\hat{X}) \subseteq \operatorname{supp}(X)$ and $\hat{X}_{P(l),i} = X_{P(l),i}$ for all $i \in [N]$ and $l \in \operatorname{supp}(\hat{X}_l)$, where $P : [n] \to [n]$ denotes the row permutation caused by pre-multiplication with the random matrix P. With $V := \hat{A}P^T$ and $U := P\hat{X}$ then by the construction of ND-EBF the residual can be expressed as

$$R = Y - \hat{A}\hat{X} = AX - VU = A(X-U).$$

Defining $Z := X - U$, as $X \in \chi_k^{n\times N}$ and $U_{l,i} = X_{l,i}$ for all $(l,i) \in \operatorname{supp}(U)$, then $Z \in \chi_k^{n\times N}$, $\operatorname{supp}(Z) \subseteq \operatorname{supp}(X)$ and $Z_{l,i} = X_{l,i}$ for all $(l,i) \in \operatorname{supp}(Z)$. Consider now a column $i \in [N]$ of the residual, inputted to EXTRACT&MATCH on line 12 of ND-EBF. By the construction of DECODE as well as Lemma 2.2.6, any partial support extracted from R_i must originate from a column A_h such that $A_h \neq \hat{A}_l$ for

all $l \in [\eta - 1]$, otherwise this partial support would have been extracted by DECODE during iteration $\eta - 1$. Let

$$\mathcal{T}_i := \{\alpha \in \mathbb{R} : f(\alpha, R_i) > (1 - 2\epsilon)d\}$$

denote the set of singleton values appearing more than $(1 - 2\epsilon)d$ extracted from R_i during EXTRACT&MATCH. Then using Lemma 2.2.2 we can lower bound the number of partial supports extracted across all columns of the residual at iteration η as follows.

$$\begin{aligned}
\sum_{i \in [N]} |\mathcal{T}_i| &> \frac{1}{(1+2\epsilon)d} \sum_{i=1}^{N} |supp(Z_i)| \\
&= \frac{1}{(1+2\epsilon)d} \sum_{j=1}^{n} |supp(\tilde{Z}_j)| \\
&\geq \frac{1}{(1+2\epsilon)d} \sum_{j=\eta}^{n} |supp(\tilde{X}_j)| \\
&\geq \frac{(n-\eta+1)\beta(n)}{(1+2\epsilon)d} \\
&= (n-\eta+1)L(n).
\end{aligned}$$

Th inequality on the first line follows directly from Lemma 2.2.2. The equality on the the second line is clearly true as $\sum_{i=1}^{N} |supp(Z_i)| = |\operatorname{supp}(Z)| = \sum_{j=1}^{n} |supp(\tilde{Z}_j)|$. The inequality on the third line follows from Lemma 2.3.1. The fourth inequality and final equality on line 5 arise as a result of the fact that we are conditioning on there being at least $\beta(n) = (1 + 2\epsilon)dL(n)$ nonzeros per row of X. As there are more than $(n - \eta + 1)L(n)$ partial supports, all of which belong to one of the $n - \eta + 1$ columns of A not yet recovered, then by the pigeon hole principle there exists at least one cluster of partial supports with cardinality at least $L(n)$. Therefore, on line 17 of MAXCLUSTER&ADD it must follow that $|\mathcal{C}^*| \geq L(n)$.

For clarity let the index of each partial support in C^* correspond to the column of Y from which it originates, for example, W_i is extracted from R_i and the corresponding column of X is therefore X_i. Note that as $\epsilon \leq 1/6$ then each column of Y can only provide a maximum of one partial support with a support cardinality of at least $2\epsilon d$ per column of A. Therefore this choice of indexing does not result in any ambiguity. From line 18 of MAXCLUSTER&ADD then equivalently

$$\hat{A}_\eta = \sigma\left(\sum_{i \in C^*} W_i\right).$$

By Lemmas 2.2.4 and 2.3.1 there exists a unique column $A_{\ell(\eta)}$ such that $\text{supp}(\hat{A}_\eta) \subseteq \text{supp}(A_{\ell(\eta)})$. We now proceed to upper bound the probability that $\hat{A}_\eta \neq A_{\ell(\eta)}$. Given that $|\mathcal{C}^*| \geq L(n)$ then

$$\begin{aligned}
\mathbb{P}(\hat{A}_\eta \neq A_{\ell(\eta)}) &= \mathbb{P}\left(\left(\bigcup_{i=1}^{|\mathcal{C}^*|} \text{supp}(W_i)\right) \neq \text{supp}(A_{\ell(\eta)})\right) \\
&\leq \mathbb{P}\left(\left(\bigcup_{i=1}^{L(n)} \text{supp}(W_i)\right) \neq \text{supp}(A_{\ell(\eta)})\right) \\
&= \mathbb{P}\left(\bigcup_{j \in \text{supp}(A_{\ell(\eta)})} \left(j \notin \left(\bigcup_{i=1}^{L(n)} \text{supp}(W_i)\right)\right)\right)
\end{aligned}$$

Given that $|\text{supp}(A_l)| = d$, then applying the union bound it follows that

$$\begin{aligned}
\mathbb{P}(\hat{A}_\eta \neq A_{\ell(\eta)}) &\leq d\mathbb{P}\left(j \notin \left(\bigcup_{i=1}^{L(n)} \text{supp}(W_i)\right) \mid j \in \text{supp}(A_{\ell(\eta)})\right) \\
&= d\mathbb{P}\left(\bigcap_{i=1}^{L(n)} (j \notin \text{supp}(W_i)) \mid j \in \text{supp}(A_{\ell(\eta)})\right).
\end{aligned}$$

If $j \notin \text{supp}(W_i)$ then there exists a $h \in \text{supp}(Z_i)\backslash\{\eta\}$ such that $j \in \text{supp}(A_h)$. To be clear, if $j \in \text{supp}(A_{\ell(\eta)})$ but $j \notin \text{supp}(W_i)$, then there must exist a column A_h in the submatrix $A_{\text{supp}(Z_i)\backslash\{\ell(\eta)\}}$ which has a 1 in its jth entry or row, thereby resulting in $R_{j,i}$ not being a singleton value. Letting $\zeta := \lceil nd/m \rceil$ then

$$\begin{aligned}
&\mathbb{P}\left(\bigcap_{i=1}^{L(n)} (j \notin \text{supp}(W_i)) \mid j \in \text{supp}(A_{\ell(\eta)})\right) \\
&\leq \mathbb{P}\left(\bigcap_{i=1}^{L(n)} \left(j \in \left(\bigcup_{h \in \text{supp}(Z_i)\backslash\{\ell(\eta)\}} \text{supp}(A_h)\right)\right)\right) \\
&\leq \mathbb{P}\left(\bigcap_{i=1}^{L(n)} \left(j \in \left(\bigcup_{h \in \text{supp}(Z_i)\backslash\{\ell(\eta)\}} \text{supp}(A_h)\right)\right) \;\middle|\; |\text{supp}(\tilde{A}_j)| = \zeta\right) \\
&\leq \mathbb{P}\left(\bigcap_{i=1}^{L(n)} \left(j \in \left(\bigcup_{h \in \text{supp}(X_i)\backslash\{\ell(\eta)\}} \text{supp}(A_h)\right)\right) \;\middle|\; |\text{supp}(\tilde{A}_j)| = \zeta\right) \\
&= \prod_{i=1}^{L(n)} \mathbb{P}\left(j \in \left(\bigcup_{h \in \text{supp}(X_i)\backslash\{\ell(\eta)\}} \text{supp}(A_h)\right) \;\middle|\; |\text{supp}(\tilde{A}_j)| = \zeta\right).
\end{aligned}$$

The inequality on the second line follows from the number of nonzeros per row of A being bounded by construction, therefore the events $j \in \text{supp}(A_{\ell(\eta)})$ and $j \in \text{supp}(A_h)$ for $h \neq \ell(\eta)$ are negatively correlated. The inequality on the third line of the above follows by considering the jth row to achieve the upper bound ζ of nonzeros per row. The inequality on the fourth line follows from the fact that $\text{supp}(Z_i) \subseteq \text{supp}(X_i)$ for all $i \in [N]$. Lastly, the equality on the fifth line follows from the mutual independence of the supports of the columns of X. Clearly

$$j \in \left(\bigcup_{h \in supp(X_i) \backslash \{\ell(\eta)\}} \text{supp}(A_h) \right) \iff (\text{supp}(X_i) \cap \text{supp}(\tilde{A}_j)) \backslash \{\ell(\eta)\} \neq \emptyset.$$

Furthermore, note that by the construction of X in the PSB model, Definition 2.1.3, that the probability that none of the supports of the columns of A with index in $\text{supp}(X_i)\backslash\{\ell(\eta)\}$ contain j is simply a count of the number of draws of $\text{supp}(X_i)\backslash\{\ell(\eta)\}$ that contain none of the column indices in $\text{supp}(\tilde{A}_j)\backslash\{\ell(\eta)\}$, divided by the total number of possible draws of $\text{supp}(X_i)\backslash\{\ell(\eta)\}$. As a result

$$\mathbb{P}\left(j \in \left(\bigcup_{r \in \text{supp}(X_i)\backslash\{\ell(\eta)\}} \text{supp}(A_r) \right) \;\middle|\; |\,\text{supp}(\tilde{A}_j)| = \zeta \right) = 1 - \binom{n-\zeta}{k-1}\binom{n-1}{k-1}^{-1},$$

from which it follows that

$$\mathbb{P}(\hat{A}_l \neq A_l) \leq d\left(1 - \binom{n-\zeta}{k-1}\binom{n-1}{k-1}^{-1}\right)^{L(n)}. \tag{2.4}$$

To simplify equation 2.4, we bound the binomial coefficients as follows:

$$\begin{aligned}
\binom{n-\zeta}{k-1}\binom{n-1}{k-1}^{-1} &= \frac{(n-\zeta)!(n-k)!}{(n-\zeta-k+1)!(n-1)!} \\
&= \frac{(n-\zeta)(n-\zeta-1)...(n-\zeta-k+2)}{(n-1)(n-2)...(n-k+1)} \\
&\geq \left(\frac{n-\zeta-k+2}{n-k+1}\right)^{k-1}.
\end{aligned}$$

Therefore

$$\mathbb{P}(\hat{A}_l \neq A_l) \leq d\left(1 - \left(\frac{n-\zeta-k+2}{n-k+1}\right)^{k-1}\right)^{L(n)}. \tag{2.5}$$

Recalling that $k = \alpha_1 n + 1$, $m = \alpha_2 n$ and $\zeta = \lceil nd/m \rceil \leq nd/m + 1$, then

$$\begin{aligned}
\left(\frac{n-\zeta-k+2}{n-k+1}\right)^{k-1} &\geq \left(1 - \frac{d}{\alpha_2(1-\alpha_1)n}\right)^{\alpha_1 n} \\
&= \left(1 - \frac{b}{\alpha_1 n}\right)^{\alpha_1 n}
\end{aligned}$$

where $b := \frac{d}{\alpha_2(1-\alpha_1)}$. Taking the exponential of the logarithm of the upper bound provided in equation 2.5, then

$$\begin{aligned}\mathbb{P}(\hat{A}_\eta \neq A_{\ell(\eta)}) &\leq d\left(1-\left(1-\frac{b}{\alpha_1 n}\right)^{\alpha_1 n}\right)^{L(n)} \\ &= de^{-\tau(n)L(n)}\end{aligned}$$

where $\tau(n) := -\ln\left(1-\left(1-\frac{b}{\alpha_1 n}\right)^{\alpha_1 n}\right)$. Observe that as $\alpha_1 \leq 1-\frac{d}{m}$ then $n > \frac{d}{\alpha_2(1-\alpha_1)}$. By inspection this ensures that $\tau(n)$ is a monotonically increasing function of n. Furthermore, as $\lim_{n\to\infty}\tau(n) = -\ln(1-e^{-b})$ then $\tau(n) = \mathcal{O}(1)$ as claimed. □

With Lemmas 2.3.4 and 2.3.5 in place we are ready to prove Theorem 2.1.3. Statements 1 and 2 of Theorem 2.1.3 are immediate consequences of Lemma 2.2.7, therefore all that is left to prove is Statement 3. To provide a brief recap, our objective is to recover up to permutation the random factor matrices A and X, as defined in the PSB model in Definition 2.1.3, from the random matrix $Y := AX$. Our strategy at a high level is as follows: using Lemma 2.3.3 and Corollary 2.3.1.1 we lower bound the probability that D-EBF successfully factorises Y up to permutation by lower bounding the probability that ND-EBF recovers A up to permutation. ND-EBF recovers A up to permutation iff at each iteration of the while loop, lines 6-31 of Algorithm 8, a new column of A is recovered. We lower bound the probability of this event in turn using Lemma 2.3.5. For the proof Theorem 2.1.3 we adopt the following notation.

- $\Lambda^*_{\text{D}-\text{EBF}}$ and $\Lambda^*_{\text{ND}-\text{EBF}}$ are the events that D-EBF and ND-EBF recover A and X up to permutation respectively, i.e., there exists a random permutation $P \in \mathcal{P}^{n\times n}$ such that $\hat{A}P^T = A$ and $P\hat{X} = X$.

- $\Lambda_0 := \{A \in \mathcal{E}^{m\times n}_{k,\epsilon,d}\} \cap \{\epsilon \leq 1/6\}$ is the event that the random matrix A is the adjacency matrix of a (k,ϵ,d)-expander graph with expansion parameter $\epsilon \leq 1/6$.

- For $\eta \in [n]$ let Λ_η denote the event that at the end of ηth iteration of the while loop of ND-EBF, there exists a unique column $A_{\ell(\eta)}$ of A satisfying $A_{\ell(\eta)} = \hat{A}_\eta$ and $A_{\ell(\eta)} \neq \hat{A}_l$ for all $l \in [n]\backslash\{\eta\}$.

- Λ_{n+1} denotes the event that each row of X has at least $\beta(n) := (1+2\epsilon)dL(n)$ nonzeros per row where $L(n) := \lceil \frac{\gamma \ln(n)}{\tau(n)} \rceil$ and $\gamma > 1$ is a constant.

Proof. Suppose that ND-EBF is deployed instead of D-EBF to recover the matrix factors of Y. From Corollary 2.3.1.1, for ND-EBF to recover both factors up to permutation it suffices to recover A up to permutation. Clearly then

$$\mathbb{P}\left(\Lambda^*_{ND-EBF} \mid \Lambda_0\right) = \mathbb{P}\left(\bigcap_{h=1}^{n} \Lambda_h \mid \Lambda_0\right).$$

We now apply Bayes' Theorem and condition on Λ_{n+1},

$$\begin{aligned}
\mathbb{P}\left(\Lambda^*_{ND-EBF} \mid \Lambda_0\right) &= \mathbb{P}\left(\bigcap_{\eta=1}^{n} \Lambda_\eta \mid \Lambda_0\right) \\
&= \frac{\mathbb{P}\left(\bigcap_{\eta=1}^{n} \Lambda_\eta, \Lambda_0\right)}{\mathbb{P}\left(\Lambda_0\right)} \\
&\geq \frac{\mathbb{P}\left(\bigcap_{\eta=1}^{n} \Lambda_\eta, \Lambda_0 \mid \Lambda_{n+1}\right)\mathbb{P}\left(\Lambda_{n+1}\right)}{\mathbb{P}\left(\Lambda_0\right)} \\
&= \frac{\mathbb{P}\left(\bigcap_{\eta=1}^{n} \Lambda_\eta \mid \Lambda_0, \Lambda_{n+1}\right)\mathbb{P}\left(\Lambda_0 \mid \Lambda_{n+1}\right)\mathbb{P}\left(\Lambda_{n+1}\right)}{\mathbb{P}\left(\Lambda_0\right)} \\
&= \mathbb{P}\left(\bigcap_{\eta=1}^{n} \Lambda_\eta \mid \Lambda_0, \Lambda_{n+1}\right)\mathbb{P}\left(\Lambda_{n+1}\right) \\
&= \mathbb{P}\left(\Lambda_{n+1}\right)\prod_{\eta=1}^{n}\mathbb{P}\left(\Lambda_\eta \mid \bigcap_{l=0}^{\eta-1} \Lambda_l, \Lambda_{n+1}\right).
\end{aligned}$$

In the above, line 2 follows as a result of Bayes Theorem, line 3 is an application of the law of total probability and line 4 is derived using the probability chain rule. The equality on line 5 follows as A and X are drawn independently of one another, therefore, given that Λ_0 is a property of A and Λ_{n+1} a property of X, $\mathbb{P}(\Lambda_0 \mid \Lambda_{n+1}) = \mathbb{P}\left(\Lambda_0\right)$. Finally, line 6 is once again an application of the probability chain rule. Assume for now that $N \geq \beta(n)\left(\mu\frac{n}{k}\ln(n)+1\right)$, where $\mu > 1$ is some constant. Then as a consequence of Lemma 2.3.4 it follows that

$$\mathbb{P}\left(\Lambda_{n+1}\right) > \left(1 - n^{-(\mu-1)}\right)^{\beta(n)}.$$

Applying Lemma 2.3.5 it also holds for any $\eta \in [n]$ that

$$\mathbb{P}\left(\Lambda_\eta \mid \bigcap_{l=0}^{\eta-1} \Lambda_l, \Lambda_{n+1}\right) > 1 - de^{-\tau(n)L(n)}.$$

where $\tau(n)$ is $\mathcal{O}(1)$. As a result

$$\mathbb{P}\left(\Lambda^*_{ND-EBF} \mid \Lambda_0\right) > \left(1 - n^{-(\mu-1)}\right)^{\beta(n)}\left(1 - de^{-\tau(n)L(n)}\right)^{n}.$$

Recalling that $L(n) := \lceil \frac{\gamma \ln(n)}{\tau(n)} \rceil$ where $\gamma > 1$ is an arbitrary constant, then analysing the right-hand factor it follows that

$$\begin{aligned}
\left(1 - de^{-\tau(n)L(n)}\right)^n &\geq \left(1 - de^{\ln(1/n^\gamma)}\right)^n \\
&= \left(1 - \frac{d}{n^\gamma}\right)^n \\
&= \sum_{i=0}^{\infty} \binom{n}{i} (-1)^i \left(\frac{d}{n^\gamma}\right)^i \\
&= 1 - \frac{d}{n^{\gamma-1}} + \sum_{i=2}^{\infty} \binom{n}{i} (-1)^i \left(\frac{d}{n^\gamma}\right)^i \\
&= 1 - \mathcal{O}\left(n^{-\gamma+1}\right).
\end{aligned}$$

Above, the inequality on the first line follows from the fact that $\lceil \frac{\gamma \ln(n)}{\tau(n)} \rceil \geq \frac{\gamma \ln(n)}{\tau(n)}$. The equality on the third line follows by applying the binomial series expansion. Analysing now the left-hand factor derived from Lemma 2.3.4, note that as $\beta(n) = (1+2\epsilon)d\lceil \frac{\gamma \ln(n)}{\tau(n)} \rceil$ and $\tau(n) = \mathcal{O}(1)$ then $\beta(n) = \mathcal{O}(\log(n))$. As a result

$$\begin{aligned}
\left(1 - n^{-(\mu-1)}\right)^{\beta(n)} &= \sum_{i=0}^{\infty} \binom{\beta(n)}{i} (-1)^i \left(n^{-\mu+1}\right)^i \\
&= 1 - \frac{\beta(n)}{n^{\mu-1}} + \sum_{i=2}^{\infty} \binom{\beta(n)}{i} (-1)^i \left(n^{-\mu+1}\right)^i \\
&= 1 - \mathcal{O}\left(n^{-\mu+1} \log(n)\right).
\end{aligned}$$

Therefore we arrive at the asymptotic lower bound

$$\mathbb{P}\left(\Lambda^*_{ND-EBF} \mid \Lambda_0\right) > \left(1 - \mathcal{O}\left(n^{-\mu+1}\log(n)\right)\right)\left(1 - \mathcal{O}\left(n^{-\gamma+1}\right)\right).$$

Noting that $\epsilon \leq 1/6$ implies $(1+2\epsilon) \leq \frac{4}{3}$, then

$$\begin{aligned}
\beta(n)\left(\mu \frac{n}{k}\ln(n) + 1\right) &= (1+2\epsilon)d\lceil \frac{\gamma \ln(n)}{\tau(n)} \rceil \left(\mu \frac{n}{k}\ln(n) + 1\right) \\
&\leq \gamma\mu \frac{4d}{3\tau(n)} \frac{n}{k} (\ln^2(n) + \ln(n)) \\
&\leq N
\end{aligned}$$

by construction. Given that $\gamma, \mu > 1$ are arbitrary constants let $\gamma = \mu$ and define $\nu := \gamma^2$, clearly it must also hold that $\nu > 1$. It follows that if $N \geq \nu \frac{4d}{3\tau(n)} \frac{n}{k} (\ln^2(n) + \ln(n))$ then

$$\mathbb{P}\left(\Lambda^*_{ND-EBF} \mid \Lambda_0\right) > 1 - \mathcal{O}(n^{-\sqrt{\nu}+1}\log(n)).$$

Therefore, by Lemma 2.3.3, it also holds that if $N \geq \nu \frac{4d}{3\tau(n)} \frac{n}{k} (\ln^2(n) + \ln(n))$ then

$$\mathbb{P}\left(\Lambda^*_{D-EBF} \mid \Lambda_0\right) \geq \mathbb{P}\left(\Lambda^*_{ND-EBF} \mid \Lambda_0\right) > 1 - \mathcal{O}(n^{-\sqrt{\nu}+1}\log(n))$$

as claimed. This concludes the proof of Theorem 2.1.3 □

2.4 Experiments

A key takeaway of Theorem 2.1.3 is that there are parameter regimes in which, at least asymptotically, D-EBF will successfully factorise a matrix Y drawn from the PSB model. In this section we investigate this matter empirically. First, in Section 2.4.1, we highlight certain adjustments to D-EBF which enable it to be deployed in practice: in particular, we discuss how to circumvent the issue of not having access to the expansion parameter ϵ, and also how D-EBF can be adapted to be used in an online setting. Second, in Section 2.4.2, we conduct experiments demonstrating the efficacy of D-EBF in factorising even mid-sized problems, i.e., $n = 10^3$.

2.4.1 Deploying D-EBF in practice

As highlighted in Section 2.1.3, in practice the decoder may not have access to the expansion parameter of the encoder matrix. One reason for this, already highlighted in Section 2.1.3, is that computing the expansion parameter is an NP-complete problem. D-EBF is designed for factorisation problems in which $\epsilon \leq 1/6$, we therefore restrict our attention to this setting and consider how the decoder might still be able to succeed in factorising $\mathbf{Y}$ without knowledge of ϵ in advance. To this end, suppose that $1/6$ is passed to D-EBF instead of ϵ: as $2\epsilon d \leq d/3$ then taking this action places a stricter requirement on the frequency with which an entry of the residual must occur in order to be accepted as a candidate singleton value. In addition, as $(1-2\epsilon)d \geq 2d/3$, then all singleton values occurring at least $(1-2\epsilon)d$ times in a given column are still extracted. Therefore, from the perspective of extracting singleton values and partial supports, the only implication of using $1/6$ instead of ϵ is that some singleton values and their associated partial supports may be missed.

In regard to clustering however, as $(1-4\epsilon)d \geq 2d/3$, then using $1/6$ instead of ϵ loses the guarantees afforded by Lemma 2.2.4. To be clear, consider two partial supports $\mathbf{w}_1$ and $\mathbf{w}_2$ satisfying $|\operatorname{supp}(\mathbf{w}_1)| > 2d/3$ and $|\operatorname{supp}(\mathbf{w}_2)| > 2d/3$: if $\mathbf{w}_1^T\mathbf{w}_2 \geq d/3 \geq 2\epsilon d$ then we can conclude that these two partial supports originate from the same column, however $\mathbf{w}_1^T\mathbf{w}_2 < d/3$ does not guarantee that these partial supports originate from different columns. If D-EBF is deployed as described in Algorithm 6 this may result in duplicate reconstructions of the same column of $\mathbf{A}$ in $\hat{\mathbf{A}}$. This issue can be overcome by adding in an additional column merge subroutine at the end of each iteration. An example merge subroutine is provided in Algorithm 9. This algorithm merges duplicate columns in $\hat{\mathbf{A}}$, as well as the corresponding rows of $\hat{\mathbf{X}}$, by checking the inner product between columns of $\hat{\mathbf{A}}$: assuming that $\epsilon \leq 1/6$ then if any

pair of columns has an inner product larger than $d/3$ then as $d/3 \geq 2\epsilon d$ then we may conclude that these two columns and their respective rows in $\hat{\mathbf{X}}$ should be combined.

Algorithm 9 MERGE$(d, \hat{\mathbf{A}}, \hat{\mathbf{X}})$

G $\leftarrow \hat{\mathbf{A}}^T \hat{\mathbf{A}}$
for $i = 1 : n$ **do**
  if $|\text{supp}(\hat{\mathbf{a}}_i)| \geq d/3$ **then**
    for $j = (i+1) : n$ **do**
      if $g_{i,j} \geq d/3$ **then**
        $\hat{\mathbf{a}}_i \leftarrow \sigma(\hat{\mathbf{a}}_i + \hat{\mathbf{a}}_j)$
        $\hat{\mathbf{a}}_j \leftarrow \mathbf{0}_m$
        $\tilde{\hat{\mathbf{x}}}_i \leftarrow \tilde{\hat{\mathbf{x}}}_i + \tilde{\hat{\mathbf{x}}}_j$
        $\tilde{\hat{\mathbf{x}}}_j \leftarrow \mathbf{0}_N^T$
      end if
    end for
  end if
end for
Return $\hat{\mathbf{A}}, \hat{\mathbf{X}}$

In summary, as long as $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m \times n}$ with $\epsilon \leq 1/6$, then in practice even without access to the expansion parameter ϵ D-EBF can still be deployed by using the upper bound $1/6$ instead. We demonstrate this empirically in Section 2.4.2. Note also that the same reasoning applies to any bound on ϵ made available to the decoder in advance.

As defined in Algorithm 6, D-EBF requires knowledge of the column sparsity d and number of columns n of $\mathbf{A}$. We speculate that it may be possible to estimate or use upper and lower bounds on d and therefore still deploy D-EBF without access to d in advance. We also hypothesise that similar results and algorithms may be able to be derived for the situation where A is not the adjacency matrix of a fixed degree expander, with instead the degree of each node being bounded in some interval with high probability. We leave the study of such questions to future work and proceed under the assumption that the decoder has access to d in advance. With regard requiring access to n, note that this is an artefact of our problem setup rather than a necessity in practice. Indeed, instead of initialising $\hat{\mathbf{A}}$ as an $m \times n$ array of zeros, the reconstruction of $\mathbf{A}$ could instead be kept dynamic, with partial or complete columns of $\mathbf{A}$ being added to $\hat{\mathbf{A}}$ as and when they are recovered. Furthermore, D-EBF can be run in advance of receiving all of the columns of $\mathbf{Y}$. Consider the situation in which the columns of $\mathbf{Y}$ are received in sequence: as each new column arrives D-EBF can be ran only on this new column, in a so called turnstile model, or in combination with the

residuals of previous columns, using the partial reconstruction of $\mathbf{A}$ already acquired as an initialisation point. In either case, after the decoder has seen sufficiently many columns to enable it to recover the encoder matrix up to permutation, then future columns can be decoded efficiently as and when they arrive using a decoding algorithm from the CCS literature.

To summarise, in addition to $\mathbf{Y}$, then in practice D-EBF requires access only to the column sparsity d of the encoder in advance. Furthermore, D-EBF can easily be adapted to process subsets of columns of $\mathbf{Y}$, allowing it to be deployed for example in data streaming contexts.

2.4.2 Performance of D-EBF on mid-sized problems

As a result of Lemma 2.1.1 and Theorem 2.1.3, then for sufficiently large, sparse problems the assumptions upon which D-EBF is based hold with high probability. However, understanding the regimes in which D-EBF will succeed or fail at a practical level is not clear. Indeed, the probability bounds derived in [9] are loose, making it hard to specify exactly how large n needs to be in order for the encoder A, sampled from the PSB model, to satisfy $A \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ with high probability. Additionally, even if this assumption holds the probability bound provided in Theorem 4.1.1 is asymptotic and also likely to be loose. The purpose of this section is to provide a preliminary investigation into the empirical performance of D-EBF, testing the parameter settings for which the algorithm is likely to be successful. In particular, as already highlighted our theory suggests that D-EBF will likely be successful on large, sparse problems and unsuccessful on small, dense ones. We therefore focus our attention on mid-sized problems, varying N and the ratio k/n, in order to better understand the transition point between success and failure.

The parameter settings for our experiments are as follows: $n = 1000$, $m = 800$ and $d = 10$ are kept fixed, with k varied between 1% and 10% of n and $N \in \{100, 200, 300\}$. For each of these parameter settings 10 trials where computed: in each case an A and an X were generated as per the PSB model, Definition 2.1.3, with the coefficients of X being sampled from the uniform distribution over an interval bounded away from 0 (in this case $[0.1, 10.1]$). For each trial the D-EBF algorithm was then deployed to recover A and X up to permutation from $Y = AX$ with the ϵ input parameter set to 1/6: we note that in our experiments D-EBF, Algorithm 6, was not amended as per the discussion in Section 2.4.1 to include an additional merge subroutine. With $\hat{A}$ and $\hat{X}$ denoting the reconstructions returned by D-EBF, then for each parameter setting the Frobenius norm of the residual $||Y - \hat{A}\hat{X}||_F$ was computed as a percentage

of $||Y||_F$ and averaged over the 10 trials. In addition, for each parameter setting the number of iterations of the while loop of D-EBF, lines 6-25 of Algorithm 6, was also counted and likewise averaged over the 10 trials. The outcomes of these experiments are shown in Figure 2.2.

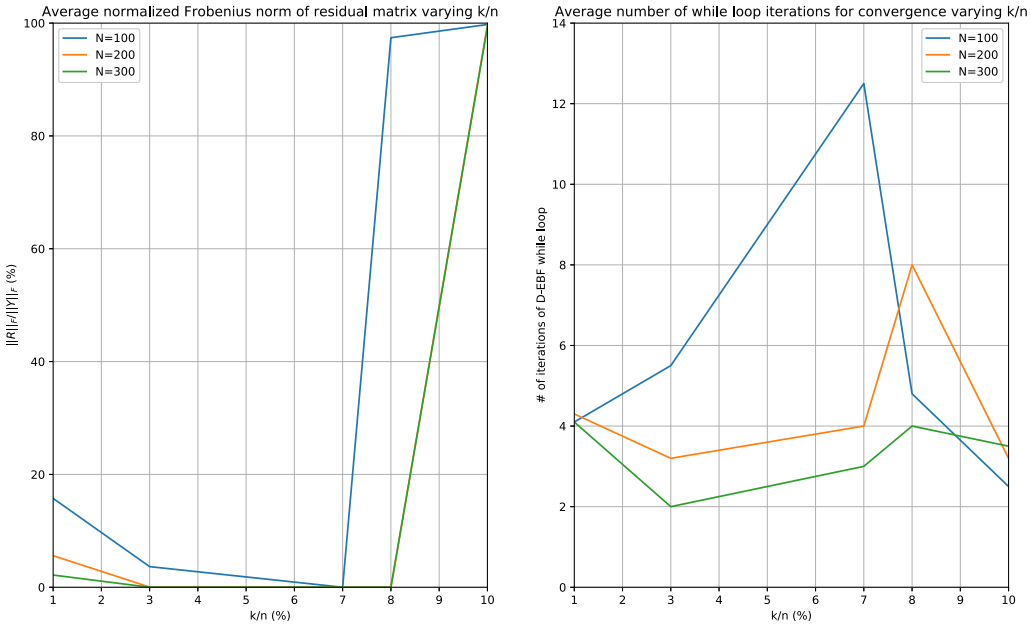


Figure 2.2: The left-hand plot shows the percentage $||\mathbf{Y} - \hat{\mathbf{A}}\hat{\mathbf{X}}||_F/||\mathbf{Y}||_F$, averaged over 10 trials with each trial generated as per the PSB model, Definition 2.1.3. Here $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ denote the reconstructions returned by D-EBF. The right-hand plot shows the average number of iterations of the while loop of D-EBF, lines 6-25, required for convergence over the same trials. The parameters k and N are varied while $n = 1000$, $m = 800$, $d = 10$ are fixed. In all trials the expansion parameter ϵ of A is not known or computed in advance, D-EBF is deployed using $1/6$ in its stead.

In the left-hand plot of Figure 2.2 we observe that Y is most likely to be successfully factorized when N is large and k/n is neither too small or too large. Indeed, a larger N will typically imply more partial supports extracted at each iteration of the while loop and therefore a greater chance of recovering A up to permutation. This in turn allows for the recovery of X up to permutation. Similar to the situation in which N is small, when k/n is small fewer partial supports are extracted per column of Y at each iteration of the while loop, making the recovery of A harder. As can be observed in the left-hand plot of Figure 2.2, this results in the incomplete factorisation of Y for $k/n < 3\%$. Note also for $k/n < 3\%$, that as N increases Y on average becomes closer to being fully factorized: this is because the partial supports extracted from the additional columns compensate for the reduced number of partial supports ex-

tracted per column. We would expect even for small k/n that as N increases further Y will converge towards being fully factorized. When k/n is large a similar issue occurs in terms of few or even no partial supports being extracted per column of Y. This is due to the fact that with m and n fixed then as k increases the probability that $A \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ converges towards 0. When this assumption is not satisfied it is likely no longer possible to identify singleton values using frequency of occurrence, let alone cluster their associated partial supports. This can be observed in the failure to make any progress in factorising Y when $k/n = 10\%$, furthermore increasing N in this context is unlikely to yield any improvement. The middle ground, $3\% \leq k/n \leq 8\%$, appears ripe for factorisation, striking the right balance between k being large enough so that each column of A contributes to many of the columns of Y, while k is sufficiently small so that singleton values can still be identified via the frequency with which they appear. Turning our attention to the right-hand plot of Figure 2.2, we observe that as N increases then the number of iterations of the while loop of D-EBF generally decreases. This is to be expected as a larger N means more partial supports extracted at each iteration, and therefore we would expect A to be recovered in fewer iterations. Observe for $N = 100$ and $k/n = 7\%$ that the iterative approach of D-EBF is invaluable in being able to factorise Y.

To summarise, these experiments demonstrate that it is possible to practically decode, in the sense of recovering the nonzero values of the underlying sparse vectors, a set of linear measurements with access only to the column sparsity d of the encoder matrix, rather than the full encoder matrix. While Theorem 2.1.3 provides asymptotic guarantees, these experiments illustrate that D-EBF is successful even on relatively small to medium sized problems. We expect for large problems the performance of D-EBF to improve, allowing for problems with larger k/n ratios to be factorized successfully. We leave a comprehensive empirical study of the D-EBF algorithm to future work.

2.5 Concluding remarks

In this chapter we introduced the permuted striped block (PSB) model, Definition 2.1.3, and its associated matrix factorisation task which is inspired by multi-measurement vector combinatorial compressed sensing (MMV-CSS). In this context, solving this factorisation task allows for the recovery of the nonzero values of the sparse codes without full knowledge of the sensing or encoder matrix. With additional information, e.g., the column ordering protocol described in Section 2.1.3,

then full MMV-CSS can be be achieved without access to the encoder. To factorise PSB matrices we presented the Decoder-Expander Based Factorisation (D-EBF) algorithm, proving that it factorises Y drawn from PSB model with high probability and that it achieves near optimal sample complexity. We then demonstrated that this algorithm performs well in practice even on mid-sized problems. In addition, the per while loop iteration computational complexity of D-EBF is $\mathcal{O}(k^2(n+N)N)$: this, combined with experimental evidence indicating only a modest number of iterations are required for convergence, as well as the opportunities for parallelization, suggest that the computational cost of D-EBF at scale is manageable. The main limitations of this work are the restrictive nature of the PSB model and the robustness of D-EBF to noise. Some potential extensions of this work are as follows.

- **Stability to noise and projections of arbitrary sparse matrices onto the set of PSB matrices:** more developed matrix factorizations, such as PCA, subspace clustering, and dictionary learning, are all known to be effective for matrices which only approximately achieve the modelling assumptions in context. A first extension of the PSB model would be to show that the factors are stable to additive noise and remain efficient and reliable to compute. Stability to additive noise one might expect to be achieved via related work in combinatorial compressed sensing, such as [87]. This could be further augmented by having repeated measurement vectors contained in Y. In order for the PSB model to be more generally applicable, approximation rates are required, in particular a derivation of the distance between an arbitrary sparse matrix $\mathbf{Y}$ and its projection onto the set of PSB matrices, which would presumably show better approximation as the number of columns n of the encoder matrix $\mathbf{A}$ is increased. To be clear, this would entail bounds of the form $\|\mathbf{Y} - P_{PSB(n)}(\mathbf{Y})\| \leq f(n)$, where $P_{PSB(n)}(\cdot)$ projects to the nearest PSB matrix with parameters k, m, n, d, N under some matrix norm, and $f(n)$ is a rapidly decreasing function of n and or potentially d and k.

- **Relaxing the $\epsilon \leq 1/6$ condition:** Lemma 2.2.4 guarantees that partial supports can be trivially clustered if the binary factor matrix A is a (k, ϵ, d)-expander graph with expansion parameter $\epsilon \leq 1/6$. This expansion parameter is guaranteed over all $\binom{n}{k}$ sets of k columns in A. As Y contains at most N such sets of k columns, and as N would typically be exponentially smaller than this binomial coefficient, it is expected that similar bounds could be derived under a model for which it holds only with high probability that a subset of k columns

of A satisfies the expansion bound. As a result, it seems reasonable that more robust extensions of the current clustering method could be conceived.

- **Broadening the applicability of PSB model:** two assumptions that restrict the expressivity of the PSB model are that there is a fixed number d nonzeros per column of A and that the columns of X are dissociated. Relaxing these conditions would clearly aid in the applicability of the model. One would expect it to be possible to derive similar results for the case in which the number of nonzeros per column of A is bounded from above and below. However, this would likely increase the computational complexity of the factorisation task as the number of nonzeros per column would also need to be inferred from Y. Robustness to the entries in X being dissociated, at least with high probability, can also be expected. However, adversarial choices of nonzero values in X may require combinatorial searches on certain entries of Y and could result in Y no longer having a unique factorisation.

- **Applications:** in Section 2.1.3 we sketched a few scenarios in which solving the PSB matrix factorisation task could be useful, namely where the objective is to recover the sparse codes or the statistics of the nonzeros when the decoder does not have access to the full encoder matrix in advance. Exploration and investigation of applications where this capability might be useful, and the efficacy of D-EBF in these contexts, could be valuable.

Chapter 3

Signal propagation in deep networks: activation pattern recovery under the DCSC model

As referenced to in Section 1.2.1, in this chapter we analyse the role of sparsifying activation functions in deep convolutional neural networks (DCNNs), with the goal of providing recovery guarantees for the forward pass. The structure of this chapter is as follows: in Section 3.1 we motivate the interpretation of the forward pass as solving a sequence of sparse coding problems, in Section 3.2 we then define and formalise the Deep Convolutional Sparse Coding (DCSC) model as well as summarising the key results of [96], in Section 3.3 we present the key contribution of this chapter, Theorem 3.3.1. Finally, in Section 3.4 we review our findings and offer some concluding remarks. This chapter is based on work accepted to the 2018 IEEE Data Science Workshop.

3.1 Motivation and related work

In this section we motivate the interpretation of the forward pass as solving a sequence of sparse coding problems as well as the DCSC model. An information processing task can be challenging or easy depending on the form or representation in which the data is provided in, we therefore proceed by adopting the common hypothesis that the role of the forward pass is to compute representations of the input data that are well suited for the task at hand. A natural question is whether there are any properties we might want the final, or indeed any intermediary, representations of the data to have, irrespective of a specific task. In general, applying priors on the representations at different layers of a network has both potential upsides and downsides: indeed, by applying a prior we in effect constrain the search over the parameter space, which

may lead to suboptimal results. On the other hand, applying such priors may help the optimisation procedure find a solution faster, or perhaps a solution which generalises better. Furthermore, beyond performance as measured by test and train accuracy, there may be other properties one might also wish to try and enforce, for instance, interpretability of the network or robustness to perturbations of the inputs.

The purpose of this section is to justify the mathematical problem we formalise in Section 3.2 and then study in Section 3.3. In Sections 3.1.1 and 3.1.2 we argue in favour of computing distributed, hierarchical representations in which the underlying independent features and causes of the data are distinct and untangled. Unfortunately, these properties are not well defined in a mathematical sense. However, we propose that a flavour of them can be captured by using multilayer neural networks with a sparse prior on the activations at each layer. In order to motivate the DCSC model defined in section 3.2, in section 3.1.3 we review the reversible network model, and argue its value from the perspective that if the features learnt during training are meaningful, then it should be possible to use them to reconstruct at least the salient aspects of the data for the task at hand. This perspective suggests analysing the forward pass as the decoder half of a weight tied encoder-decoder neural network pair.

3.1.1 Distributed representations

The key idea behind distributed representations [54, 85] is that one can encode a set of objects using subsets of some common, underlying set. To illustrate this point through a simple example, consider the task of labelling 2^n different objects, clearly a binary encoding is far more efficient than a one hot encoding. More generally, using distributed representations it is possible to distinctly represent $\mathcal{O}(n^d)$ different classes using only $\mathcal{O}(nd)$ model parameters. The efficiency of algorithms which compute distributed representations arises from the fact that the model parameters are shared across different classes and objects. In the non-distributed setting this is not true, for instance, algorithms like k means assign a unique set of parameters to each class. Distributed representations are also related to the concept of causal representations, which is the the subject of the next section, to be clear and returning to our previous example the n elements of a binary vector can be interpreted as representing the common underlying features or causes which, in combination, help us discriminate between the 2^n different objects. As a result, distributed representations also provide a natural framework for generalisation.

Unfortunately distributed is not a well-defined mathematical property, at least in this context. However, empirical evidence indicates that the representations generated by a multilayer neural network capture many of the qualities of such representations. For instance, Yosinski et al [125] showed that the nodes in a convolutional layer activate strongly for specific and distinct features in images. Furthermore, nodes in layers closer to the input were shown to activate for low level features, such as edges and textures, while nodes in deeper layers activate for higher level concepts, such as different types of bird, animal or vehicle. These high level features therefore are constructed on a hierarchy of low level features, with sets of edges producing local shapes and groups of local shapes forming more recognisable objects.

3.1.2 Causal, independent and sparse representations

An old, popular and natural hypothesis for a good representation is one in which the underlying and independent causes of the data are separated and disentangled [54]. There are a number of obvious motivations for why we might want to compute such representations: in the unsupervised setting they afford interpretability, while in the supervised setting they are expected to improve accuracy and robustness. To illustrate this point through another example, it seems natural to assume that an algorithm which predicts sun screen sales using ice cream sales will be less accurate and robust than one using local weather data. Unfortunately, causal, independent representations are not something we can prescribe or encourage. As in the distributed case we settle for a proxy: just as multilayer networks appear to generate distributed hierarchical representations, in this section we will argue that sparse representations, which are mathematically well defined, capture a flavour of causal, independent ones. It has been postulated that the combination of these, i.e., hierarchical, distributed and sparse, captures the generation processes behind much real world data, an idea that has been explored since the 1980s [63].

The motivation for obtaining a sparse representation follows from Occam's razor: amongst all equal and competing explanations for a phenomenon, choose the simplest one. In the context of deep learning, sparse representations are thought to encourage interpretability and information disentangling [13]. To motivate this perspective consider the role of individual or subsets of nodes in a network: if the output of a layer is densely activated for any input, then it is difficult to attribute particular features of the input to a single or subset of nodes. Indeed, in this circumstance it is very difficult if not impossible to understand the role of a node without also taking into account the activity of many other nodes. However, if the activations of nodes or

subsets of nodes are uncorrelated, then distinguishing the features of the input that they identify is far easier. For instance, in the unsupervised setting and working on MNIST, Ranzato et al [106] empirically studied sparse feature learning in Deep Belief Networks (DBNs), finding that the weights of deep nodes often act as templates for specific digits.

Another benefit of sparse activations in the context of deep learning is, relative to dense activations, that sparse representations can more flexibly balance dimensionality reduction with information preservation [13]. Given that the activation at each layer has a fixed dimension, it is thought to be advantageous to map onto a higher dimensional space and encourage sparsity, rather than map directly onto a lower dimensional space. To understand this intuition, suppose that a given layer is required to produce a binary output vector which can distinguish between m different inputs. If the dimension, n_l, of this binary vector satisfies $n_l < log_2(m)$ then this is not possible. A key challenge potentially is that the number of distinct inputs a given layer needs to distinguish between may not be known in advance. As a result, choosing an overly small n_l risks poor information propagation. On the other hand, dimensionality reduction is important so as to achieve invariance to certain nuisance variables which act as noise for the task at hand [83]. As such, there is a tension between reducing the dimension of the data in order to remove unwanted factors and noise, while still preserving the relevant information. A practical compromise is to choose n_l large while encouraging the activation to be as sparse as possible. This approach means that instead of mapping each data point to the same low dimensional subspace, data points are mapped onto a union of low dimensional subspaces. Additionally, these subspaces can vary in dimension and as a result may encourage a more efficient sized representation [51]. Furthermore, empirical evidence suggests that sparsity plays an important role in deep learning even when sparsity is not explicitly encouraged. As a first example, perhaps the most popular activation function used by practitioners today is ReLU, which is a sparsifying operator. Additionally, Arpit et al [7] showed, in the context of autoencoders, that many forms of regularisation and other types of activation function indirectly encourage sparsity.

3.1.3 Reversible networks as generative models

A natural way in which to ensure that the features learnt during training are meaningful, in the sense of allowing for generalisation beyond the training data set, is to test whether they can be used to generate new data points, or at least the aspects of them salient to the task at hand. Networks whose features can be used in such a

manner we will refer to as being reversible, rather than invertible, as in many cases reconstruction of the input is not the objective. This line of reasoning suggests the idea that neural networks, sharing the same parameters, can act both as encoders and decoders of data. We now survey some of the literature in this regard, which provides some precedence for the DCSC model proposed in section 3.2.

On the empirical side, Mahendran and Vedaldi [81] explored reversibility in neural networks in the context of processing image data. Their experiments highlight that the convolutional layers near the input of the network provide codes for near perfect reconstruction. However, the reconstructions from the fully connected layers and pooling operations encountered deeper in the network produce far more abstract reconstructions. There are numerous other empirical results that corroborate this finding; for instance, Zhang et al [128] showed that the features learnt by high-capacity neural networks preserve the input information, despite spatial invariance induced by pooling, while Dosovitskiy and Brox [38] trained up-convolutional networks to invert representations of images in CNNs.

On the theoretical side, Gilbert et al [48] proposed and analysed a particular model based compressed sensing problem in which the data is considered as generated from a sparse, linear combination of learnt filters, i.e., a dictionary. To make the connection with CNNs, and in a very similar vein to Papyan et al [96], this dictionary can be factored to omit a sequence of sparse coding problems which can be solved using one iteration of iterative hard thresholding (IHT) at each layer. Gilbert et al were able to derive an RIP condition for this model based compressed sensing problem, providing a theoretical bound for layerwise reconstruction. Arora et al [6] proposed an algorithm for learning a class of multi-layer networks in a layerwise fashion based on network reversibility. The authors proved that this algorithm can recover the generative model of the data, and hence the parameters of the network, in a layerwise, sequential manner so long as the weight matrices and representations at each layer are sufficiently sparse. As a final example, Giryes et al [49] proved that deep networks with random weights preserve the metric structure of the data as it propagates through the layers: this allows for stable recovery of the input data from the features calculated by the network. The metric preservation property of the network provides a formal relationship between the complexity of the input data and the size of the required training set. In particular, the authors proved that the recovery of the input data is possible if the size of the network output is proportional to the intrinsic dimension of the data at the input, meaning that the input data is compressible to a representation whose sparsity is equal to the dimension of the output.

3.2 Inference in deep learning as sequential sparse coding

To summarise the key takeaways of section 3.1, first we argued that sparse, deep neural network representations act as a good proxy for distributed and causal ones. We then argued that learnt features are meaningful if they can be used to construct at least the salient aspects of new data points. Guided by these ideas, we considered reversible neural networks and in particular the weight tied encoder-decoder pair, surveying a number of results providing both theoretical and empirical justification in its favour. In this section we show, in the situation where the decoder network is linear and the encoder network is equipped with a sparsifying activation function, that inference in deep learning can be interpreted as sequential sparse coding. To this end we first introduce the DCSC model, a variant of the DCP model presented in [96], and then present the key results from [96] most relevant to our analysis.

3.2.1 The Deep Convolutional Sparse Coding (DCSC) model

We now define the DCSC model, a particular instance of a weight tied encoder-decoder neural network pair based on the DCP model [96].

Definition 3.2.1 (DCSC data model)**.** *In order to define the DCSC encoder we introduce the following variables.*

- $\boldsymbol{A}^{(l)} \in \mathbb{R}^{n_{l-1}M \times n_l M}$ *is a deterministic convolutional matrix (see [96] and [95] for further details) which is circular, banded and created by shifting a local dictionary* $\boldsymbol{A}^{(l)}_{Local} \in \mathbb{R}^{m_l \times n_l}$ *across all spatial locations. At each layer we interpret* n_l *as the number of local filters and* m_l *as the dimension of each local filter. We further specify that* $n_0 := 1$, $m_l := n_{l-1}m_{l-1}$ *and for* $l \geq 2$ *there is a stride* $s_l = n_{l-1}$ *between each spatially shifted* $\boldsymbol{A}^{(l)}_{local}$*. The columns of* $\boldsymbol{A}^{(l)}$ *are assumed to have unit* ℓ_2 *norm and are denoted as* $\boldsymbol{A}^{(l)} = [\boldsymbol{a}^{(l)}_1 \ \boldsymbol{a}^{(l)}_2 \ ... \ \boldsymbol{a}^{(l)}_{n_l M}]$.

- $D^{(l)}$ *is a random, square, binary, diagonal matrix of size* $n_l M \times n_l M$ *whose diagonal entries are independent and identically distributed, taking values in* $\{-1, 1\}$ *each with probability* 0.5.

The DCSC encoder is a function $E_{DCSC} : \mathbb{R}^{n_L M} \to \mathbb{R}^M$ *parameterized by the forward pass of a linear neural network. The input to the encoder is denoted* $\mathbf{x}^{(L)} \in \mathbb{R}^{n_l M}$ *and*

the representations of $\mathbf{x}^{(L)}$ *generated at each layer of the DCSC encoder are given by the recurrence relation*

$$\mathbf{x}^{(l-1)} := \mathbf{A}^{(l)} D^{(l)} \mathbf{x}^{(l)} \ \ \forall \ \ l \in [L].$$

Given a DCSC encoder, the corresponding weight tied decoder function $D_{DCSC} : \mathbb{R}^M \to \mathbb{R}^{n_L M}$ *is parameterized by the forward pass of another neural network equipped with a nonlinear, sparsifying activation function. The input to the decoder is defined as*

$$\hat{\mathbf{x}}^{(0)} := E_{DCSC}(\mathbf{x}^{(L)}) + \mathbf{v}^{(0)} = \mathbf{x}^{(0)} + \mathbf{v}^{(0)},$$

where $\mathbf{v}^{(0)} \in \mathbb{R}^M$ *denotes the model noise. The representations of* $\hat{\mathbf{x}}^{(0)}$ *generated at each layer of the decoder are defined recursively for* $l \in [L]$ *as*

$$\hat{\mathbf{x}}^{(l)} := Proj_{|\operatorname{supp}(\cdot)|=k} \left((\boldsymbol{A}^{(l)} D^{(l)})^T \hat{\mathbf{x}}^{(l-1)} \right), \tag{3.1}$$

where the projection operator $Proj_{|\operatorname{supp}(\cdot)|=k}(\cdot)$ *keeps the* k *largest elements in terms of absolute value unchanged and sets all other elements to zero. We further define the representation error between the encoder and decoder at the* l*th layer as*

$$\mathbf{v}^{(l)} := \mathbf{x}^{(l)} - \hat{\mathbf{x}}^{(l)}.$$

In regard to Definition 3.2.1 a few remarks are in order. First, by replacing $D^{(l)}$ with the identity matrix at each layer $l \in [L]$, then the DCP_λ and DCP_λ^ϵ models presented in [96] are recovered. Second, by substituting $Proj_{|\operatorname{supp}(\cdot)|=k}(\cdot)$ with a ReLU operator we obtain the standard forward pass algorithm across a ReLU layer of a neural network with $(\mathbf{A}^{(l)}\mathbf{D}^{(l)})^T$ the weight matrix between the $l-1$th and lth layers. One can interpret $Proj_{|\operatorname{supp}(\cdot)|=k}(\cdot)$ as model for a family of sparsifying operators, of which ReLU and the soft and hard thresholding operators are examples. Indeed, in almost any practical circumstance, by adjusting the bias at each layer appropriately it should be clear how each specific sparsifying activation function can implement a projection onto the k largest entries of the argument vector in question.

There are numerous questions one might ask concerning the DCSC model, for instance, what conditions are sufficient and or necessary for recovery in the noiseless case, or to ensure$\|\mathbf{x}^{(l)} - \hat{\mathbf{x}}^{(l)}\|_2 \leq \epsilon$ for all $l \in [L]$ and some $\epsilon \in \mathbb{R}_{>0}$. These are some of the questions studied in [96] in the case where $D^{(l)}$ is the identity matrix. Also covered in this work is a study on the recovery of the support at each layer, it is this notion of recovery that we will focus on.

Definition 3.2.2 (Activation pathway)**.** *The activation pathway associated with* $\mathbf{x}^{(L)} \in \mathbb{R}^{n_L M}$ *is the sequence of supports* $\{\text{supp}(\mathbf{x}^{(l)})\}_{l=1}^{L}$*. We say that the activation pathway of* $\mathbf{x}^{(L)}$ *is recovered up to layer* l *iff* $\text{supp}(\hat{\mathbf{x}}^{(k)}) = \text{supp}(\mathbf{x}^{(k)})$ *for all* $k \in [l]$*. An activation pathway is recovered iff it is recovered at all layers, i.e, up to layer* L*.*

To motivate why we are interested in the recovery of activation pathways we note, as discussed previously, that we are not necessarily interested in achieving perfect reconstruction. Instead, we wish to ensure that a trained network is able to identify the key features of the observed data salient for the task at hand. If we consider the encoder as generating these salient features, then recovery of an activation pathway implies by construction that the decoder identifies the presence of these salient features in the data.

3.2.2 Uniform guarantees for activation pathway recovery

In [96], and under the assumption that $D^{(l)}$ is the identity matrix for all $l \in [L]$, Papyan, Romano, and Elad studied the $\{\hat{\mathbf{x}}^{(l)}\}_{l=1}^{L}$ obtained by the DCSC encoder in relation to the $\{\mathbf{x}^{(l)}\}_{l=1}^{L}$ generated by the encoder. In particular, they proved various forms of recovery guarantees under sparsity constraints on the encoder representations. Their analysis relies heavily on the notion of the coherence of a dictionary,

$$\mu(\mathbf{A}) := \max_{i \neq j} |\langle \mathbf{a}_i, \mathbf{a}_j \rangle|, \tag{3.2}$$

where $\mathbf{a}_i$ is the i^{th} column of $\mathbf{A}$. One of the main technical innovations in [96] was the derivation of traditional sparse approximation bounds in the convolutional, multilayer setting. To this end they introduced the following novel local sparsity measures, based on the banded, circular structure of each $\mathbf{A}^{(l)}$, and used them to ameliorate the limited lower bound on (3.2).

- $\|\mathbf{x}\|_{\alpha,\infty}^{P^{(l)}} := \max_{i \in n_l M} \|P^{(l)}(i)\mathbf{x}\|_\alpha$ where $P^{(l)}(i)$, the patch operator at the lth layer, is an $n_l M \times n_l M$ diagonal, binary matrix with exactly m_{l+1} consecutive nonzeros starting at row i with wraparound. To be clear, if $i + m_l - 1 \leq n_l M$ then $P_{j,j}^{(l)}(i) = 1$ iff $i \leq j \leq i + m_l - 1$, if $i + m_l - 1 > n_l M$ then $P_{j,j}^{(l)}(i) = 1$ iff $i \leq j \leq n_l M$ or $1 \leq j \leq m_l - 1 - (n_l M - i)$ (see [96] for further details). In this paper we will only consider $\alpha \in \{0, 2\}$, hence $\|\cdot\|_\alpha$ refers to the euclidean norm when $\alpha = 2$, and a function counting the number of non-zeros in the argument vector when $\alpha = 0$.

- $\|\mathbf{x}\|_{\alpha,\infty}^{Q^{(l)}} := \max_{i \in n_l M} \|Q_i^{(l)}\mathbf{x}\|_\alpha$ where $Q^{(l)}(i)$, the stripe operator, is an $n_l M \times n_l M$ diagonal, binary matrix with exactly $\lfloor((2(m_l/s_l)-1)n_l)\rfloor$ consecutive nonzeros starting at row i with wraparound. To be clear, if $i + \lfloor((2(m_l/s_l)-1)n_l)\rfloor - 1 \leq n_l M$ then $Q_{j,j}^{(l)}(i) = 1$ iff $i \leq j \leq i + \lfloor((2(m_l/s_l)-1)n_l)\rfloor - 1$, if $i + \lfloor((2(m_l/s_l)-1)n_l)\rfloor - 1 > n_l M$ then $Q_{j,j}^{(l)}(i) = 1$ iff $i \leq j \leq n_l M$ or $1 \leq j \leq \lfloor((2(m_l/s_l)-1)n_l)\rfloor - 1 - (n_l M - i)$ (again see [96] for further details). A stripe of $\mathbf{x}^{(l)}$ then is the sparse code associated with a particular patch of $\mathbf{x}^{(l-1)}$. As before we will only consider $\alpha \in \{0, 2\}$.

Papyan et al's analysis is wide-ranging, including sparsity conditions under which the representations generated by the decoder are unique. They also consider a variety of thresholding operators such as soft and hard thresholding as well as more advanced algorithms to compute $\hat{\mathbf{x}}^{(l)}$ from $\mathbf{A}^{(l)}$ and $\hat{\mathbf{x}}^{(l-1)}$. We focus only on the derived uniform bound concerning the recovery of activation pathways. Assume that $\text{supp}(\hat{\mathbf{x}}^{(k)}) = \text{supp}(\mathbf{x}^{(k)})$ for all $k < l$, and that the cardinality of $\text{supp}(\mathbf{x}^{(l)})$ is known for all $l \in [L]$ by the decoder network. Papyan et al proved that as long as

$$\|\mathbf{x}^{(l)}\|_{0,\infty}^{Q^{(l)}} < \frac{1}{\mu^{(l)}|x_{max}^{(l)}|}\left(\frac{1}{2}|x_{min}^{(l)}| - \zeta_l\right) + \frac{1}{2}, \tag{3.3}$$

where $\zeta_l \geq \|\mathbf{v}^{(l)}\|_{2,\infty}^{P^{(l)}}$ is an upper bound on the patch error at the lth layer, $\mu_l := \mu(\mathbf{A}^{(l)})$ and $|x_{min}^{(l)}|$ and $|x_{max}^{(l)}|$ are the smallest and largest non-zero entries of $\mathbf{x}^{(l)}$ respectively, then $\text{supp}(\hat{\mathbf{x}}^{(l)}) = \text{supp}(\mathbf{x}^{(l)})$.

3.3 Recovery of denser activation pathways

Notable in the sparsity bound (3.3) is the presence of μ_l, which allows for a nontrivial stripe sparsity. Bounds of the form (3.3) are prevalent in the theory of sparse approximation, see for instance [47, Chapter 5], where it is known [105] for a generic matrix $\mathbf{B} \in \mathbb{R}^{m \times \gamma m}$ that $\mu(\mathbf{B}) > m^{-1/2}\sqrt{1-\gamma^{-1}}$. This is colloquially referred to as the square-root bottleneck in that $\mu^{-1} \sim m^{1/2}$. In many applications, e.g. imaging, typically m_l is not more than 7^2 and n_l is approximately $2m_l$. In addition, guaranteeing the recovery of denser activations is also made challenging due to the fact that $\mathbf{A}^{(l)}$ is a convolutional matrix. This structure can result in a large mutual coherence if the stride between shifted versions of the local dictionary $\mathbf{A}_{Local}^{(l)}$ is small. As a result, the proportionality of μ_l to the signal complexity, measured in terms of the sparsity, limits the ability of this prior work to provide guarantees in many practical situations.

It is well known from the work of Schnass and Vandergheynst [110] that, in the single layer context, if one introduces a randomised sign pattern then a Rademacher concentration inequality can be used to derive bounds demonstrating that the recovery of activations is typically possible even when the sparsity constraint is relaxed to depend on μ_l^{-2}. Our main contribution is to extend the techniques used in [110] to the multi-layer setting of [96], which explains and motivates the introduction of the random diagonal matrix $D^{(l)}$ at each layer of the DCSC model. This matrix applies a random sign pattern to the columns of $\mathbf{A}^{(l)}$ and although this matrix is primarily an artefact necessary for our analysis, it is interesting to note its connection with dropout. Dropout is a technique commonly used when training DCNNs in which a random set of nodes (or columns of the weight matrix) are ignored in every update of the weights. Indeed, one can tentatively interpret $D^{(l)}$ as a special form of dropout, which selects either the positive or negative signed column from a wider dictionary that contains both. Under this adaption, and recalling that $|x_{min}^{(l)}|$ and $|x_{max}^{(l)}|$ are the smallest and largest non-zeros in terms of absolute value of $\mathbf{x}^{(l)}$, then we are able to provide Theorem 3.3.1.

Theorem 3.3.1. *Under the DCSC model, for each $l \in [L]$ let $S_l \in \mathbb{N}$ be an upper bound on the stripe sparsity, $\|\mathbf{x}^{(l)}\|_{0,\infty}^{Q^{(l)}} \leq S_l$. Assume that the model noise $\mathbf{v}^{(0)}$ is such that $\operatorname{supp}(\hat{\mathbf{x}}^{(0)}) = \operatorname{supp}(\mathbf{x}^{(0)})$. Furthermore, for each $l \in [L] \cup \{0\}$ let $\zeta_l \in \mathbb{R}_{>0}$ be an upper bound on the patch error, $\|\mathbf{v}^{(l)}\|_{2,\infty}^{P^{(l)}} \leq \zeta_l$. Then the probability that the activation pathway of $\mathbf{x}^{(L)}$ is recovered is at least*

$$1 - 2M \sum_{l=1}^{L} n_l \exp\left(-\frac{|x_{min}^{(l)}|^2}{8\left(|x_{max}^{(l)}|^2 \mu_l^2 S_l + \zeta_{l-1}^2\right)} \right).$$

In addition, assuming that $\operatorname{supp}(\hat{\mathbf{x}}^{(l)}) = \operatorname{supp}(\mathbf{x}^{(l)})$ and defining $\zeta_0 := ||\mathbf{v}^{(0)}||_{2,\infty}^{P^{(l)}}$, then

$$\zeta_l = \sqrt{\|\hat{\boldsymbol{x}}^{(l)}\|_{0,\infty}^{P^{(l)}}} \left(\mu_l (S_l - 1) |x_{max}^{(l)}| + \zeta_{l-1} \right)$$

is a valid upper bound on the error $||\mathbf{v}^{(l)}||_{2,\infty}^{P^{(l)}} \leq \zeta_l$ for each layer $l \in [L]$.

A key implication of Theorem 3.3.1 is that the bound on the density of nonzeros scales proportional to μ_l^{-2} across a given layer rather than μ_l^{-1}. Assume that $\operatorname{supp}(\hat{\mathbf{x}}^{(k)}) = \operatorname{supp}(\mathbf{x}^{(k)})$ for all $k < l$, and that the cardinality of $\operatorname{supp}(\mathbf{x}^{(l)})$ is known for all $l \in [L]$ by the decoder network. With $\delta \in \left(2n_l M \exp\left(-\frac{|x_{min}^{(l)}|^2}{8\left(|x_{max}^{(l)}|^2 \mu_l^2 S_l + \zeta_{l-1}^2\right)} \right), 1 \right]$ (we refer the reader to Lemma 3.3.2 for details) then as long as

$$S_l \leq \left(\frac{|x_{min}^{(l)}|^2}{8|x_{max}^{(l)}|^2 \ln\left(\frac{2Mn_l}{\delta}\right)} - \frac{\zeta_{l-1}^2}{|x_{max}^{(l)}|^2} \right) \mu_l^{-2} \tag{3.4}$$

then the probability that the activation pattern at the lth layer is recovered is at least $1-\delta$.

We develop a proof of Theorem 3.3.1 using induction, analysing the probability that the forward pass fails to recover the activation pathway at an arbitrary layer $l \in [L]$ conditioned on recovery up to layer $l-1$. To this end we provide Lemma 3.3.2, which extends bounds provided in [110] to also include additive noise and the notion of local stripe sparsity.

Lemma 3.3.2. *Under the DCSC model, for each $l \in [L]$ let $S_l \in \mathbb{N}$ be an upper bound on the stripe sparsity $\|\mathbf{x}^{(l)}\|_{0,\infty}^{Q^{(l)}} \leq S_l$ of the encoder representation at the lth layer. Suppose for some $l \in [L]$ that $\operatorname{supp}(\hat{\mathbf{x}}^{(l-1)}) = \operatorname{supp}(\mathbf{x}^{(l-1)})$ and that $\zeta_{l-1} \geq \|\mathbf{v}^{(l-1)}\|_{2,\infty}^{P^{(l-1)}}$. Then the probability that $\operatorname{supp}(\hat{\mathbf{x}}^{(l)}) \neq \operatorname{supp}(\mathbf{x}^{(l)})$ is at most*

$$2n_l M \exp\left(-\frac{|x_{min}^{(l)}|^2}{8\left(|x_{max}^{(l)}|^2\mu_l^2 S_l + \zeta_{l-1}^2\right)}\right).$$

If $\operatorname{supp}(\hat{\mathbf{x}}^{(l)}) = \operatorname{supp}(\mathbf{x}^{(l)})$ then a valid upper bound for the patch error $||\mathbf{v}^{(l)}||_{2,\infty}^{P^{(l)}}$ is

$$\zeta_l = \sqrt{\|\hat{\boldsymbol{x}}^{(l)}\|_{0,\infty}^{P^{(l)}}}\left(\mu_l(S_l-1)|x_{max}^{(l)}| + \zeta_{l-1}\right).$$

Proof. First, and for typographical ease, we drop the l superscript on both $\mathbf{A}^{(l)}$ and $D^{(l)}$. We further denote the jth diagonal element of D as ε_j, and recall that these are mutually independent random variables with value either -1 or 1 both with probability 0.5. Furthermore, and again for notational ease, we define $\Lambda := \operatorname{supp}(\mathbf{x}^{(l)})$. A superscript bar will be used to denote the compliment of a set, for example $\bar{\Lambda} = [n_l M]\backslash\Lambda$. The event that the DCSC decoder recovers the support of the encoder representation, i.e., $\operatorname{supp}(\hat{\mathbf{x}}^{(l)}) = \operatorname{supp}(\mathbf{x}^{(l)})$, will be denoted $W^{(l)}$, and, in keeping with the other notational aspects just mentioned, $\bar{W}^{(l)}$ will denote the event that $\operatorname{supp}(\hat{\mathbf{x}}^{(l)}) \neq \operatorname{supp}(\mathbf{x}^{(l)})$. Finally, due to the presence of various superscripts, we will use $\langle\cdot,\cdot\rangle : \mathbb{R}^{n_l M} \times \mathbb{R}^{n_l M} \to \mathbb{R}$ to refer to the Euclidean inner product or dot product on $\mathbb{R}^{n_l M}$.

Considering Equation (3.2.1), then for the DCSC decoder to fail to recover the support of the encoder representation, there must exist a nonzero entry in the decoder representation which is not in the support of the encoder representation and whose magnitude is larger than at least one of the nonzeros in the encoder representation. This means that there exists some $i \in \Lambda$ and some $k \in \bar{\Lambda}$ such that

$$|\langle \varepsilon_i \mathbf{a}_i, \hat{\mathbf{x}}^{(l-1)}\rangle| < |\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle|.$$

This condition is equivalent to requiring

$$\min_{i\in\Lambda} |\langle \varepsilon_i \mathbf{a}_i, \hat{\mathbf{x}}^{(l-1)}\rangle| < \max_{k\in\bar{\Lambda}} |\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle|$$

and therefore

$$\mathbb{P}(\bar{W}^{(l)}) = \mathbb{P}(\min_{i\in\Lambda} |\langle \varepsilon_i \mathbf{a}_i, \hat{\mathbf{x}}^{(l-1)}\rangle| < \max_{k\in\bar{\Lambda}} |\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle|).$$

For an arbitrary $p \in \mathbb{R}$, if $\min_{i\in\Lambda} |\langle \varepsilon_i \mathbf{a}_i, \hat{\mathbf{x}}^{(l-1)}\rangle| < \max_{k\in\bar{\Lambda}} |\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle|$ holds true then the event $\{\min_{i\in\Lambda} |\langle \varepsilon_i \mathbf{a}_i, \hat{\mathbf{x}}^{(l-1)}\rangle| < p\} \cup \{\max_{k\in\bar{\Lambda}} |\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle| > p\}$ is also true. Applying the union bound it therefore follows that

$$\mathbb{P}(\bar{W}^{(l)}) \leq \mathbb{P}(\min_{i\in\Lambda} |\langle \varepsilon_i \mathbf{a}_i, \hat{\mathbf{x}}^{(l-1)}\rangle| < p) + \mathbb{P}(\max_{k\in\bar{\Lambda}} |\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle| > p). \tag{3.5}$$

We now provide bounds on each of the terms on the right hand side of the above inequality using the Rademacher concentration inequality stated in Theorem A.1.1 in Appendix A.1. Considering first the second term,

$$\begin{aligned}
P\left(\max_{k\in\bar{\Lambda}} |\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle| > p\right) &\leq \sum_{k\in\bar{\Lambda}} P\left(|\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle| > p\right) \\
&= \sum_{k\in\bar{\Lambda}} P\left(|\sum_{j\in\Lambda} \varepsilon'_j x_j^{(l)} \langle \mathbf{a}_k, \mathbf{a}_j\rangle + \varepsilon_k \langle \mathbf{a}_k, \mathbf{v}^{(l-1)}\rangle| > p\right) \\
&\leq 2\sum_{k\in\bar{\Lambda}} \exp\left(\frac{-p^2}{2\left(\sum_{j\in\Lambda\cap\Gamma} |x_j^{(l)}|^2 |\langle \mathbf{a}_k, \mathbf{a}_j\rangle|^2 + \zeta_{l-1}^2\right)}\right) \\
&\leq 2(n_l M - |\Lambda|) \exp\left(\frac{-p^2}{2\left(|x_{max}^{(l)}|^2 S_l \mu_l^2 + \zeta_{l-1}^2\right)}\right).
\end{aligned}$$

The first line and inequality arises from $\max_{k\in\bar{\Lambda}}\{|\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle| > p\}$ implying that $\cup_{k\in\bar{\Lambda}}\{|\langle \varepsilon_k \mathbf{a}_k, \hat{\mathbf{x}}^{(l-1)}\rangle|\} > p$ and then applying the union bound. The second line is an expansion of the inner product using $\hat{\mathbf{x}}^{l-1} = \mathbf{x}^{l-1} + \mathbf{v}^{(l-1)} = \mathbf{A}^{(l)} D^{(l)} \mathbf{x}^l + \mathbf{v}^{(l-1)}$. Here we also introduce a new Rademacher random variable $\varepsilon'_j := \varepsilon_j \varepsilon_k$ and note that the set of random variables $\left(\bigcup_{j\in n_l M}\{\varepsilon'_j\}\right) \cup \{\varepsilon_k\}$ are mutually independent. Moving from the second to the third line, we use Theorem A.1.1 given in Appendix A.1 and introduce the set Γ to denote the indices of columns of $\mathbf{A}^{(l)}$ which have a nonzero inner product with the column $\mathbf{a}_k$. Furthermore, as $\mathbf{a}_k$ has unit ℓ_2 norm and $|\operatorname{supp}(\mathbf{a}_k)| = m_l$, then $|\langle \mathbf{a}_i, \mathbf{v}^{(l-1)}\rangle| \leq \zeta_{l-1}^2$ by construction. The final line then follows from the fact that $|\langle a_k, a_j\rangle| \leq \mu_l^2$ for any $j \neq k$ and $|\Lambda \cap \Gamma| \leq S_l$, which in turn is a consequence of the assumption that $\|\mathbf{x}^{(l)}\|_{0,\infty}^{Q^{(l)}} \leq S_l$.

Turning our attention to bounding the probability of $\min_{i\in\Lambda}|\langle\varepsilon_i\mathbf{a}_i,\hat{\mathbf{x}}^{(l-1)}\rangle|<p$, we first expand the inner product as before and then use the triangle inequality to conclude that

$$|\langle\varepsilon_i\mathbf{a}_i,\hat{\mathbf{x}}^{(l-1)}\rangle|\geq|x_i|-\left|\sum_{j\in\Lambda,j\neq i}\varepsilon'_jx_j^{(l)}\langle\mathbf{a}_i,\mathbf{a}_j\rangle+\varepsilon_i\langle\mathbf{a}_i,\mathbf{v}^{(l-1)}\rangle\right|.$$

We are then able to bound the probability that $\min_{i\in\Lambda}|\langle\varepsilon_i\mathbf{a}_i,\hat{\mathbf{x}}^{(l-1)}\rangle|<p$ using the same steps as before for $\max_{k\in\bar{\Lambda}}\{|\langle\varepsilon_k\mathbf{a}_k,\hat{\mathbf{x}}^{(l-1)}\rangle|\}>p$.

$$\begin{aligned}\mathbb{P}(\min_{i\in\Lambda}|\langle\varepsilon_i\mathbf{a}_i,\hat{\mathbf{x}}^{(l-1)}\rangle|<p)&\leq P\left(\max_{i\in\Lambda}|\sum_{j\in\Lambda,j\neq i}\varepsilon'_jx_j^{(l)}\langle\mathbf{a}_i,\mathbf{a}_j\rangle+\varepsilon_i\langle\mathbf{a}_i,\mathbf{v}^{(l-1)}\rangle|>|x_{min}^{(l)}|-p\right)\\&\leq\sum_{i\in\Lambda}P\left(|\sum_{j\in\Lambda,j\neq i}\varepsilon'_jx_j^{(l)}\langle\mathbf{a}_i,\mathbf{a}_j\rangle+\varepsilon_i\langle\mathbf{a}_i,\mathbf{v}^{(l-1)}\rangle|>|x_{min}^{(l)}|-p\right)\\&\leq 2\sum_{i\in\Lambda}\exp\left(-\frac{(|x_{min}^{(l)}|-p)^2}{2\left(\sum_{j\in\Lambda\cap\Gamma/i}|x_j|^2|\langle\mathbf{a}_k,\mathbf{a}_j\rangle|^2+\zeta_{l-1}^2\right)}\right)\\&\leq 2|\Lambda|\exp\left(\frac{-(|x_{min}^{(l)}|-p)^2}{2\left(|x_{max}^{(l)}|^2S_l\mu_l^2+\zeta_{l-1}^2\right)}\right).\end{aligned}$$

The first line is a result of rearranging and bounding the expanded inner product, the subsequent lines then follow in the same manner as for $\max_{k\in\bar{\Lambda}}\{|\langle\varepsilon_k\mathbf{a}_k,\hat{\mathbf{x}}^{(l-1)}\rangle|\}>p$. Recalling that $p\in\mathbb{R}_{\geq 0}$ is arbitrary, then to recover the bound claimed we let $p=|x_{min}|/2$. Indeed, for this value of p it follows that

$$\begin{aligned}\mathbb{P}(\bar{W}^{(l)})&\leq 2(n_lM-|\Lambda|)\exp\left(\frac{-p^2}{2\left(|x_{max}^{(l)}|^2S_l\mu_l^2+\zeta_{l-1}^2\right)}\right)+2|\Lambda|\exp\left(\frac{-(|x_{min}^{(l)}|-p)^2}{2\left(|x_{max}^{(l)}|^2S_l\mu_l^2+\zeta_{l-1}^2\right)}\right)\\&=2n_lM\exp\left(-\frac{|x_{min}^{(l)}|^2}{8\left(|x_{max}^{(l)}|^2S_l\mu_l^2+\zeta_{l-1}^2\right)}.\right)\end{aligned}$$

In order to bound the patch error $\mathbf{v}^{(l)}$ under the assumption that the $\text{supp}(\mathbf{x})$ is recovered, we adopt the approach of Theorem 8 of [96]. First

$$\begin{aligned}\|\hat{\mathbf{x}}^{(l)}-\mathbf{x}^{(l)}\|_{2,\infty}^{P^{(l)}}&=\max_i\|P^{(l)}(i)\left(\mathbf{x}^{(l)}-\hat{\mathbf{x}}^{(l)}\right)\|_2\\&=\sqrt{\|\mathbf{x}^{(l)}\|_{0,\infty}^{P^{(l)}}}\left(\max_i\|P^{(l)}(i)\left(\mathbf{x}^{(l)}-\hat{\mathbf{x}}^{(l)}\right)\|_\infty\right)\\&\leq\sqrt{\|\mathbf{x}^{(l)}\|_{0,\infty}^{P^{(l)}}}\left(\|\mathbf{x}^{(l)}-\hat{\mathbf{x}}^{(l)}\|_\infty\right).\end{aligned}$$

The first equality follows from the definition of the patch norm $\|\cdot\|_{2,\infty}^{P^{(l)}}$. The second inequality arises from the fact that for any $\mathbf{z} \in \mathbb{R}^{n_l M}$, with k nonzeros, then $\|\mathbf{z}\|_2 \leq \sqrt{k}\|\mathbf{z}\|_\infty$. Given that we are assuming that $\text{supp}(\hat{\mathbf{x}}^{(l)}) = \text{supp}(\mathbf{x}^{(l)})$, then the inequality on the third line follows from the fact that the largest element in a vector is at least as large as the largest element of any subset of elements of that vector. In what follows subscript notation is used to indicate the subset of entries of a vector or columns of a matrix in an index set. As $\|\mathbf{x}^{(l)} - \hat{\mathbf{x}}^{(l)}\|_\infty = \|\mathbf{x}_\Lambda^{(l)} - \hat{\mathbf{x}}_\Lambda^{(l)}\|_\infty$, and recalling that the ℓ_∞ matrix norm is the maximum row sum of the absolute elements of the matrix, then

$$\begin{aligned}
\|\mathbf{x}_\Lambda^{(l)} - \hat{\mathbf{x}}_\Lambda^{(l)}\|_\infty &= \|(\mathbf{A}_\Lambda D_\Lambda)^+(\mathbf{A}_\Lambda D_\Lambda)\mathbf{x}_\Lambda^{(l)} - (\mathbf{A}_\Lambda D_\Lambda)^T\hat{\mathbf{x}}^{(l-1)}\|_\infty \\
&= \|(\mathbf{I} - (\mathbf{A}_\Lambda D_\Lambda)^T(\mathbf{A}_\Lambda D_\Lambda))\mathbf{x}_\Lambda^{(l)} - (\mathbf{A}_\Lambda D_\Lambda)^T\mathbf{v}^{(l-1)}\|_\infty \\
&\leq \|(\mathbf{I} - \mathbf{A}_\Lambda^T\mathbf{A}_\Lambda)\|_\infty\|\mathbf{x}_\Lambda^{(l)}\|_\infty + \|\mathbf{A}_\Lambda^T\mathbf{v}^{(l-1)}\|_\infty \\
&\leq \mu_l(\|\mathbf{x}^{(l)}\|_{0,\infty}^{Q^{(l)}} - 1)|x_{max}^{(l)}| + \zeta_{l-1}.
\end{aligned}$$

On line one, $(\mathbf{A}_\Lambda D_\Lambda)^+$ denotes the Moore-Penrose inverse, the equality then follows from the fact that $\mathbf{I} = (\mathbf{A}_\Lambda D_\Lambda)^+(\mathbf{A}_\Lambda D_\Lambda)$ and from the definition of the sparse projection carried out by the forward pass of the decoder at each layer, Equation (3.2.1). The equality on line 2 is obtained by introducing a positive and negative $\mathbf{A}_\Lambda D_\Lambda \mathbf{x}_\Lambda^{(l)}$. The inequality on the third line is obtained by applying the triangle inequality and then using the submultiplicative property of the induced matrix norm. The fourth and final inequality follows as a result of the definition of the ℓ_∞ matrix norm. The diagonal elements of $\mathbf{I} - \mathbf{A}_\Lambda^T\mathbf{A}_\Lambda$ are all zero as $|\langle \mathbf{a}_i, \mathbf{a}_i\rangle| = 1$. Off of the diagonal, at most $\|\mathbf{x}^{(l)}\|_{0,\infty}^{Q^{(l)}} - 1$ entries are nonzero due to the convolutional structure of $\mathbf{A}$. In addition, as $|\langle \mathbf{a}_i, \mathbf{a}_j\rangle| \leq \mu_l$ it follows that

$$\|(\mathbf{I} - \mathbf{A}_\Lambda^T\mathbf{A}_\Lambda)\|_\infty \leq \mu_l(\|\mathbf{x}^{(l)}\|_{0,\infty}^{Q^{(l)}} - 1) \leq \mu_l S_l.$$

Finally, letting $\alpha := \min_l\{l \in [n_{l-1}M] : \ l \in \text{supp}(\mathbf{a}_i)\}$ and recalling that $\|\mathbf{a}_i\|_2 \leq 1$, applying the Cauchy-Schwarz inequality it follows that

$$\|\mathbf{A}_\Lambda^T\mathbf{v}^{(l-1)}\|_\infty = \max_i |\langle \mathbf{a}_i, \mathbf{v}^{(l-1)}\rangle| = \max_i |\langle \mathbf{a}_i, P^{(l-1)}(\alpha)\mathbf{v}^{(l-1)}\rangle| \leq \|P^{(l-1)}(\alpha)\mathbf{v}^{l-1}\|_2 \leq \zeta_{l-1}.$$

This concludes the proof of Lemma 3.3.2. □

We now proceed to prove Theorem 3.3.1.

Proof. With Lemma 3.3.2 in place then Theorem 3.3.1 can be proved by induction. For the sake of convenience we let

$$\gamma_l := 2Mn_l \exp\left(-\frac{|x_{min}^{(l)}|^2}{8\left(|x_{max}^{(l)}|^2\mu_l^2 S_l + \zeta_{l-1}^2\right)}\right).$$

Furthermore, and in keeping with our notation, let $Y^{(l)}$ and $\bar{Y}^{(l)}$ denote the events that the activation pathway of $\mathbf{x}^{(L)}$ is recovered and not recovered up to the lth layer respectively, and $W^{(l)}$ and $\bar{W}^{(l)}$ be the events that the support at the lth layer is correctly and not correctly recovered respectively.

The base case $l = 1$ follows by the construction of $\mathbf{v}^{(0)}$ and by direct application of Lemma 3.3.2. As a result $\mathbb{P}(\bar{Y}^{(1)}) = \mathbb{P}(\bar{W}^{(1)}) \leq \gamma_1$. While the bound in Lemma 3.3.2 was derived by conditioning on recovery at the previous layer, note that the bound still in fact applies if we condition on recovery across all preceding layers. Indeed, we could make such assumptions and still derive the same bound by using only the information concerning the layer immediately before the layer of interest. As a result

$$\mathbb{P}(\bar{W}^{(l)}|\cap_{k=1}^{l-1} W^{(k)}) = \mathbb{P}(\bar{W}^{(l)}|Y^{(l-1)}) \leq \gamma_l$$

for all $l \in [L]$. Assume now that the desired result holds true for the lth layer, meaning $\mathbb{P}(\bar{Y}^{(l)}) \leq \sum_{k=1}^{l} \gamma_k$. Considering $\bar{Y}_{l+1}$, then

$$\begin{aligned}
\mathbb{P}(\bar{Y}_{l+1}) &= \mathbb{P}(\bigcup_{k=1}^{l+1} \bar{W}^{(k)}) \\
&= \mathbb{P}(\bar{W}^{(l+1)} \cup \bar{Y^{(l)}}) \\
&= \mathbb{P}(\bar{Y}_l) + \mathbb{P}(\bar{W}^{(l+1)} \cap Y^{(l)}) \\
&= \mathbb{P}(\bar{Y^{(l)}}) + \mathbb{P}(\bar{W}^{(l+1)}|Y^{(l)})\mathbb{P}(Y^{(l)}) \\
&\leq \sum_{k=1}^{l} \gamma_k + \gamma_{l+1}\mathbb{P}(Y_l) \\
&\leq \sum_{k=1}^{l} \gamma_k + \gamma_{l+1} \\
&= \sum_{k=1}^{l+1} \gamma_k.
\end{aligned}$$

This proves the result holds for the $l + 1$th case, given that this and the base case hold true then all other cases must follow. Finally, the bound on the patch error at each layer follows immediately from Lemma 3.3.2. □

3.4 Concluding remarks

Modelling the forward pass algorithm as a sparse coding problem allows us to derive recovery guarantees, which ensure that the representations computed by the forward pass are meaningful and interpretable. Our contributions in this chapter are a) an approach to carrying out a probabilistic rather than worst case analysis for the recovery of activation pathways using the DCSC model, given in Definition 3.2.1, and b) Theorem 3.3.1, which extends the prior uniform bound in [96] to one which holds with high probability. The key benefit of this result is that the proportionality of the stripe sparsity bound in regard to the dictionary coherence improves from μ_l^{-1} to μ_l^{-2} at each layer. Assuming the weight matrices are suitably conditioned, then this indicates that the forward pass algorithm is likely to recover the latent representations generated by the encoder of the DCSC for a more complex, measured in terms of the number of nonzeros per stripe, family of signals than previously suggested. From a practical perspective, if sparse coding is an important factor explaining the efficacy of the forward pass of DCNNs, then explicitly encouraging weight matrices with low coherence during training could improve their performance.

Chapter 4

Designing activation functions for a better initialisation

In this chapter we continue our analysis of the role of the activation function in deep learning, albeit in a different context. In particular, we study and identify properties of activation functions which can be used in order to ameliorate certain problems at initialisation, thereby resulting in improved training outcomes. The structure of this chapter is as follows: in Section 4.1 we review the foundational results of prior works, identify two key problems at initialisation and summarise our contributions. In Section 4.2 we prove our main theoretical result, which highlights the benefit of deploying an activation function which has a scalable linear region around the origin. Finally, in Section 4.3, we investigate the implications of our theory for training deep networks in practice.

4.1 Principles for initialising deep networks

The prior works on which our results are based can be divided into two distinct themes, one concerned with the dynamics of the preactivation correlations in the forward pass [103, 111, 59], and the other dynamical isometry [100, 108, 101]. We review these two themes in detail in Sections 4.1.1 and 4.1.2 respectively, then in Section 4.1.3 we introduce and present the key contribution of this paper, Theorem 4.1.1. Note that in what follows $Z, Z_1, Z_2 \sim \mathcal{N}(0,1)$ are considered to be independent and identically distributed standard Gaussian random variables. Furthermore, the standard Gaussian measures in one and two dimensions are denoted by γ and $\gamma^{(2)}$ respectively.

4.1.1 Correlation dynamics in the forward pass

In [91] it was established that the outputs of certain random, feedforward, single hidden layer neural networks converge in distribution to centred Gaussian processes as the width of the hidden layer goes to infinity. More recently in [33, 79], this result was extended to random multilayer neural networks, with the preactivations at each layer being found to converge in distribution to centred Gaussian processes in the large width limit. The behaviour of a centred Gaussian process is fully described by its covariance matrix. In particular, for networks whose forward pass is described by (1.1), with the weights and biases at each layer $l \in [L]$ being mutually independent, centred Gaussian random variables with variances σ_w^2/N_{l-1} and σ_b^2 respectively, and assuming that the same activation function ϕ is deployed at each neuron, then the kernel used to compute the entries of the covariance matrix of the Gaussian process at the lth layer is defined by the following recurrence relation,

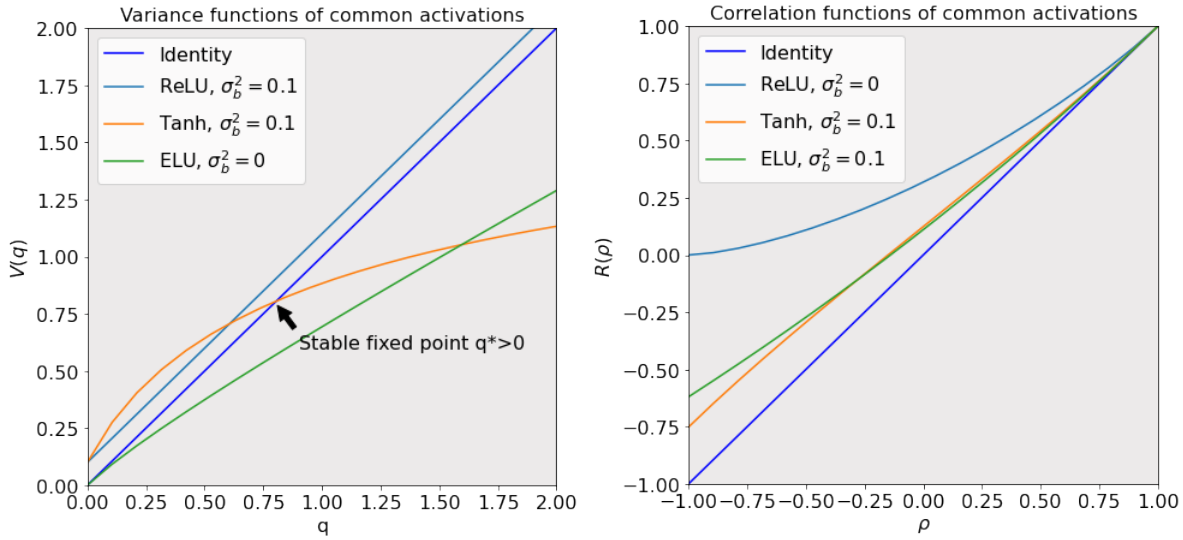
$$\kappa^{(l)}(\mathbf{x}^{(\alpha)}, \mathbf{x}^{(\beta)}) = \sigma_w^2 \mathbb{E}[\phi(z_i^{(l)}(\mathbf{x}^{(\alpha)}))\phi(z_i^{(l)}(\mathbf{x}^{(\beta)}))] + \sigma_b^2. \tag{4.1}$$

Note here that $i \in [N_l]$ can be any entry of the preactivation. This recurrence relation has been highlighted a number of times in a variety of contexts, the one most relevant for this paper being a mean field approximation used to study signal propagation in the forward pass [103, 111, 59]. In this work we do not focus on the correspondence between wide, random neural networks and Gaussian processes as in [33, 79]. Rather, as in [59, 103, 111], we adopt the Gaussian process or mean field approximation in order to better understand and analyse key statistics of the network at initialisation. In particular, and as highlighted in [103], in the infinite width limit one can interpret the variance of the preactivations of an input at a given layer, corresponding to (4.1) with $\alpha \neq \beta$, as the expected euclidean length of said input at said layer. Adopting the notation of [103], then the sequence of variances, or expected lengths, $(q_\alpha^{(l)})_{l=1}^L$, associated with an arbitrary input $\mathbf{x}^{(\alpha)}$, are generated by the following recurrence relation,

$$q_\alpha^{(l)} = V_\phi(q_\alpha^{(l-1)}) := \sigma_w^2 \int_{\mathbb{R}} \phi\left(\sqrt{q_\alpha^{(l-1)}} z\right)^2 d\gamma(z) + \sigma_b^2 = \sigma_w^2 \mathbb{E}\left[\phi\left(\sqrt{q_\alpha^{(l-1)}} Z\right)^2\right] + \sigma_b^2 \tag{4.2}$$

for $l \in [L]$ with $q_\alpha^{(0)} := ||\mathbf{x}^{(\alpha)}||_2^2$. As demonstrated in Figure 4.1(a), analysis of the variance function $V_\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ provides insight into the impact of $(\phi, \sigma_w^2, \sigma_b^2)$ on the dynamics of the expected length of the preactivations with depth. Indeed, unless $(\phi, \sigma_w^2, \sigma_b^2)$ is chosen appropriately then $q_\alpha^{(l)}$ can either rapidly converge towards zero or

diverge. This problem of vanishing or exploding activation lengths can be mitigated by choosing $(\phi, \sigma_w^2, \sigma_b^2)$ in order that there exists a stable fixed point $q^* = V_\phi(q^*)$ for which $q_\alpha^{(l)} \to q^*$ for all inputs $\mathbf{x}^{(\alpha)}$, see for example Tanh with $\sigma_b^2 = 0.1$ in Figure 4.1(a). We remark that this problem was also studied in [58] in the context of finite width deep ReLU networks.



(a) (b)

Figure 4.1: (a) Variance functions for ReLU, Tanh and ELU with $\sigma_b^2 \in \{0, 0.1\}$. With $\sigma_b^2 = 0.1$ the variance function associated with ReLU has no fixed point and $q^{(l)} \to \infty$ from any initial variance $q^{(0)} \in \mathbb{R}_{\geq 0}$. With $\sigma_b^2 = 0$ the variance function associated with ELU has a unique and marginally stable point $q^* = 0$, therefore $q^{(l)} \to 0$ from any initial variance $q^{(0)} \in \mathbb{R}_{\geq 0}$. Finally, with $\sigma_b^2 = 0.1$, the variance function associated with Tanh has a unique and stable fixed point $q^* > 0$, therefore $q^{(l)} \to q^*$ from any initial variance $q^{(0)} \in \mathbb{R}_{\geq 0}$. (b) Correlation functions associated with Relu, Tanh and ELU with $\sigma_b^2 \in \{0, 0.1\}$. For ReLU, and as illustrated by (a), for there to exist a fixed point of the variance function it is necessary that $\sigma_b^2 = 0$. Note that without a fixed point q^* it is not possible to define a correlation function which is fixed with depth. Relative to the other activation functions considered, the correlation function of ReLU is further from the identity function, and as a result correlations converge faster with the depth. Note that in both (a) and (b) σ_w^2 is chosen so that $\chi_1 = 1$, see (4.6).

In [103] the covariance and correlation between preactivations of different inputs was also analysed and a recurrence relation for the covariance derived, which is equivalent to (4.1) with $\alpha \neq \beta$. In the infinite width limit the covariance can be interpreted as the inner product between the preactivations of two inputs, and the correlation as the angle. The evolution of these quantities with depth has important implications for the performance of the network at initialisation as well as subsequent training.

Denoting the covariance between the preactivations of $\mathbf{x}^{(\alpha)}$ and $\mathbf{x}^{(\beta)}$ at the lth layer as $q^{(l)}_{\alpha\beta}$, with respective variances $q^{(l)}_{\alpha}$ and $q^{(l)}_{\beta}$, then it was shown in [103] that

$$\begin{aligned} q^{(l)}_{\alpha\beta} &= \sigma_w^2 \int_{\mathbb{R}^2} \phi(u_1)\phi(u_2) d\gamma^{(2)}(z_1, z_2) + \sigma_b^2, \\ u_1 &:= \sqrt{q^{(l-1)}_{\alpha}} z_1, \; u_2 := \sqrt{q^{(l-1)}_{\beta}} (\rho^{(l-1)}_{\alpha\beta} z_1 + \sqrt{1 - (\rho^{(l-1)}_{\alpha\beta})^2} z_2), \end{aligned} \tag{4.3}$$

where $\rho^{(l-1)}_{\alpha\beta} = q^{(l-1)}_{\alpha\beta} / (q^{(l-1)}_{\alpha} q^{(l-1)}_{\beta})$ is the correlation at the previous layer. If the variance function defined in (4.2) has a fixed point $q^* > 0$ then an additional key benefit beyond that already discussed is that the analysis of the evolution of the covariance and correlation with depth can be simplified. In particular, assuming that the inputs are normalised so that $q^{(0)}_{\alpha} = q^{(0)}_{\beta} = q^* > 0$, then $q^{(l)}_{\alpha} = q^{(l)}_{\beta} = q^*$ for all $l \in [L]$. In this setting the sequence of correlations $(\rho^{(l)}_{\alpha\beta})_{l=1}^{L}$ are generated by the following recurrence relation,

$$\begin{aligned} \rho^{(l)}_{\alpha\beta} = R_{\phi,q^*}(\rho^{(l-1)}_{\alpha\beta}) := \frac{q^{(l)}_{\alpha\beta}}{q^*} &= \frac{\sigma_w^2}{q^*} \int_{\mathbb{R}} \phi(u_1)\phi(u_2) d\gamma^{(2)}(z_1, z_2) + \frac{\sigma_b^2}{q^*} \\ &= \frac{\sigma_w^2}{q^*} \mathbb{E}[\phi(U_1)\phi(U_2)] + \frac{\sigma_b^2}{q^*}. \end{aligned} \tag{4.4}$$

Here we refer to $R_{\phi,q^*} : [-1, 1] \to [-1, 1]$ as the correlation function, and $U_1 := \sqrt{q^*} Z_1$, $U_2 := \sqrt{q^*}(\rho Z_1 + \sqrt{1 - \rho^2} Z_2)$, where ρ is the input argument $R_{\phi,q^*}(\rho)$, are dependent Gaussian random variables with $U_1, U_2 \sim \mathcal{N}(0, q^*)$.

Analysing the univariate recurrence relation $\rho^{(l)} = R_{\phi,q^*}(\rho^{(l-1)})$ allows for the identification of both depth limits, beyond which information cannot propagate, as well as issues around stability, in a manner analogous to that of the dynamical systems perspective given in [56]. In order to appreciate these points, observe, as in [103], that by construction as long as the correlation function is well defined on $[-1, 1]$ then one is always a fixed point,

$$R_{\phi,q^*}(1) = \frac{\sigma_w^2}{q^*} \mathbb{E}[\phi(U_1)^2] + \frac{\sigma_b^2}{q^*} = \frac{V(q^*)}{q^*} = 1.$$

The general shape of the correlation function, as well as the stability of the fixed point at one and the existence of other fixed points, is determined by the choice of $(\phi, \sigma_w^2, \sigma_b^2)$. This can be observed in Figure 4.1(b), in which the correlation functions of three commonly used activation functions are plotted side by side for comparison. Figure 4.2(a) shows the correlation function for three different values of σ_w^2 with $\sigma_b^2 = 0.1$ and $\phi(z) = \text{htanh}(z)$ fixed. For $\sigma_w^2 = 0.63$ and $\sigma_w^2 = 1.26$ it is clear from Figure 4.2(a) that $\rho^* = 1$ is the only positive fixed point. Furthermore, for these two

choices of σ_w^2 it is also clear, by inspection, that $\rho^* = 1$ is a stable and marginally stable fixed point respectively, and that from any initial correlation the sequence of correlations $\rho^{(l)} \to 1$ with depth. As a result, in the infinite width limit any pair of inputs, no matter how large the angle between them is, are mapped to the same point by the network asymptotically with depth. However, for $\sigma_w^2 = 0.63$ a new, stable fixed point $\rho^* < 1$ is introduced. In this case, then for any initial correlation the sequence $\rho^{(l)}$ converges to the limit $\rho^* < 1$. Therefore, in the infinite width limit any pair of inputs, no matter how small the initial angle between them is, are mapped to different points by the network asymptotically with depth. Figure 4.2(a) therefore illustrates the existence of two very distinct regimes for a given choice of activation function, one ordered, in which all inputs are mapped asymptotically to the same output, and the other chaotic, in which even arbitrarily small perturbations of an input lead to a different output. Figure 4.2(b) illustrates the limiting ρ^* throughout the (σ_w^2, σ_b^2) plane for $\phi = \text{htanh}(\cdot)$.

In order to understand and separate these two regimes [103, 111] studied the slope of $R_{\phi,q^*}(\rho)$ at $\rho = 1$: for a sufficiently smooth activation function ϕ the derivative is given by

$$R'_{\phi,q^*}(\rho) = \sigma_w^2 \mathbb{E}[\phi'(U_1)\phi'(U_2)], \tag{4.5}$$

the slope at $\rho = 1$ is therefore

$$\chi_1 := R'_{\phi,q^*}(1) = \sigma_w^2 \mathbb{E}[\phi'(U_1)^2]. \tag{4.6}$$

The set of pairs $(\sigma_w^2, \sigma_b^2) \in \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ such that $\chi_1 = 1$ is colloquially referred to in the literature as the edge of chaos (EOC) and in Figure 4.2(c) this set, or curve, is plotted for three different activation functions. In [111] it was proved that if (σ_w^2, σ_b^2) does not lie on the EOC then the sequences generated by the variance and correlation functions, (4.2) and (4.4) respectively, approach their respective fixed points asymptotically exponentially fast. As a result, the EOC curve can be interpreted as a transition boundary between order, corresponding to $\chi_1 < 1$, in which pairwise correlations converge asymptotically exponentially fast to 1 with depth, and chaos, corresponding to $\chi_1 > 1$, in which even pairwise input correlations that are arbitrarily close to 1 diverge asymptotically exponentially fast with depth. The dynamics of the preactivation correlations can also be understood and interpreted in terms of a network's sensitivity to perturbations of the input, with networks initialised in the chaotic regime carrying the risk of being overly sensitive, and those initialised in the ordered regime being insensitive. These observations are also of practical relevance

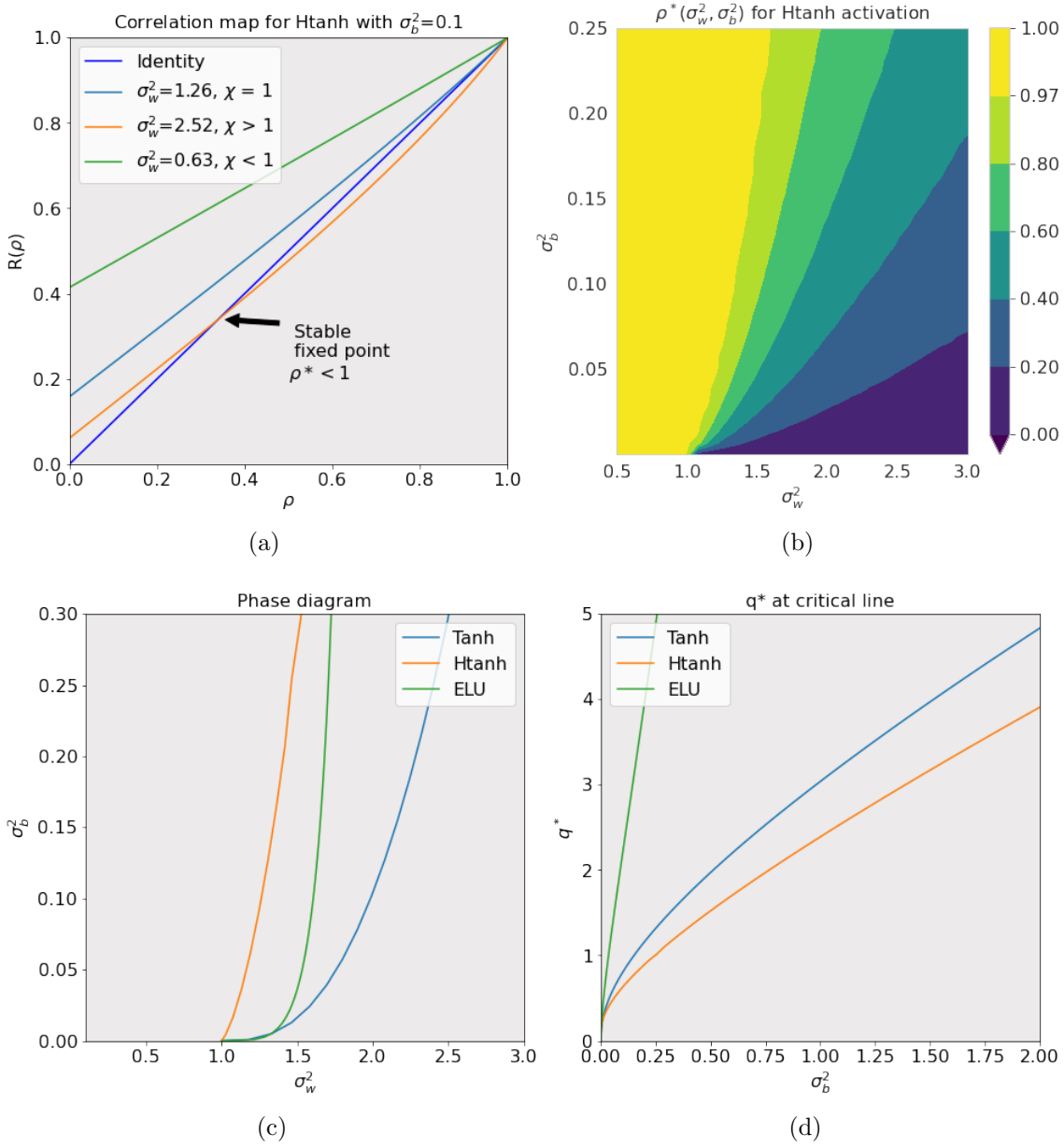


Figure 4.2: (a) Correlation map for htanh activation function, plotted on the interval $[0, 1]$ to better display fixed point behaviour (there are no fixed points in the interval $[-1, 0]$). When $\chi \leq 1$ then one is the unique fixed point of the correlation function. Furthermore, for $\chi < 1$ then one is a stable fixed point, and for $\chi = 1$ then one is a marginally stable fixed point. When $\chi > 1$ then although one is still a fixed point it is unstable, and a new stable fixed point less than one is introduced. (b) Stable fixed point ρ^* as a function of (σ_w^2, σ_b^2) for the tanh activation function. (c) The EOC critical line $\chi = 1$ for different activation functions. (d) σ_w^2 as a function of q^*, as $\sigma_b^2 \to 0$ then $q^* \to 0$.

for training, indeed experimental evidence [111] illustrates that initialising closer to the EOC consistently results in reduced training times.

To recap the discussion presented so far, rapid convergence in correlations to a fixed point appears to result in poor training outcomes, with the network being either highly sensitive or insensitive to perturbations of the input. To avoid asymptotically exponentially fast convergence it is necessary to choose (σ_w^2, σ_b^2) so that $\chi_1 = 1$. The specific asymptotic rate however depends on the activation function deployed. In [59], and under the assumption that initialisation is on the EOC, the asymptotic convergence rate of the sequence of correlations for different families of activation functions was analysed. A key finding of this work is that while ReLU like activation functions have asymptotic convergence $|1 - \rho^{(l)}| = \mathcal{O}(l^{-2})$, a broad class of smooth activation functions, including, tanh, elu and swish, have convergence $|1 - \rho^{(l)}| = \mathcal{O}(l^{-1})$. However, it seems highly reasonable that avoiding fast non-asymptotic, i.e., away for $\rho^* = 1$, convergence of correlations is also potentially of value. Figure 4.1(b) illustrates, assuming initialisation on the EOC, that certain combinations of (ϕ, σ_b^2) can result in a correlation function which is closer to the identity function. This in turn implies a slower convergence of the correlation sequence throughout the layers, rather than just at the depth limit. This idea was promoted in [59, Appendix B.2], where it was suggested that one should choose a small σ_b^2 and an activation function which is approximately linear, for example, one which is the weighted sum of a linear and nonlinear activation function.

4.1.2 Dynamical isometry

We now turn our attention to reviewing the findings of the literature concerning the propagation of $\delta^{(l)}$, see (1.3), in the backwards pass. Analogous to the recurrence relation for the variance function (4.2) in the forward pass, in [111] a mean field approximation for $\delta^{(l)}$ was developed. Specifically, the recurrence relation for the variance, $\tilde{q}^l := \mathbb{E}[(\delta_i^l)^2]$, of entries of the error vectors δ_i^l was shown to be

$$\tilde{q}^l = \frac{N^{l+1}}{N^l} \tilde{q}^{l+1} \chi_1 \tag{4.7}$$

with χ_1 defined as in (4.6). Although certain assumptions needed for this approximation are undesirable, notably the weights used during forward propagation are drawn independently from the weights used in backpropagation, empirical evidence indicates that it is nonetheless a useful model [111]. In particular, unlike the analysis of the gradients used to update the network parameters. The reappearance of χ_1 in (4.7) shows the benefit of selecting $(\phi, \sigma_w^2, \sigma_b^2)$ such that $\chi = 1$, as this ensures, at least in expectation, that the magnitudes of the error vectors are stable during the backward pass, i.e., they neither converge to zero or diverge.

Unfortunately, initialising on the EOC suffices only to ensure that vanishing and exploding gradients are avoided on average, and in practice does not guarantee good training performance. In particular, if, as it propagates backwards, an error vector is projected onto a smaller and smaller subspace then the directions in which the parameters in the lower layers can be updated becomes limited, thereby resulting in model degeneracy [101, 108]. In order to improve training, in [101, 108] the authors proposed that the product of Jacobians at each layer should act as an isometry on as large a subspace as possible, implying that the singular values of the input-output Jacobian at any layer should concentrate around one. As in [100, 101], we denote the input-output Jacobian of the network as $\mathbf{J} : \mathbb{R}^{N_0} \to \in \mathbb{R}^{N_0 \times N_L}$,

$$\mathbf{J}(\mathbf{x}) = \prod_{l=1}^{L} \mathbf{D}^{(l)}(\mathbf{x})\mathbf{W}^{(l)}. \tag{4.8}$$

Comparing (4.8) with (1.3), then the Jacobian of $\mathbf{x}$ can be viewed as the transpose of the product of linear backpropagation operators used to compute the error vectors associated with $\mathbf{x}$ at each layer. In [100, 101] the authors considered two initialisation schemes, in both the biases $\mathbf{b}_i^{(l)}$ at each layer are mutually independent and identically distributed Gaussians with mean 0 and variance σ_b^2. The weight matrices at each layer however either have mutually independent identically distributed Gaussian entries with mean 0 and variance σ_w^2/N_{l-1}, or are drawn from a uniform distribution over scaled orthogonal matrices such that $(\mathbf{W}^{(l)})^T\mathbf{W}^{(l)} = \sigma_w^2\mathbf{I}$. To analyse the Jacobian of the network at an arbitrary point $\mathbf{x}$, the authors considered again the large width setting, in which, as per the discussion in Section 4.1.1, the preactivations at layer l layer can be modelled as mutually independent, identically distributed, centred Gaussian random variables with variance $q^{(l)}$ [1]. For both of the initialisation schemes described, the limiting spectral density of $\mathbf{J}(\mathbf{x})$, in terms of its moment generating transform

$$M_{JJ^T}(z) = \sum_{k=1}^{\infty} \frac{m_k}{z^k}, \tag{4.9}$$

was computed and analysed using tools from free probability [100, 101, 99]. We refer the reader to [100, Appendix 6] for further details. In the context of ensuring dynamical isometry then of particular interest are the first and second moments, which can be expressed in terms of the moments μ_k of $\mathbf{D}^2$ and s_k of $(\mathbf{W}^T\mathbf{W})$. Here

[1] Note that (4.2) was derived in the context of Gaussian initialisation, for now we also assume it is also a reasonable approximation for the Orthogonal case.

we drop the dependence on the layer index l by assuming that the condition $q^{(0)} = q^*$ holds true for all inputs $\mathbf{x}$. Adopting again the notation used in [100],

$$\begin{aligned} \mu_k &:= \int \phi'(\sqrt{q^*}z)^{2k} d\gamma(z) = \mathbb{E}[\phi'(\sqrt{q^*})Z)^{2k}], \\ m_1 &:= (\sigma_w^2 \mu_1)^L = (\chi_1)^L, \\ m_2 &:= (\chi_1)^{2L} L\Big(\frac{\mu_1}{\mu_2} + \frac{1}{L} - 1 - s_1\Big). \end{aligned} \tag{4.10}$$

It is evident from (4.10) that the mean squared singular value m_1 of the Jacobian either exponentially explodes or vanishes unless the network is initialised on the EOC. However, although initialisation on the EOC makes m_1 independent of depth, the variance

$$\sigma_{JJ^T}^2 = m_2 - m_1^2 = L\Big(\frac{\mu_2}{\mu_1^2} - 1 - s_1\Big) \tag{4.11}$$

grows linearly with depth. As a result, under the limiting width assumption then for an arbitrary, normalised input vector initialisation on the EOC may still result in an increasingly ill-conditioned Jacobian with depth L. Indeed, we require that $m_1 = 1$ and $\sigma_{JJ^T}^2 \approx 0$ so as to at least approximately achieve dynamical isometry. Equation (4.11) shows that using a Gaussian initialisation scheme, corresponding to $s_1 = -1$, results in a linear growth in $\sigma_{JJ^T}^2$ with depth regardless of the activation function used. As a result, deep, dense feed-forward networks cannot achieve dynamical isometry using Gaussian initialisation. In contrast, orthogonal initialisation, corresponding to $s_1 = 0$, can in theory be used to ensure that $\sigma_{JJ^T}^2$ is arbitrarily close to one by controlling the moment ratio μ_2/μ_1^2. However, the ability to control the moment ratio depends on the activation function deployed: in the case of ReLU for instance, whose EOC is a singleton, the moment ratio is a constant and hence once again dynamical isometry cannot be achieved. As highlighted in [100], for certain activation functions, e.g., erf and tanh, which have a non-singleton EOC as illustrated by Figure 4.2(c), then the moment ratio can be reduced by shrinking q^*, which in turn can be achieved by reducing σ_b^2, see Figure 4.2 (d). Finally we remark that similar analyses were conducted in [57] in the context of finite width deep ReLU networks.

4.1.3 Contribution: activations which approximately preserve correlations and achieve dynamical isometry

To recap, in Section 4.1.1 we discussed how, in order to achieve deep information propagation and avoid the network being either highly sensitive or insensitive to perturbations of the input, it is important to avoid a rapid rate of convergence of

the correlation. To this end it is necessary to choose $(\phi, \sigma_w^2, \sigma_b^2)$ so that $\chi_1 = 1$ and the associated correlation function is close to identity. In Section 4.1.2 we discussed the problem of vanishing and exploding gradients. Prior works suggest that this problem can be avoided if the singular values of the input-output Jacobian at each layer concentrate around 1. The condition $\chi_1 = 1$ is equivalent to ensuring that the mean of the input-output Jacobian's spectrum at any layer is one. Therefore, if the variance $\sigma^2_{JJ^T}$ of the spectrum, defined in (4.11), is close to zero, then the spectrum of the input-output Jacobian is guaranteed to concentrate around one. Furthermore, with orthogonal initialisation on the EOC then so long as $\mu_2/\mu_1^2 \approx 1$ then $\sigma^2_{JJ^T} \approx 0$.

In this paper we present principles for choosing the activation function ϕ, see Definition 4.1.1, and an additional linear scaling parameter which simultaneously enables uniform convergence of the correlation function $R_{\phi,q^*}(\rho)$ to the identity map, and convergence of the variance $\sigma^2_{JJ^T}$ of the input-output Jacobian's spectrum to zero, without requiring σ_b^2 to shrink towards zero. For further details we refer the reader to Theorem 4.1.1. From prior work it was unclear how one could achieve this. For example, initialising on the EOC, using orthogonal initialisation and deploying the modulus activation function achieves perfect dynamical isometry, but results in rapid convergence of correlations in the forward pass, leading to poor training outcomes. Treated separately, a key theme that emerges in the prior works towards achieving both goals is that of reducing σ_b^2 in order to make q^* small. As mentioned briefly in section 4.1.1, in [59] the authors highlight that, for a smooth class of activation functions, taking this action can reduce the gap between the correlation map and the identity. The authors also provide a rule for selecting σ_b^2 just small enough so that pairwise correlations avoid being within some $\epsilon > 0$ of one at the output layer. Furthermore, experiments conducted in [59] indicate that selecting the variance hyperparameters in this manner accelerates training in practice. However, the authors also highlight that as the depth of the network increases this approach still requires $\sigma_b^2 \to 0$. Likewise, and as mentioned in section 4.1.2, in [100] the authors show, for the htanh and erf activation functions, that reducing q^* by shrinking σ_b^2 results in the moment ratio converging towards one. This solution however is not entirely satisfactory, first if the expected length of the activations goes to zero then asymptotically the network will on average annihilate inputs and fail to pass information to the output. Second, experimental evidence indicates that a small, but not overly small σ_b^2, gives the best results in practice, which we demonstrate in Section 4.3. Third, this solution relies only on the choice of σ_b^2 and ignores the role and design of ϕ. Our work seeks

to address this issue by achieving both goals through the design of ϕ, even when σ_b^2 and q^* are fixed away from zero.

In light of the above, and assuming σ_b^2 is fixed and that σ_w^2 is chosen in order that $\chi_1 = 1$, our goal is to design or choose ϕ in order that both $\max_{\rho\in[-1,1]} |R_{\phi,q^*}(\rho) - \rho| \approx 0$ and $|\mu_2/\mu_1^2 - 1| \approx 0$. As highlighted in [59], linear activation functions are advantageous from the perspective of slowing the convergence of the pairwise correlations in the forward pass. Likewise, [99, 108] highlight that linear networks with orthogonal initialisation achieve perfect dynamical isometry. However, linear activations are not a suitable option as a linear network can only represent linear functions. To preserve the rich approximation capabilities of nonlinear networks we analyse a set of activation functions characterised by being odd, bounded, Lipschitz continuous and linear around the origin. We refer to activation functions of this type, defined in Definition 4.1.1, as *scaled-bounded activations*.

Definition 4.1.1 (**scaled-bounded activations**). *We refer to the set of activation functions $\phi : \mathbb{R} \to \mathbb{R}$ which satisfy the following properties as scaled-bounded activations.*

1. *Continuous.*
2. *Odd, meaning that $\phi(z) = -\phi(-z)$ for all $z \in \mathbb{R}$.*
3. *Linear around the origin and bounded: in particular there exists $a, k \in \mathbb{R}_{>0}$ such that $\phi(z) = kz$ for all $z \in [-a, a]$ and $\phi(z) \leq ak$ for all $z \in \mathbb{R}$.*
4. *Twice differentiable at all points $z \in \mathbb{R}\backslash\mathcal{D}$, where $\mathcal{D} \subset \mathbb{R}$ is a finite set. Furthermore $|\phi'(z)| \leq k$ for all $z \in \mathbb{R}\backslash\mathcal{D}$.*

As illustrated in Figure 4.3, the structure of scaled-bounded activation functions outside of their linear region $[-a, a]$ can vary substantially. We note that for any scaled-bounded activation ϕ there exists $a, k \in \mathbb{R}_{>0}$ such that $|\phi(z)| \leq |\text{Shtanh}_{a,k}(z)|$, where

$$\text{Shtanh}_{a,k}(z) = \begin{cases} zk, & |z| < a \\ ak, & |z| \geq a. \end{cases} \tag{4.12}$$

Here Shtanh stands for scaled-bounded hard tanh, and we further note that $\text{Shtanh}_{a,k}(\cdot)$ is a scaled-bounded activation for any $a, k \in \mathbb{R}_{>0}$. Our main contribution is Theorem 4.1.1, which asserts, for scaled-bounded activations, that the correlation function R_{ϕ,q^*} can be made arbitrarily close to the identity on $[0, 1]$, and the moment ratio μ_2/μ_1^2 arbitrarily close to one, by choosing σ_b^2/a^2 to be small.

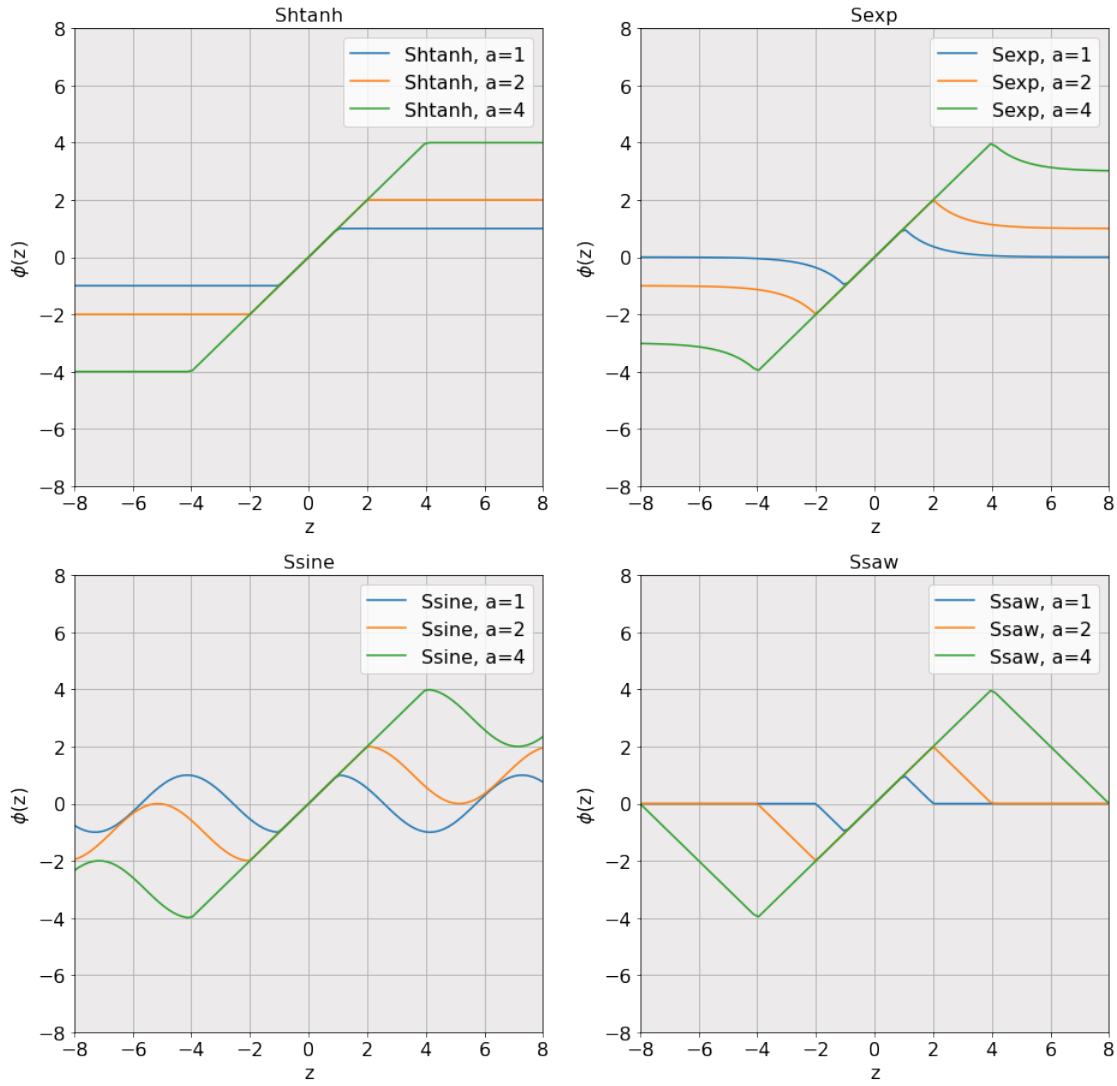


Figure 4.3: examples of scaled-bounded activation functions with $k = 1$ and $a \in \{1, 2, 4\}$. The prefix S refers to the scaling of the linear region via the parameter a.

Theorem 4.1.1. *Let ϕ be a scaled-bounded activation, see Definition 4.1.1, $\sigma_b^2 > 0$ and suppose that*

$$\chi_1 := \sigma_w^2 \mathbb{E}[\phi'(\sqrt{q^*}Z)^2] = 1,$$

where $q^* > 0$ is a fixed point of the associated variance function V_ϕ. In addition, assume that all inputs $\mathbf{x}$ are normalised so that $||\mathbf{x}||_2^2 = q^*$. With $y := \frac{\sigma_b^2}{a^2}$ and Λ defined as in Lemma 4.2.5, then

$$\max_{\rho \in [0,1]} |R_\phi(\rho) - \rho| < \left(\frac{8}{\pi}\right)^{1/3} y^{1/3} \tag{4.13}$$

and

$$\left|\frac{\mu_2}{\mu_1^2} - 1\right| \leq erf\left(\frac{\Lambda(y)}{\sqrt{2}}\right)^{-2} - 1. \tag{4.14}$$

We emphasise that as $y := \sigma_b^2/a^2 \to 0$ then both

$$\max_{\rho\in[0,1]} |R_{\phi,q^*}(\rho) - \rho|, \left|\mu_2/\mu_1^2 - 1\right| \to 0.$$

We remark that the uniform bound on the correlation provided by Theorem 4.1.1 is only for nonnegative correlations. However, Figure 4.7 indicates that letting $\sigma_b^2/a^2 \to 0$ results in $\max_{\rho\in[-1,1]} |R_{\phi,q^*}(\rho) - \rho| \to 0$. The key takeaway of Theorem 4.1.1 for practitioners is that an improved initialisation can be achieved by using an activation function which has a sufficiently large linear region spanning either side of the origin. The size of this linear region, a, needs to be selected on the basis of both the bias variance hyperparameter σ_b^2 and the depth of the network. The deeper the network or the larger σ_b^2 is, the larger a needs to be in order to avoid both convergence of the correlations and achieve approximate dynamical isometry. In practice, as per Figure 4.6, modest increases in a typically suffice to capture many of the potential benefits at initialisation. We emphasise that there is a tension between the linear region being large enough to achieve a good initialisation, while being small enough so that the expressivity of the network is not reduced. In particular, if the network is initialised so that it acts as linear transform on the input data, and if the optimiser becomes trapped in a local minimum close to the initialisation point, then the network may continue to act as a linear transform throughout training.

4.2 Analysis of scaled-bounded activation functions

4.2.1 Derivation of Theorem 4.1.1

We start our analysis by revisiting what it means to initialise on the EOC: the condition in (4.6) presupposes the existence of a fixed point q^* of V_ϕ. To resolve this matter we introduce and study the fixed points of a related function, W. Before presenting this analysis we introduce the following slight abuse of notation,

$$\phi'(z) := \frac{1}{2}\left(\frac{d\phi}{dz}(z^-) + \frac{d\phi}{dz}(z^+)\right) \tag{4.15}$$

where $\frac{d\phi}{dz}(z^-)$ and $\frac{d\phi}{dz}(z^+)$ are the left and right derivatives of ϕ at $z \in \mathbb{R}$ respectively. This is convenient for what follows as it allows us to define a notion of a derivative of a scaled-bounded activation ϕ over the whole of the domain: we remark, by Definition

4.1.1, that the true derivative of ϕ, and $\phi'(\cdot)$ as defined above, are equal almost everywhere.

Lemma 4.2.1. *Let ϕ be a scaled-bounded activation, see Definition 4.1.1, and $\sigma_b^2 > 0$. Define*

$$W_\phi(q) := \frac{\mathbb{E}[\phi(\sqrt{q}Z)^2]}{\mathbb{E}[\phi'(\sqrt{q}Z)^2]} + \sigma_b^2 \tag{4.16}$$

for all $q \in \mathbb{R}_{\geq 0}$. Then $W_\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ and W has a fixed point $q^* > 0$.

Proof. We first prove that $W_\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. To bound the numerator term then for any scaled-bounded activation function ϕ, see Definition 4.1.1, it follows that there exists $a, k \in \mathbb{R}> 0$ such that

$$0 \leq \mathbb{E}[\phi(\sqrt{q}Z)^2] < a^2k^2 < \infty. \tag{4.17}$$

for all $q \in \mathbb{R}_{\geq 0}$. As $\mathbb{E}[\phi'(\sqrt{q}Z)^2] \geq 0$ it suffices to show that the denominator is nonzero. For a given $q \in \mathbb{R}_{\geq 0}$. as $\phi'(\sqrt{q}z)^2 \geq 0$ for all $z \in \mathbb{R}\backslash\mathcal{D}$, then $\mathbb{E}[\phi'(\sqrt{q}Z)^2] = 0$ implies that $\phi'(\sqrt{q}z) = 0$ almost everywhere for $z \in \mathbb{R}$. This is a contradiction however as there exists by construction an $a > 0$ such that $\phi(\sqrt{q}z) = kz$ for $z \in [-a/\sqrt{q}, a/\sqrt{q}]$. As a result $\mathbb{E}[\phi'(\sqrt{q}Z)^2] > 0$. We therefore conclude that $W_\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$.

To prove that W_ϕ has a fixed point $q^* > 0$ we need to lower and upper bound $\mathbb{E}[\phi'(\sqrt{q}Z)^2]$. A lower bound can be derived as follows,

$$\begin{aligned}
\mathbb{E}[\phi'(\sqrt{q}Z)^2] &= \int_{\mathbb{R}} \phi'(\sqrt{q}Z)^2 d\gamma \\
&= 2\int_0^\infty \phi'(\sqrt{q}Z)^2 d\gamma \\
&= 2\int_0^{a/\sqrt{q}} k^2 d\gamma + 2\int_{a/\sqrt{q}}^\infty \phi'(\sqrt{q}Z)^2 d\gamma \\
&> 2k\int_0^{a/\sqrt{q}} \phi'(\sqrt{q}Z)^2 d\gamma \\
&= k^2 \mathrm{erf}\left(\frac{a}{\sqrt{2q}}\right).
\end{aligned}$$

Here the second equality follows from the fact that integrand is an even function and the equality on the third line by the construction of ϕ. The inequality on the fourth line follows from zeroing the second integral which is positive. Furthermore $|\phi'(z)| \leq k$ almost everywhere by construction, therefore we conclude that

$$k^2 \mathrm{erf}\left(\frac{a}{\sqrt{2q}}\right) \leq \mathbb{E}[\phi'(\sqrt{q}Z)^2] < k^2 < \infty \tag{4.18}$$

for all $q \in \mathbb{R}_0$.

We now prove that W_ϕ is continuous. Due to fact that $\mathbb{E}[\phi'(\sqrt{q}Z)^2] > 0$ then W_ϕ has no singularities. It therefore suffices to prove that both $q \mapsto \mathbb{E}[\phi(\sqrt{q}Z)^2]$ and $q \mapsto \mathbb{E}[\phi'(\sqrt{q}Z)^2]$ are continuous functions on $\mathbb{R}_{\geq 0}$. In both cases we achieve this by applying Lemma A.2.1 in the context of the measure space $(\mathbb{R}, \mathcal{B}(\mathbb{R}), \gamma)$, where γ denotes the standard one dimensional Gaussian measure and $\mathcal{B}(\mathbb{R})$ the completion of the Borel σ algebra on the real numbers. For $q \mapsto \mathbb{E}[\phi(\sqrt{q}Z)^2]$ with $q \in \mathbb{R}_{\geq 0}$ then condition 1 is satisfied due to (4.17), condition 2 follows from the continuity of ϕ and condition 3 is satisfied by $g(z) := a^2k^2$. For $q \mapsto \mathbb{E}[\phi'(\sqrt{q}Z)^2]$ with $q \in \mathbb{R}_{\geq 0}$ then condition 1 is satisfied due to (4.18), condition 2 follows from the fact that ϕ' is continuous almost everywhere in $\mathbb{R}$ and condition 3 is satisfied for all $q \in \mathbb{R}_{\geq 0}$ by $g(z) := k^2$. We conclude then that W_ϕ is continuous on $\mathbb{R}_{\geq 0}$.

To prove that 0 is not a fixed point, observe that $\frac{\mathbb{E}[\phi(\sqrt{q}Z)^2]}{\mathbb{E}[\phi'(\sqrt{q}Z)^2]} \geq 0$ and therefore $W_\phi(q) \geq \sigma_b^2 > 0$ for all $q \in \mathbb{R}_{\geq 0}$. Using (4.17) and (4.18) we may in addition derive the following upper bound on W_ϕ,

$$\sigma_b^2 \leq W_\phi(q) < \frac{a^2}{\text{erf}\left(\frac{a}{\sqrt{2q}}\right)} + \sigma_b^2 =: U(q) \tag{4.19}$$

for all $q \in \mathbb{R}_{\geq 0}$. Observe that as W_ϕ is continuous and $W_\phi(0) > 0$, then if W_ϕ has no fixed points it must hold that $W_\phi(q) > q$ for all $q \in \mathbb{R}_{\geq 0}$. Otherwise, by the intermediate value theorem, the function $W_\phi(p) - p$ must have a root and hence W must have a fixed point $q^* > 0$. Considering the upper bound U on W_ϕ from equation 4.19, assume that $U(q) > q$ for all $q \in \mathbb{R}_{\geq 0}$. Then

$$\frac{a^2}{\text{erf}\left(\frac{a}{\sqrt{2q}}\right)} + \sigma_b^2 > q,$$

$$a^2 + \sigma_b^2 \text{erf}\left(\frac{a}{\sqrt{2q}}\right) > q\text{erf}\left(\frac{a}{\sqrt{2q}}\right)$$

However, $\lim_{q\to\infty} a^2 + \sigma_b^2 \text{erf}\left(\frac{a}{\sqrt{2q}}\right) = a^2 < \infty$ while $\lim_{q\to\infty} q\text{erf}\left(\frac{a}{\sqrt{2q}}\right) = \infty$. As $q\text{erf}\left(\frac{a}{\sqrt{2q}}\right)$ is continuous then there must exist a $q \in \mathbb{R}_{\geq 0}$ such that $U(q) < q$, which is a contradiction. We therefore conclude that $W_\phi(q) < U(q) < q$ for some $q \in \mathbb{R}_{\geq 0}$ and that therefore W_ϕ must have a fixed point $q^* > 0$. □

As a consequence of Lemma 4.2.1, we are able to make the following claims concerning the existence of fixed points of the variance function V_ϕ for any scaled-bounded activation ϕ.

Corollary 4.2.1.1. *Let ϕ be a scaled-bounded activation, see Definition 4.1.1, $\sigma_b^2 > 0$ and suppose*

$$\chi_1 := \sigma_w^2 \mathbb{E}[\phi'(\sqrt{q^*}Z)^2] = 1, \tag{4.20}$$

where $q^* > 0$ is a fixed point of W_ϕ, defined in (4.16). Then q^* is a fixed point of the associated variance function V_ϕ, defined in (4.2).

Proof. From Lemma 4.2.1 it holds that there exists $q^* > 0$ such that $W_\phi(q^*) = q^*$. Inspecting (4.2) and (4.16) it follows that $V_\phi(q^*) = W_\phi(q^*) = q^*$, therefore q^* is a fixed point of V_ϕ. □

The key takeaway of Corollary 4.2.1.1 is that for any scaled-bounded activation there exists a fixed point q^* of V_ϕ satisfying $q^* > 0$. In fact, as $V_\phi(0) = \sigma_b^2 > 0$ then any fixed point of V_ϕ is greater than 0. We emphasise that such analysis is necessary as in general care is required when making assumptions concerning the existence of fixed points. For example, the variance function for ReLU like functions do not have any fixed points unless $\sigma_b^2 = 0$.

For what follows we require the following specific adaptation of integration by parts for piecewise continuously differentiable functions of Gaussian random variables with a finite number of discontinuities and bounded derivative.

Lemma 4.2.2. *Suppose $f : \mathbb{R} \to \mathbb{R}$ is bounded, piecewise continuously differentiable at all but a finite number $T \in \mathbb{N}$ of non-differentiable points, $t_1 < t_2 < ... < t_T$, and has bounded derivative. Then*

$$\int_{\mathbb{R}} z f(z) d\gamma(z) = \sum_i^T \left[-\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}} f(z) \right]_{t_i^+}^{t_i^-} + \int_{\mathbb{R}} f'(z) d\gamma(z) \tag{4.21}$$

Proof. We first note that as f and f' are continuous and continuous almost everywhere but at a finite number of points respectively, then they are clearly measureable with respect to the completion of the Borel sigma algebra on $\mathbb{R}$. Additionally, under the assumptions that f and f' are bounded, it follows that $f(z), zf(z), f'(z) \in L^1(\mathbb{R}, \mathcal{B}(\mathbb{R}), \gamma)$ where γ is the standard one dimensional Gaussian measure. Defining, for typographical ease, $g(z) := zf(z)$, $g_-(z) := \max\{-zf(z), 0\}$ and $g_+(z) := \max\{zf(z), 0\}$, then as $|g_\pm(z)| \leq |g(z)|$ it follows that $g_-(z), g_+(z) \in L^1(\mathbb{R}, \mathcal{B}(\mathbb{R}), \gamma)$. As both g_- and g_+ are nonnegative functions then $(g_- \mathbf{1}_{[-n,n]})_{n\in\mathbb{N}}$ and $(g_+ \mathbf{1}_{[-n,n]})_{n\in\mathbb{N}}$

are sequences of non-decreasing functions converging to g_- and g_+ respectively. Therefore, by monotone convergence

$$\begin{aligned}\int_{\mathbb{R}} zf(z)d\gamma(z) &= \int_{\mathbb{R}} g_+(z)d\gamma(z) - \int_{\mathbb{R}} g_-(z)d\gamma(z) \\ &= \lim_{n\to\infty}\int_{-n}^{n} g_+(z)d\gamma(z) - \lim_{n\to\infty}\int_{-n}^{n} g_-(z)d\gamma(z) \\ &= \lim_{n\to\infty}\int_{-n}^{n} zf(z)d\gamma(z).\end{aligned}$$

Let $n \in \mathbb{N}$ be any integer such that $n > |t_T|$. For typographical ease let $t_0 := -n$ and $t_{T+1} := n$, we proceed to analyse the integral of interest over $[-n, n]$.

$$\int_{-n}^{n} zf(z)d\gamma(z) = \sum_{i=0}^{T}\int_{t_i}^{t_{i+1}} f(z)z\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}dz.$$

By construction f is continuously differentiable on each of the above intervals of integration. Standard integration by parts gives

$$\int_{t_i}^{t_{i+1}} f(z)z\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}dz = \left[-\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}f(z)\right]_{t_i^+}^{t_{i+1}^-} + \int_{t_i}^{t_{i+1}} f'(z)d\gamma(z).$$

Collecting terms it follows that

$$\begin{aligned}\int_{-n}^{n} zf(z)d\gamma(z) &= \sum_{i=0}^{T+1}\left(\left[-\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}f(z)\right]_{t_i^+}^{t_{i+1}^-} + \int_{t_i}^{t_{i+1}} f'(z)d\gamma(z)\right) \\ &= \left[-\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}f(z)\right]_{-n}^{n} + \sum_{i=1}^{T}\left[-\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}f(z)\right]_{t_i^+}^{t_i^-} + \int_{-n}^{n} f'(z)d\gamma(z).\end{aligned}$$

Defining $f'_-(z) := \max\{-f'(z), 0\}$ and $f'_+(z) := \max\{f'(z), 0\}$, then as $|f_\pm(z)| \leq |f(z)|$ it follows that $f_-(z), f_+(z) \in L^1(\mathbb{R}, \mathcal{B}(\mathbb{R}), \gamma)$. As both f_- and f_+ are non-negative functions, $(f_-\mathbf{1}_{[-n,n]})_{n\in\mathbb{N}}$ and $(f_+\mathbf{1}_{[-n,n]})_{n\in\mathbb{N}}$ are sequences of non-decreasing functions converging to f^+ and f^- respectively. Therefore by monotone convergence

$$\begin{aligned}\int_{\mathbb{R}} f'(z)d\gamma(z) &= \int_{\mathbb{R}} f'_-(z)d\gamma(z) - \int_{\mathbb{R}} f'_+(z)d\gamma(z) \\ &= \lim_{n\to\infty}\int_{-n}^{n} f'_-(z)d\gamma(z) - \lim_{n\to\infty}\int_{-n}^{n} f'_+(z)d\gamma(z) \\ &= \lim_{n\to\infty}\int_{-n}^{n} f'(z)d\gamma(z).\end{aligned}$$

As a result

$$\begin{aligned}\int_{\mathbb{R}} zf(z)d\gamma(z) &= \lim_{n\to\infty}\int_{-n}^{n} zf(z)d\gamma(z)\\ &= \lim_{n\to\infty}\left[-\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}f(z)\right]_{-n}^{n} + \sum_{i=1}^{T}\left[-\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}f(z)\right]_{t_i^+}^{t_i^-}\\ &+ \lim_{n\to\infty}\int_{-n}^{n} f'(z)d\gamma(z)\\ &= \sum_{i=1}^{T}\left[-\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}}f(z)\right]_{t_i^+}^{t_i^-} + \int_{\mathbb{R}} f'(z)d\gamma(z)\end{aligned}$$

as claimed. □

To simplify the analysis we will now proceed under the assumption that the input data is normalised to have euclidean length q^* and that $q^{(l)} = q^*$ for all $l \in [L]$. If one where interested in initialising in this manner in practice, it might also be of interest to explore guarantees concerning the ease with which a stable fixed point of V_ϕ can be computed: in particular if the fixed point is not attractive then numerical precision issues might mean that the sequence $q^{(l)}$ diverges from q^*. We do not pursue this line of inquiry and instead leave it as potential future work. Now that we have identified the existence of fixed points of the variance functions of scaled-bounded activations, it is possible to study the associated correlation functions.

Lemma 4.2.3. *Let ϕ be a scaled-bounded activation, see Definition 4.1.1, $\sigma_b^2 > 0$ and suppose*

$$\chi_1 := \sigma_w^2\mathbb{E}[\phi'(\sqrt{q^*}Z)^2] = 1, \tag{4.22}$$

where $q^* > 0$ is a fixed point of W_ϕ. In addition, assume that all inputs $\mathbf{x}$ are normalised so that $||\mathbf{x}||_2^2 = q^*$. Then the associated correlation map R_{ϕ,q^*}, defined in (4.4), is fixed at each layer $l \in [L]$, satisfies $R_{\phi,q^*} : [-1,1] \to [-1,1]$ and is differentiable with

$$R'_{\phi,q^*}(\rho) = \sigma_w^2\mathbb{E}[\phi'(U_1)\phi'(U_2)] \tag{4.23}$$

for all input correlations $\rho \in (-1,1)$.

Proof. Recall that $U_1 := \sqrt{q^*}Z_1$ and $U_2 := \sqrt{q^*}(\rho Z_1 + \sqrt{1-\rho^2}Z_2)$. From Corollary 4.2.1.1 we know that q^* is a fixed point of V_ϕ and therefore the variance of all inputs, given the assumed normalisation, will remain fixed at q^* for all layers of the network. Since $q^* > 0$, then $R_{\phi,q^*} : [-1,1] \to [-1,1]$ by construction as long as the correlation function (4.4) is finite for any $\rho \in [-1,1]$. This follows by Cauchy-Schwarz,

$$|\mathbb{E}[\phi(U_1)\phi(U_2)]| \leq \mathbb{E}[\phi(U_1)^2]^{1/2}\mathbb{E}[\phi'(U_2)^2]^{1/2} < a^2k^2 < \infty. \tag{4.24}$$

It remains to be proved that R_{ϕ,q^*} is differentiable on $(-1,1)$ and to derive (4.23). To this end it suffices to show that $H(\rho) := \mathbb{E}[\phi(U_1)\phi(U_2)]$ is differentiable on $(-1,1)$ and derive an expression for its derivative. We rewrite H as follows,

$$H(\rho) := \int_{\mathbb{R}\times\mathbb{R}} \phi(u_1)\phi(u_2) d\gamma^{(2)}(z_1, z_2)$$

recalling that $\gamma^{(2)}$ denotes the standard two dimensional Gaussian measure. We proceed by applying Lemma A.2.2 in the context of the measure space $(\mathbb{R}^2, \mathcal{B}(\mathbb{R}^2), \gamma^2)$, where $\mathcal{B}(\mathbb{R}^2)$ denotes the completion of the Borel σ-algebra on $\mathbb{R}^2$, and the interval $(-1,1)$. First observe that condition 1 of Lemma A.2.2 is satisfied as

$$\int_{\mathbb{R}\times\mathbb{R}} |\phi(u_1)\phi(u_2)| d\gamma^{(2)}(z_1, z_2) < a^2k^2 < \infty.$$

For condition 2, by construction $(\phi \circ u_2)(\rho, z_1, z_2)$ is non-differentiable only on the set

$$\bigcup_{i=1}^{|\mathcal{D}|} \{(z_1, z_2) : \rho z_1 + \sqrt{1-\rho^2} z_2 = \frac{d_i}{\sqrt{q^*}}\}.$$

This is the union of a finite number of one dimensional lines in $\mathbb{R}^2$ and hence has measure 0. Therefore, for each $\rho \in (-1,1)$, then $\frac{\partial \phi \circ u_2}{(\partial \rho)}(x, z_1, z_2)$ exists almost everywhere and hence condition 2 is also satisfied. Finally, for condition 3 note that this partial derivative can be expressed as

$$\begin{aligned}
\frac{\partial \phi \circ u_2}{\partial \rho}(\rho, z_1, z_2) &= \frac{\partial u_2}{\partial \rho}(\rho, z_1, z_2)\phi'(u_2) \\
&= \sqrt{q^*}\left(z_1 - \frac{\rho}{\sqrt{1-\rho^2}} z_2\right)\phi'(u_2),
\end{aligned}$$

from which it follows

$$\begin{aligned}
\left|\frac{\partial \phi \circ u_2}{\partial \rho}(\rho, z_1, z_2)\right| &\leq \sqrt{q^*}\left(|z_1| + \frac{|\rho|}{\sqrt{1-\rho^2}}|z_2|\right)|\phi'(u_2)| \\
&< \sqrt{q^*}k\left(|z_1| + \frac{|\rho|}{\sqrt{1-\rho^2}}|z_2|\right).
\end{aligned}$$

For any $\rho \in (-1,1)$ consider the open interval $(-1+\delta(\rho), 1-\delta(\rho))$, where $\delta(\rho) := \frac{|1-\rho|}{2}$. It follows that

$$\sup_{\rho \in (-1+\delta(\rho), 1-\delta(\rho))} \left|\frac{\partial \phi \circ u_2}{\partial \rho}(\rho, z_1, z_2)\right| \leq \sqrt{q^*}k\left(|z_1| + \frac{1-\delta(\rho)}{\sqrt{2\delta(\rho) - \delta(\rho)^2}}|z_2|\right) =: g_K(z_1, z_2).$$

Letting $\kappa_\delta := \frac{1-\delta(\rho)}{\sqrt{2\delta(\rho)-\delta(\rho)^2}}$, then by applying the Fubini-Tonelli theorem it follows that

$$\begin{aligned}\int_{\mathbb{R}^2} g_K(z_1,z_2)d\gamma^{(2)}(z_1,z_2) &= \sqrt{q^*}k\left(\int_{\mathbb{R}^2}|z_1|d\gamma^{(2)}(z_1,z_2)+\kappa_\delta\int_{\mathbb{R}^2}|z_2|d\gamma^{(2)}(z_1,z_2)\right)\\ &= \sqrt{q^*}k\sqrt{\frac{2}{\pi}}\left(1+\kappa_\delta\right)\\ &< \infty.\end{aligned}$$

Hence for any $\rho \in (-1,1)$ there exists an open interval $K = (-1+\delta(\rho), 1-\delta(\rho))$ with $\rho \in K$ and an integrable function $g_K(z_1,z_2) : \mathbb{R}^2 \to \mathbb{R}$, such that

$$\left|\frac{\partial(\phi\circ u_2)}{\partial\rho}(\rho,z_1,z_2)\right| \leq g_K(z_1,z_2).$$

We conclude then that condition 3 is also satisfied, and therefore, by Lemma A.2.2, H is differentiable on $(-1,1)$. The derivative of H can be expressed as follows,

$$\begin{aligned}H'(\rho) &= \int_{\mathbb{R}^2}\phi(u_1)\frac{\partial u_2}{\partial\rho}\phi'(u_2)d\gamma^{(2)}(z_1,z_2)\\ &= \sqrt{q^*}\int_{\mathbb{R}}\int_{\mathbb{R}}\phi(u_1)\left(z_1-\frac{\rho}{\sqrt{1-\rho^2}}z_2\right)\phi'(u_2)d\gamma(z_1)d\gamma(z_2)\\ &= \sqrt{q^*}\int_{\mathbb{R}}\left(\int_{\mathbb{R}}z_1\phi(u_1)\phi'(u_2)d\gamma(z_1)\right)d\gamma(z_2)\\ &- \frac{\rho\sqrt{q^*}}{\sqrt{1-\rho^2}}\int_{\mathbb{R}}\phi(u_1)\left(\int_{\mathbb{R}}z_2\phi'(u_2)d\gamma(z_2)\right)d\gamma(z_1),\end{aligned} \tag{4.25}$$

where the third equality in the above follows by applying the Fubini-Tonelli theorem. We proceed first to derive an expression for $H'(0)$ and then to analyse $H'(\rho)$ for $\rho \in (-1,0)\cup(0,1)$. In all that follows, and as ϕ is odd, we let $-d_T... < -d_2 < -d_1 < d_1 < d_2.. < d_T$ be the elements of $\mathcal{D}$ where $T := |\mathcal{D}|/2$. From (4.25) it follows that

$$\begin{aligned}H'(0) &= \sqrt{q^*}\int_{\mathbb{R}}\left(\int_{\mathbb{R}}z_1\phi(\sqrt{q^*}z_1)\phi'(\sqrt{q^*z_2})d\gamma(z_1)\right)d\gamma(z_2)\\ &= \sqrt{q^*}\int_{\mathbb{R}}\phi'(\sqrt{q^*z_2})\left(\int_{\mathbb{R}}z_1\phi(\sqrt{q^*}z_1)d\gamma(z_1)\right)d\gamma(z_2)\end{aligned}$$

By construction $\phi(\sqrt{q^*}z_1)$ is differentiable at all points other than

$$\begin{aligned}s_i &:= \frac{d_i}{\sqrt{q^*}},\\ l_i &:= -\frac{d_i}{\sqrt{q^*}}\end{aligned}$$

for $i \in [T]$. Applying Lemma 4.2.2

$$\begin{aligned}\int_{\mathbb{R}} z_1\phi(\sqrt{q^*}z_1)d\gamma(z_1) &= -\sum_i^T \left(\left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\right]_{l_i^+}^{l_i^-} + \left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\right]_{s_i^+}^{s_i^-}\right) \\ &\quad + \sqrt{q^*}\int_{\mathbb{R}} \phi'(\sqrt{q^*}z_1)d\gamma(z_1) \\ &= \sqrt{q^*}\int_{\mathbb{R}} \phi'(\sqrt{q^*}z_1)d\gamma(z_1).\end{aligned}$$

Here the second equality follows from the continuity of ϕ. Therefore

$$\begin{aligned}H'(0) &= \sqrt{q^*}\int_{\mathbb{R}} \phi'(\sqrt{q^*}z_2)\left(\sqrt{q^*}\int_{\mathbb{R}} \phi'(\sqrt{q^*}z_1)d\gamma(z_1)\right)d\gamma(z_2) \\ &= q^*\int_{\mathbb{R}^2} \phi'(\sqrt{q^*}z_1)\phi'(\sqrt{q^*}z_2)d\gamma^{(2)}(z_1,z_2)\end{aligned} \tag{4.26}$$

To derive an expression for $H'(\rho)$ for any $\rho \in (-1,0)\cup(0,1)$, we apply integration by parts to each of the inner integrals in (4.25) using Lemma 4.2.2. Starting with the second inner integral, with $\rho \in (-1,0)\cup(0,1)$ and $z_1 \in \mathbb{R}$ fixed, we define $\psi : \mathbb{R} \to \mathbb{R}$ as the function $z_2 \mapsto (\phi' \circ u_2)(z_1, z_2, \rho)$. By construction ϕ' is continuously differentiable at all points other than the following,

$$\begin{aligned}p_i(z_1) &:= \frac{d_i}{\sqrt{q^*(1-\rho^2)}} - z_1\frac{\rho}{\sqrt{1-\rho^2}}, \\ n_i(z_1) &:= -\frac{d_i}{\sqrt{q^*(1-\rho^2)}} - z_1\frac{\rho}{\sqrt{1-\rho^2}}\end{aligned}$$

for all $i \in [T]$. Applying Lemma 4.2.2 it follows that

$$\begin{aligned}\int_{\mathbb{R}} z_2\psi(z_2)d\gamma(z_2) &= -\sum_i^T \left(\left[\frac{\exp(-\frac{1}{2}z_2^2)}{\sqrt{2\pi}}\psi(z_2)\right]_{n_i^+}^{n_i^-} + \left[\frac{\exp(-\frac{1}{2}z_2^2)}{\sqrt{2\pi}}\psi(z_2)\right]_{p_i^+}^{p_i^-}\right) + \int_{\mathbb{R}} \psi'(z_2)d\gamma(z_2) \\ &=: -\kappa_1(z_1) + \int_{\mathbb{R}} \psi'(z_2)d\gamma(z_2).\end{aligned}$$

Therefore the second integral on the final line of (4.25) can be expressed as

$$\begin{aligned}&\int_{\mathbb{R}} \phi(u_1)\left(\int_{\mathbb{R}} z_2\phi'(u_2)d\gamma(z_2)\right)d\gamma(z_1) \\ &= -\int_{\mathbb{R}} \phi(u_1)\kappa_1(z_1)d\gamma(z_1) + \int_{\mathbb{R}} \phi(u_1)\int_{\mathbb{R}} \psi'(z_2)d\gamma(z_2)d\gamma(z_1) \\ &= -\int_{\mathbb{R}} \phi(u_1)\kappa_1(z_1)d\gamma(z_1) + \sqrt{q^*(1-\rho^2)}\int_{\mathbb{R}^2} \phi(u_1)\phi''(u_2)d\gamma^{(2)}(z_1,z_2).\end{aligned}$$

Analysing $\kappa_1(z_1)$, then by construction $\psi(n_i) = \phi'(-d_i)$ and $\psi(p_i) = \phi'(d_i)$. Furthermore, as ϕ is odd and continuous then ϕ' is even and as a result $\phi'(-d_i^-) = \phi'(d_i^+)$ and $\phi'(-d_i^+) = \phi'(d_i^-)$. Therefore

$$\begin{aligned}
\kappa_1(z_1) &= \sum_i^T \left(\left[\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}} \psi(z) \right]_{n_i^+}^{n_i^-} + \left[\frac{\exp(-\frac{1}{2}z^2)}{\sqrt{2\pi}} \psi(z) \right]_{p_i^+}^{p_i^-} \right) \\
&= \sum_i^T \left(\frac{\exp(-\frac{1}{2}n_i^2)}{\sqrt{2\pi}} \left(\psi(n_i^-) - \psi(n_i^+) \right) + \frac{\exp(-\frac{1}{2}p_i^2)}{\sqrt{2\pi}} \left(\psi(p_i^-) - \psi(p_i^+) \right) \right) \\
&= \sum_i^T \left(\frac{\exp(-\frac{1}{2}n_i^2)}{\sqrt{2\pi}} \left(\phi'(-d_i^-) - \phi'(-d_i^+) \right) + \frac{\exp(-\frac{1}{2}p_i^2)}{\sqrt{2\pi}} \left(\phi'(d_i^-) - \phi'(d_i^+) \right) \right) \\
&= \sum_i^T \left(\frac{\exp(-\frac{1}{2}n_i^2)}{\sqrt{2\pi}} \left(\phi'(d_i^+) - \phi'(d_i^-) \right) + \frac{\exp(-\frac{1}{2}p_i^2)}{\sqrt{2\pi}} \left(\phi'(d_i^-) - \phi'(d_i^+) \right) \right) \\
&= \frac{1}{\sqrt{2\pi}} \sum_i^T \left(\phi'(d_i^+) - \phi'(d_i^-) \right) \left(\exp(-\frac{1}{2}n_i^2) - \exp(-\frac{1}{2}p_i^2) \right).
\end{aligned}$$

Expanding the integral involving κ_1 then

$$\int_{\mathbb{R}} \phi(u_1)\kappa_1(z_1)d\gamma(z_1) = \sum_i^T \left(\phi'(d_i^+) - \phi'(d_i^-) \right) \int_{\mathbb{R}} \phi(u_1) \frac{\exp(-\frac{1}{2}n_i^2) - \exp(-\frac{1}{2}p_i^2)}{\sqrt{2\pi}} d\gamma(z_1).$$

Observe that as

$$\begin{aligned}
p_i(z_1)^2 &= \frac{d_i^2}{q^*(1-\rho^2)} - z_1 \frac{2d_i\rho}{\sqrt{q^*}(1-\rho^2)} + z_1^2 \frac{\rho^2}{1-\rho^2} \\
n_i(z_1)^2 &= \frac{d_i^2}{q^*(1-\rho^2)} + z_1 \frac{2d_i\rho}{\sqrt{q^*}(1-\rho^2)} + z_1^2 \frac{\rho^2}{1-\rho^2}
\end{aligned}$$

then clearly $n_i(z_1)^2 = p_i(-z_1)^2$. Let $\beta_i(z_1) := e^{-\frac{1}{2}n_i(z_1)^2} - e^{-\frac{1}{2}p_i(z_1)^2}$ for all $i \in [T]$, then these β_i are odd functions as

$$\begin{aligned}
\beta_i(-z_1) &= e^{-\frac{1}{2}n_i(-z_1)^2} - e^{-\frac{1}{2}p_i(-z_1)^2} \\
&= e^{-\frac{1}{2}p_i(z_1)^2} - e^{-\frac{1}{2}n_i(z_1)^2} \\
&= -\beta_i(z_1).
\end{aligned}$$

As the product of two odd functions is odd then

$$\int_{\mathbb{R}} \phi(u_1)\kappa_1(z_1)d\gamma(z_1) = \frac{1}{\sqrt{2\pi}} \sum_i^T \left(\phi'(d_i^+) - \phi'(d_i^-) \right) \int_{\mathbb{R}} \phi(u_1)\beta_i(z_1)d\gamma(z_1) = 0$$

and so

$$\int_{\mathbb{R}} \phi(u_1) \left(\int_{\mathbb{R}} z_2\phi'(u_2)d\gamma(z_2) \right) d\gamma(z_1) = \sqrt{q^*(1-\rho^2)} \int_{\mathbb{R}^2} \phi(u_1)\phi''(u_2)d\gamma^{(2)}(z_1, z_2). \tag{4.27}$$

Now we turn our attention to the first inner integral on the final line of (4.25). With $\rho \in (-1,0) \cup (0,1)$ and $z_1 \in \mathbb{R}$ fixed, and defining for typographical ease $\varphi : \mathbb{R} \to \mathbb{R}$ as the function $z_1 \mapsto (\phi' \circ u_2)(z_1, z_2, \rho)$, then $\phi(u_1)\phi'(u_2)$ is continuously differentiable at all points other than the following,

$$\begin{aligned}
o_i(z_2) &:= \frac{d_i}{\rho\sqrt{q^*}} - z_2\frac{\sqrt{1-\rho^2}}{\rho} \\
e_i(z_2) &:= -\frac{d_i}{\rho\sqrt{q^*}} - z_2\frac{\sqrt{1-\rho^2}}{\rho} \\
s_i &:= \frac{d_i}{\sqrt{q^*}} \\
l_i &:= -\frac{d_i}{\sqrt{q^*}}
\end{aligned}$$

for all $i \in [T]$. Applying Lemma 4.2.2,

$$\begin{aligned}
&\int_{\mathbb{R}} z_1\phi(\sqrt{q^*}z_1)\varphi(z_1)d\gamma(z_1) = \\
&-\sum_i^T \left(\left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\varphi(z_1) \right]_{e_i^+}^{e_i^-} + \left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\varphi(z_1) \right]_{o_i^+}^{o_i^-} \right) \\
&-\sum_i^T \left(\left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\varphi(z_1) \right]_{l_i^+}^{l_i^-} + \left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\varphi(z_1) \right]_{s_i^+}^{s_i^-} \right) \\
&+ \int_{\mathbb{R}} \phi(\sqrt{q^*}z_1)\varphi'(z_1) + \sqrt{q^*}\phi'(\sqrt{q^*}z_1)\varphi(z_1)d\gamma(z_1).
\end{aligned}$$

As both ϕ and φ are continuous at l_i and s_i for all $i \in [T]$, then the left and right limits of $\phi(\sqrt{q^*}z_1)\varphi(z_1)$ at these points are equal. Therefore

$$\begin{aligned}
&\int_{\mathbb{R}} z_1\phi(\sqrt{q^*}z_1)\varphi(z_1)d\gamma(z_1) = \\
&-\sum_i^T \left(\left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\varphi(z_1) \right]_{e_i^+}^{e_i^-} + \left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\varphi(z_1) \right]_{o_i^+}^{o_i^-} \right) \\
&+ \int_{\mathbb{R}} \phi(\sqrt{q^*}z_1)\varphi'(z_1) + \sqrt{q^*}\phi'(\sqrt{q^*}z_1)\varphi(z_1)d\gamma(z_1) \\
&=: -\kappa_2(z_2) + \int_{\mathbb{R}} \phi(\sqrt{q^*}z_1)\varphi'(z_1) + \sqrt{q^*}\phi'(\sqrt{q^*}z_1)\varphi(z_1)d\gamma(z_1).
\end{aligned}$$

Analysing κ_2, then by construction $\varphi(e_i) = \phi'(-d_i)$ and $\varphi(o_i) = \phi'(d_i)$. Similar to

before it follows that

$$\begin{aligned}
\kappa_2(z_2) &= \sum_{i=1}^{T}\left(\left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\varphi(z_1)\right]_{e_i^+}^{e_i^-} + \left[\frac{\exp(-\frac{1}{2}z_1^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}z_1)\varphi(z_1)\right]_{o_i^+}^{o_i^-}\right)\\
&= \sum_{i=1}^{T}\left(\frac{\exp(-\frac{1}{2}e_i^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}e_i)\left(\varphi(-d_i^-)-\varphi(-d_i^+)\right) + \frac{\exp(-\frac{1}{2}o_i^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}o_i)\left(\varphi(d_i^-)-\varphi(d_i^+)\right)\right)\\
&= \sum_{i=1}^{T}\left(\frac{\exp(-\frac{1}{2}e_i^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}e_i)\left(\varphi(d_i^+)-\varphi(d_i^-)\right) + \frac{\exp(-\frac{1}{2}o_i^2)}{\sqrt{2\pi}}\phi(\sqrt{q^*}o_i)\left(\varphi(d_i^-)-\varphi(d_i^+)\right)\right)\\
&= \frac{1}{\sqrt{2\pi}}\sum_{i=1}^{T}\left(\left(\varphi(d_i^+)-\varphi(d_i^-)\right)\left(\exp(-\frac{1}{2}e_i^2)\phi(\sqrt{q^*}e_i)-\exp(-\frac{1}{2}o_i^2)\phi(\sqrt{q^*}o_i)\right)\right)
\end{aligned}$$

Note that $o_i(-z_2) = -e_i(z_2)$, therefore $\phi(\sqrt{q^*}o_i(-z_2)) = \phi(-\sqrt{q^*}e_i(z_2)) = -\phi(\sqrt{q^*}e_i(z_2))$ as ϕ is odd. Likewise $e_i(-z_2) = -o_i(z_2)$ and so $\phi(\sqrt{q^*}e_i(-z_2)) = \phi(-\sqrt{q^*}o_i(z_2)) = -\phi(\sqrt{q^*}o_i(z_2))$. Furthermore as

$$\begin{aligned}
o_i(z_2)^2 &= \frac{d_i^2}{\rho^2 q^*} - z_2\frac{\sqrt{1-\rho^2}}{\rho} + z_2^2\frac{1-\rho^2}{\rho^2},\\
e_i(z_2)^2 &= \frac{d_i^2}{\rho^2 q^*} + z_2\frac{\sqrt{1-\rho^2}}{\rho} + z_2^2\frac{1-\rho^2}{\rho^2}
\end{aligned}$$

then $o_i(-z_2)^2 = e_i(z_2)^2$. Analogous to β_i, we now define $\Gamma_i(z_2) := \exp(-\frac{1}{2}e_i^2)\phi(\sqrt{q^*}e_i) - \exp(-\frac{1}{2}o_i^2)\phi(\sqrt{q^*}o_i)$. The fact that $\Gamma_i(z_2)$ is odd follows from

$$\begin{aligned}
\Gamma_i(-z_2) &= \exp(-\frac{1}{2}e_i(-z_2)^2)\phi(\sqrt{q^*}e_i(-z_2))) - \exp(-\frac{1}{2}o_i(-z_2)^2)\phi(\sqrt{q^*}o_i(-z_2))\\
&= -\exp(-\frac{1}{2}o_i(z_2)^2)\phi(\sqrt{q^*}o_i(z_2))) + \exp(-\frac{1}{2}e_i(z_2)^2)\phi(\sqrt{q^*}e_i(z_2))\\
&= -\Gamma_i(z_2).
\end{aligned}$$

Analysing the first integral on the last line of 4.25), then as Γ_i is odd for each $i \in [T]$

$$\begin{aligned}
&\int_{\mathbb{R}}\left(\int_{\mathbb{R}} z_1\phi(u_1)\phi'(u_2)d\gamma(z_1)\right)d\gamma(z_2)\\
&= -\int_{\mathbb{R}}\kappa_2(z_2)d\gamma(z_2) + \int_{\mathbb{R}}\int_{\mathbb{R}}\phi(\sqrt{q^*}z_1)\varphi'(z_1) + \sqrt{q^*}\phi'(\sqrt{q^*}z_1)\varphi(z_1)d\gamma(z_1)d\gamma(z_2)\\
&= -\frac{1}{\sqrt{2\pi}}\sum_{i=1}^{T}\left(\varphi(d_i^+)-\varphi(d_i^-)\right)\int_{\mathbb{R}}\Gamma_i(z_2)d\gamma(z_2) + \int_{\mathbb{R}^2}\phi(\sqrt{q^*}z_1)\varphi'(z_1)d\gamma^{(2)}(z_1,z_2)\\
&+ \sqrt{q^*}\int_{\mathbb{R}^2}\phi'(\sqrt{q^*}z_1)\varphi(z_1)d\gamma^{(2)}(z_1,z_2)\\
&= \sqrt{q^*}\rho\int_{\mathbb{R}^2}\phi(u_1)\phi''(u_2)d\gamma^{(2)}(z_1,z_2) + \sqrt{q^*}\int_{\mathbb{R}^2}\phi'(u_1)\phi'(u_2)d\gamma^{(2)}(z_1,z_2).
\end{aligned} \tag{4.28}$$

Substituting (4.28) and (4.27) into (4.25) it follows that

$$
\begin{aligned}
&H'(\rho) \\
&= \sqrt{q^*}\left(\int_{\mathbb{R}}\left(\int_{\mathbb{R}} z_1\phi(u_1)\phi'(u_2)d\gamma(z_1)\right)d\gamma(z_2) - \frac{\rho}{\sqrt{1-\rho^2}}\int_{\mathbb{R}}\phi(u_1)\left(\int_{\mathbb{R}} z_2\phi'(u_2)d\gamma(z_2)\right)d\gamma(z_1)\right) \\
&= q^*\left(\rho\int_{\mathbb{R}^2}\phi(u_1)\phi''(u_2)d\gamma^{(2)}(z_1,z_2) + \int_{\mathbb{R}^2}\phi'(u_1)\phi'(u_2)d\gamma(z_1,z_2) - \rho\int_{\mathbb{R}^2}\phi(u_1)\phi''(u_2)d\gamma^{(2)}(z_1,z_2)\right) \\
&= q^*\int_{\mathbb{R}^2}\phi'(u_1)\phi'(u_2)d\gamma^{(2)}(z_1,z_2).
\end{aligned}
$$

It therefore follows for all $\rho \in (-1,1)$ that

$$R'_{\phi,q^*}(\rho) = \sigma_w^2\int_{\mathbb{R}^2}\phi'(u_1)\phi'(u_2)d\gamma^{(2)}(z_1,z_2) = \sigma_w^2\mathbb{E}[\phi'(U_1)\phi'(U_2)]$$

as claimed. □

The expression for the correlation provided in (4.5) is equivalent to that given in [103], however we emphasise, due to the fact that ϕ is not necessarily continuously differentiable, that significantly more care is required to derive it. Indeed, if ϕ were not odd then this equivalence would not hold and additional terms would appear, potentially complicating the analysis downstream. We are now ready to present a key lemma, which provides a tight uniform bound on the interval $[0,1]$ between the correlation functions associated with scaled-bounded activations and the identity function.

Lemma 4.2.4. *Under the same conditions and assumptions as in Lemma 4.2.3, it holds that*

$$\max_{\rho\in[0,1]}|R_{\phi,q^*}(\rho) - \rho| = \frac{\sigma_b^2}{q^*}. \tag{4.29}$$

Proof. Observe that

$$
\begin{aligned}
R_{\phi,q^*}(0) &= \frac{\mathbb{E}[\phi(\sqrt{q^*}Z_1)\phi(\sqrt{q^*}Z_2)]}{q^*\mathbb{E}[\phi'(\sqrt{q^*}Z)^2]} + \frac{\sigma_b^2}{q^*} \\
&= \frac{\mathbb{E}[\phi(\sqrt{q^*}Z_1)]^2}{q^*\mathbb{E}[\phi'(\sqrt{q^*}Z)^2]} + \frac{\sigma_b^2}{q^*}.
\end{aligned}
$$

As ϕ is odd then $\mathbb{E}[\phi(\sqrt{q^*}Z_1)] = 0$ and therefore

$$R_{\phi,q^*}(0) = \frac{\sigma_b^2}{q^*} > 0.$$

Lemma 4.2.4 follows then as long as $|R_{\phi,q^*}(\rho) - \rho| < R_{\phi,q^*}(0)$ for all $\rho \in [0,1]$. As

$$R_{\phi,q^*}(1) = \frac{\sigma_w^2}{q^*}\mathbb{E}[\phi(\sqrt{q^*}Z_1)^2] + \frac{\sigma_b^2}{q^*} = \frac{V(q^*)}{q^*} = 1$$

then $|R_{\phi,q^*}(1) - 1| = 0 < R_{\phi,q^*}(0)$. All that remains to show is that the inequality holds for $\rho \in (0, 1)$. We proceed using an approach similar to that used to prove Proposition 3 in [59]. Using Lemma 4.2.3 then for any $\rho \in (0, 1)$ we have

$$\begin{aligned} R'_{\phi,q^*}(\rho) &= \frac{\mathbb{E}[\phi'(U_1)\phi'(U_2)]}{\mathbb{E}[\phi'(\sqrt{q^*}Z)^2]} \\ &\leq \frac{\mathbb{E}[\phi'(\sqrt{q^*}Z_1)^2]^{\frac{1}{2}}\mathbb{E}[\phi'(U_2)^2]^{\frac{1}{2}}}{\mathbb{E}[\phi'(\sqrt{q^*}Z)^2]} \\ &= \frac{\mathbb{E}[\phi'(U_2)^2]^{\frac{1}{2}}}{\mathbb{E}[\phi'(\sqrt{q^*}Z)^2]^{\frac{1}{2}}} \\ &= 1. \end{aligned}$$

The inequality on the second line of the above follows from Cauchy-Schwarz and the equalities on the third and fourth lines are due to the fact that $Z_1, Z, U_2 \sim \mathcal{N}(0, 1)$ are all identically distributed. Note that equality holds iff either $\rho = 0$ or there exists an $\alpha \in \mathbb{R}$ such that $\phi'(U_1) = \alpha\phi'(U_2)$. Since Z_1 and Z_2 are i.i.d. this can only occur if ϕ' is a constant, which in turn would imply that ϕ must be linear. However, by construction linear functions are clearly not scaled-bounded activations and therefore for any $\rho \in (0, 1)$ it holds that

$$R'_{\phi,q^*}(\rho) < 1.$$

For $\rho \in (0, 1)$ then integrating both sides of the above inequality and applying the fundamental theorem of calculus we have

$$\int_0^\rho R'_{\phi,q^*}(t)dt < \int_0^\rho 1dt \implies R_{\phi,q^*}(\rho) - \rho < R_{\phi,q^*}(0)$$

and

$$\int_\rho^1 R'_{\phi,q^*}(t)dt < \int_\rho^1 1dt \implies \rho - R_{\phi,q^*}(\rho) < 0.$$

As $R_{\phi,q^*}(0) > 0$ then we conclude for $\rho \in (0, 1)$ that $|R_{\phi,q^*}(\rho) - \rho| < R_{\phi,q^*}(0)$. Therefore

$$|R_{\phi,q^*}(\rho) - \rho| \leq R_{\phi,q^*}(0) = \frac{\sigma_b^2}{q^*}$$

for all $\rho \in [0, 1]$ as claimed. □

The final lemma we present before proving Theorem 4.1.1 concerns the relationship between a, the size of the linear region, and q^*, the fixed point of the variance function. This lemma provides lower and upper bounds on the ratio $\frac{a}{\sqrt{q^*}}$ for all scaled-bounded activations as a function of a and σ_b^2. The fact that this is possible indicates that the particular shape of the tails of a scaled-bounded activation do not play a key role in achieving a good initialisation. In Figure 4.4 we plot these bounds, as well as example $a/\sqrt{q^*}$ ratios, for a number of activation functions as a function of a.

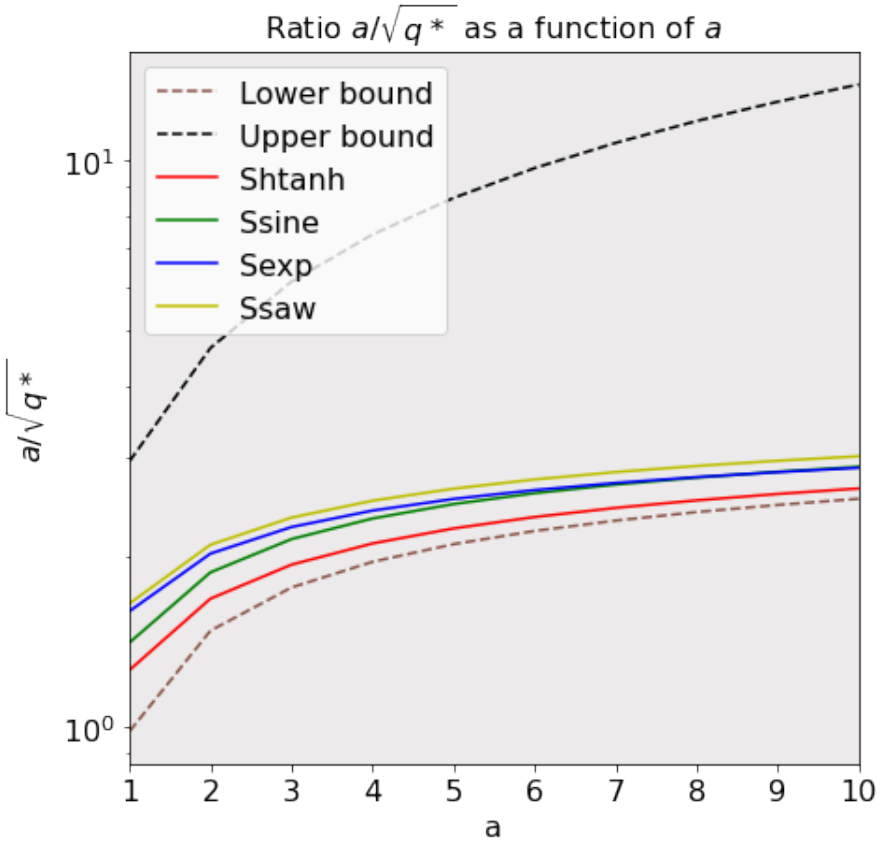


Figure 4.4: Bounds on $\frac{a}{\sqrt{q^*}}$ given in (4.30) vs. $\frac{a}{\sqrt{q^*}}$, computed numerically for the same scaled-bounded activations visualised in Figure 4.3.

Lemma 4.2.5. *Under the same conditions and assumptions as in Lemma 4.2.3, and defining* $y := \frac{\sigma_b^2}{a^2}$*, then*

$$\Lambda(y) < \frac{a}{\sqrt{q^*}} < \left(\frac{8}{\pi}\right)^{1/6} y^{-1/3}, \tag{4.30}$$

where $\Lambda(y)$ *is defined as*

$$\left(W_0\left(\frac{2}{\pi}\left(\sqrt{\frac{8}{\pi}}\exp\left(-\frac{\left(\sqrt{W_0\left(\frac{2}{\pi}y^{-2}\right)}\right)^2}{2}\right)\left(\frac{1}{\left(\sqrt{W_0\left(\frac{2}{\pi}y^{-2}\right)}\right)}+\left(\sqrt{W_0\left(\frac{2}{\pi}y^{-2}\right)}\right)\right)\right)^{-2}\right)\right)^{1/2}$$

and W_0 *denotes the principal branch of the Lambert W function.*

Proof. To derive the upper bound on $a/\sqrt{q^*}$ we study lower bounds for $V_\phi(q)$. To this end we first lower bound $\mathbb{E}[\phi(\sqrt{q}Z)^2]$. The fact that ϕ is odd implies ϕ^2 is even,

therefore

$$\begin{aligned}\mathbb{E}[\phi(\sqrt{q}Z)^2] &= 2\left(\int_0^{a/\sqrt{q}} k^2 q z^2 d\gamma(z) + \int_{a/\sqrt{q}}^{\infty} \phi(\sqrt{q}z)^2 d\gamma(z)\right) \\ &\geq 2\left(k^2 q \int_0^{a/\sqrt{q}} z^2 d\gamma(z)\right) \\ &= qk^2 \mathrm{erf}\left(\frac{a}{\sqrt{2q}}\right) - \sqrt{\frac{2}{\pi}} a k^2 \sqrt{q} \exp\left(-\frac{a^2}{2q}\right).\end{aligned}$$

To now upper bound $\mathbb{E}[\phi^{'}(\sqrt{q*}Z)^2]$ we use the fact that $|\phi'(z)| \leq k$, which implies

$$\begin{aligned}\mathbb{E}[\phi^{'}(\sqrt{q^*}Z)^2] &\leq 2k^2 \int_0^{\infty} d\gamma(z) \\ &= k^2.\end{aligned}$$

As $q^* = V_\phi(q^*)$, it therefore follows that

$$q^* \geq q^* \mathrm{erf}\left(\frac{a}{\sqrt{2q^*}}\right) - \sqrt{\frac{2}{\pi}} a \sqrt{q^*} \exp\left(-\frac{a^2}{2q^*}\right) + \sigma_b^2.$$

Rearranging and dividing by a^2 gives

$$\frac{q^*}{a^2} \mathrm{erfc}\left(\frac{a}{\sqrt{2q^*}}\right) \geq \frac{\sigma_b^2}{a^2} - \frac{\sqrt{q^*}}{a}\sqrt{\frac{2}{\pi}} \exp\left(-\frac{a^2}{2q^*}\right).$$

For typographical ease we now substitute $x = \frac{a}{\sqrt{q^*}}$ and multiply by x^2,

$$\mathrm{erfc}\left(\frac{x}{\sqrt{2}}\right) \geq x^2 \frac{\sigma_b^2}{a^2} - x\sqrt{\frac{2}{\pi}} \exp\left(-\frac{x^2}{2}\right).$$

It is known that $\mathrm{erfc}\left(\frac{x}{\sqrt{2}}\right) \leq \sqrt{\frac{2}{\pi}}\frac{1}{x} \exp\left(-\frac{x^2}{2}\right) \leq \sqrt{\frac{2}{\pi}}\frac{1}{x}$ (see e.g., [72]). Additionally it holds that $\exp\left(-\frac{x^2}{2}\right) < \frac{1}{x^2}$, this follows from the fact that $x > 2\ln(x)$, which can be proved using elementary calculus. Using these results we formulate the following inequality,

$$x^2 \frac{\sigma_b^2}{a^2} - x\sqrt{\frac{2}{\pi}}\frac{1}{x^2} < \sqrt{\frac{2}{\pi}}\frac{1}{x},$$

which simplifies to

$$\frac{a}{\sqrt{q^*}} =: x < \left(\frac{8}{\pi}\right)^{1/6} \frac{a^{2/3}}{\sigma_b^{2/3}} = \left(\frac{8}{\pi}\right)^{1/6} y^{-1/3}.$$

as claimed.

The derivation of the lower bound is more involved, we proceed by deriving an upper bound on $V_\phi(q)$ which will allow us to upper bound q^*. First, as ϕ is upper bounded by k^2a^2 then

$$\begin{aligned}\mathbb{E}[\phi(\sqrt{q}Z)^2] &\leq 2\left(\int_0^{a/\sqrt{q}} k^2 q z^2 d\gamma(z) + k^2a^2\int_{a/\sqrt{q}}^{\infty} d\gamma(z)\right) \\ &= k^2 q \text{erf}\left(\frac{a}{\sqrt{2q}}\right) - k^2\sqrt{\frac{2}{\pi}}a\sqrt{q}\exp\left(-\frac{a^2}{2q}\right) + k^2a^2\text{erfc}\left(\frac{a}{\sqrt{2q}}\right).\end{aligned}$$

A simple lower bound for $\mathbb{E}[\phi^{'}(\sqrt{q^*}Z)^2]$ is as follows,

$$\begin{aligned}\mathbb{E}[\phi^{'}(\sqrt{q^*}Z)^2] &\geq 2k^2\int_0^{a/\sqrt{q^*}}\gamma(z) \\ &= k^2\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right).\end{aligned}$$

As a result of these inequalities we may formulate the following inequality,

$$q^* \leq \frac{k^2q^*\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right) - k^2\sqrt{\frac{2}{\pi}}a\sqrt{q^*}\exp\left(-\frac{a^2}{q^*}\right) + k^2a^2\text{erfc}\left(\frac{a}{\sqrt{2q^*}}\right)}{k^2\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)} + \sigma_b^2.$$

Rearranging this expression gives

$$q^*\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right) \leq q^*\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right) - \sqrt{\frac{2}{\pi}}a\sqrt{q^*}\exp\left(-\frac{a^2}{2q^*}\right) + a^2\text{erfc}\left(\frac{a}{\sqrt{2q^*}}\right) + \sigma_b^2\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right).$$

Further simplification leads to

$$\begin{aligned}\sqrt{\frac{2}{\pi}}a\sqrt{q^*}\exp\left(-\frac{a^2}{2q^*}\right) &\leq (a^2-\sigma_b^2)\text{erfc}\left(\frac{a}{\sqrt{2q^*}}\right) + \sigma_b^2 \\ &< a^2\text{erfc}\left(\frac{a}{\sqrt{2q^*}}\right) + \sigma_b^2.\end{aligned}$$

Dividing by a^2 and making the substitution $x = \frac{a}{\sqrt{q^*}}$ we arrive at the key inequality

$$\sqrt{\frac{2}{\pi}}\frac{1}{x}\exp\left(-\frac{x^2}{2}\right) < \text{erfc}\left(\frac{x}{\sqrt{2}}\right) + \frac{\sigma_b^2}{a^2}. \tag{4.31}$$

Recalling that our objective is to lower bound the quantity $\frac{a}{\sqrt{q^*}}$, then by construction any value of x which satisfies the above inequality is a viable candidate. The challenging aspect now is to find a non-trivial candidate, in particular one that scales appropriately with a and σ_b^2. For typographical ease we now define $g(x) := \sqrt{\frac{2}{\pi}}\frac{1}{x}\exp\left(-\frac{x^2}{2}\right)$

and $h(x) := \operatorname{erfc}\left(\frac{x}{\sqrt{2}}\right) + \frac{\sigma_b^2}{a^2}$. The derivatives of each of these function are as follows,

$$g'(x) = -\sqrt{\frac{2}{\pi}} \exp\left(-\frac{x^2}{2}\right)\left(\frac{1}{x^2}+1\right)$$
$$h'(x) = -\sqrt{\frac{2}{\pi}} \exp\left(-\frac{x^2}{2}\right),$$

note that $g'(x) > h'(x)$ for all $x \in \mathbb{R}$. We now identify a candidate lower bound by considering the linear equation $u(x)$ defined by being tangent to $g(x)$ at $x = \delta := \sqrt{W_0\left(\frac{2}{\pi}y^{-2}\right)}$. Here W_0 refers to the principle branch of the Lambert W function (see e.g., [30]). The line $u(x)$ therefore has the form

$$u(x) = -\sqrt{\frac{2}{\pi}} \exp\left(-\frac{\delta^2}{2}\right)\left(\frac{1}{\delta^2}+1\right)x + \beta.$$

Observe that as $u(\delta) = y = g(\delta)$ by construction, then

$$\begin{aligned}\beta &= y + \sqrt{\frac{2}{\pi}} \exp\left(-\frac{\delta^2}{2}\right)\left(\frac{1}{\delta}+\delta\right)\\ &= \sqrt{\frac{2}{\pi}}\frac{1}{\delta} \exp\left(-\frac{\delta^2}{2}\right) + \sqrt{\frac{2}{\pi}} \exp\left(-\frac{\delta^2}{2}\right)\left(\frac{1}{\delta}+\delta\right)\\ &= \sqrt{\frac{2}{\pi}} \exp\left(-\frac{\delta^2}{2}\right)\left(\frac{2}{\delta}+\delta\right)\end{aligned}$$

We now identify a candidate lower bound γ as the solution of $g(\gamma) = \beta$, the solution of which can be expressed using the Lambert W function, $\gamma = \sqrt{W_0\left(\frac{2}{\pi}\beta^{-2}\right)}$. Consider the following compositions of functions, $\Phi_g(y) := g(\gamma(\beta(\delta(y))))$ and $\Psi_h(y) := h(\gamma(\beta(\delta(y))))$. To ensure that $\gamma \leq \frac{a}{\sqrt{q^*}}$, i.e., that γ is indeed a valid lower bound, then it must hold that $\Phi_g(y) \geq \Phi_h(y)$. We refer the reader to Figure 4.2.1 for a graphical description of this approach to identifying a candidate lower bound. Our task then is to inspect the range of $y \in \mathbb{R}_{\geq 0}$ values for which this inequality holds true. First we note that $\Phi_f(0) = \Phi_h(0) = 0$. Second we observe h, g, γ, β and δ are differentiable functions on $\mathbb{R}_{>0}$. As

$$\frac{d\delta}{dy} = -\frac{\sqrt{W_0\left(\frac{2}{\pi}y^{-2}\right)}}{y\left(W_0\left(\frac{2}{\pi}y^{-2}\right)+1\right)} < 0,$$
$$\frac{d\beta}{d\delta} = -\frac{\sqrt{\frac{2}{\pi}}\exp\left(-\frac{\delta^2}{2}\right)(\delta^4+\delta^2+2)}{\delta^2} < 0,$$
$$\frac{d\gamma}{d\beta} = -\frac{\sqrt{W_0\left(\frac{2}{\pi}\beta^{-2}\right)}}{\beta\left(W_0\left(\frac{2}{\pi}\beta^{-2}\right)+1\right)} < 0$$

then $\frac{d\gamma}{dy} = \frac{d\gamma}{d\beta}\frac{d\beta}{d\delta}\frac{d\delta}{dy} < 0$. Applying the chain rule

$$\begin{aligned}\Phi'_g(y) &= \frac{dg}{d\gamma}\frac{d\gamma}{dy} \\ &= -\sqrt{\frac{2}{\pi}}\exp\left(-\frac{\gamma^2}{2}\right)\left(\frac{1}{\gamma^2}+1\right)\frac{d\gamma}{dy} \\ &= \sqrt{\frac{2}{\pi}}\exp\left(-\frac{\gamma^2}{2}\right)\left(\frac{1}{\gamma^2}+1\right)\left|\frac{d\gamma}{dy}\right|\end{aligned}$$

and

$$\begin{aligned}\Phi'_h(y) &= \frac{dh}{d\gamma}\frac{d\gamma}{dy} \\ &= \sqrt{\frac{2}{\pi}}\exp\left(-\frac{\gamma^2}{2}\right)\left|\frac{d\gamma}{dy}\right|.\end{aligned}$$

The fact that

$$\sqrt{\frac{2}{\pi}}\exp\left(-\frac{\gamma^2}{2}\right)\left(\frac{1}{\gamma^2}+2\right)\left|\frac{d\gamma}{dy}\right| > \sqrt{\frac{2}{\pi}}\exp\left(-\frac{\gamma^2}{2}\right)\left|\frac{d\gamma}{dy}\right|,$$

which can be further simplified to

$$\left(\frac{1}{\gamma^2}+1\right) > 1,$$

holds for all $\gamma \in \mathbb{R}$ implies that $\Phi'_g(y) > \Phi'_h(y)$ for all $y \in \mathbb{R}_{>0}$. Therefore

$$\int_0^y \Phi'_g(t) - \Phi'_h(t)dt > 0$$

for all $y \in \mathbb{R}_{>0}$. Now applying the fundamental theorem of calculus

$$\begin{aligned}\int_0^y \Phi'_g(t) - \Phi'_h(t)dt &= \Phi_g(y) - \Phi_h(y) + \Phi_h(0) - \Phi_f(0) \\ &= \Phi_g(y) - \Phi_h(y)\end{aligned}$$

we conclude that $\Phi_g(y) > \Phi_h(y)$ for all $y \in \mathbb{R}_{>0}$. As a result, γ is a valid lower bound for $\frac{a^2}{\sqrt{q^*}}$ as long as $\frac{\sigma_b^2}{a^2} > 0$. Finally, to recover the statement of the theorem, we define the composite function $\Lambda : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as $\Lambda(y) := \gamma(\beta(\delta(y)))$. □

We note that while the upper bound on $a/\sqrt{q^*}$ provided by Lemma 4.2.5 is easy to interpret, the lower bound is not immediately interpretable. However, this lower bound still allows us to compute a numerical lower bound for any scaled-bounded activation as per Figure 4.4. In terms of asymptotic behaviour it is easy to check that as $y \to 0$ then $y^{-1/3}, \Lambda(y) \to \infty$, and as $y \to \infty$ then $y^{-1/3}, \Lambda(y) \to 0$. We also observe from Figure 4.4 that, at least empirically, the lower bound seems to be far

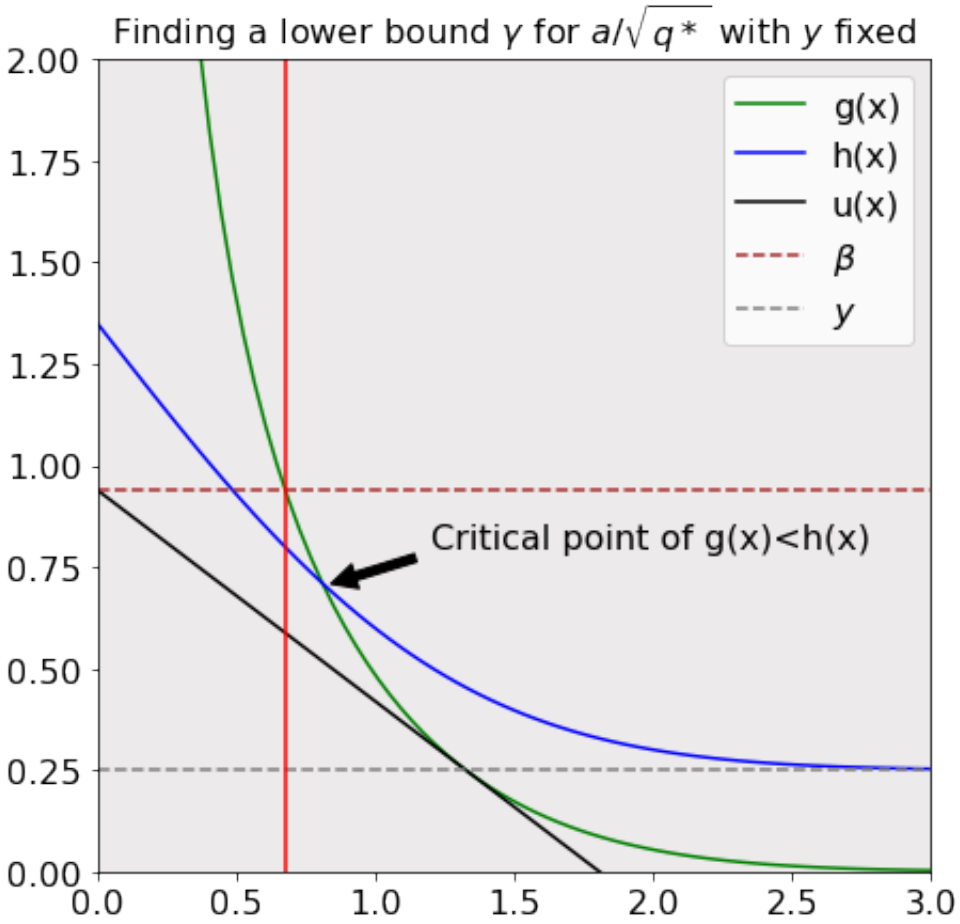


Figure 4.5: illustration of proof approach to finding a lower bound γ for $\frac{a}{\sqrt{q^*}}$ for a given value of $y = \sigma_b^2/a^2$ (in this example $y = 0.25$). The red line indicates the value on the x-axis defined to be γ. This is computed by first identifying where the line $u(x)$, which lies tangent to the point where g hits the asymptotic limit of $h(x)$ which in turn corresponds to the value of y, intercepts the y-axis, denoted as β. We define γ then to be the point at which the horizontal line at β (the upper dashed brown line) intercepts $g(x)$. To ensure that β intercepts $g(x)$ above the critical point of the inequality $g(x) < h(x)$, it is necessary to check that $h(\gamma) < g(\gamma)$ is true.

tighter: we leave it as potential future work to investigate deriving a tighter upper bound.

We are now ready to prove Theorem 4.1.1.

Proof. To derive the inequality concerning the correlation function given (4.13) we use the upper bound

$$\frac{a}{\sqrt{q^*}} < \left(\frac{8}{\pi}\right)^{1/6} \frac{a^{2/3}}{\sigma_b^{2/3}}$$

derived in Lemma 4.2.5. Squaring both sides, dividing by a^2 and multiplying by σ_b^2 then

$$\frac{\sigma_b^2}{q^*} < \left(\frac{8}{\pi}\right)^{1/3} \frac{\sigma_b^{2/3}}{a^{2/3}}.$$

To conclude we apply Lemma 4.2.4. We now turn our attention to proving the inequality concerning the moment ratio provided in equation 4.14. Analysing the lth moment,

$$\begin{aligned}\mu_l &:= \mathbb{E}[\phi'(\sqrt{q^*}Z)^{2l}] \\ &= 2\left(k^{2l}\int_0^{a/\sqrt{q^*}} d\gamma(z) + \int_{a/\sqrt{q^*}}^{\infty} \phi'(\sqrt{q^*}z)^{2l} d\gamma(z)\right).\end{aligned}$$

By assumption $|\phi'(z)| \leq k$, hence we can bound this quantity as

$$k^{2l}\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right) \leq \mu_l \leq k^{2l},$$

where the lower bound arises simply by zeroing the second integral term. It therefore follows that

$$\begin{aligned}\frac{k^4\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)}{k^4} \leq \frac{\mu_2}{\mu_1^2} &\leq \frac{k^4}{k^4\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^2}, \\ \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right) \leq \frac{\mu_2}{\mu_1^2} &\leq \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^{-2}\end{aligned}$$

Observe that $\mu_2/\mu_1^2 - 1 \leq \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^{-2} - 1$ and $1 - \mu_2/\mu_1^2 \leq 1 - \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)$. We now prove, by contradiction, that $\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^{-2} - 1 \geq 1 - \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)$. Indeed, assume that $\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^{-2} - 1 < 1 - \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)$, then

$$\begin{aligned}1 - \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^2 &< \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)\left(\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right) - \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^2\right) \\ &\leq \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)\left(1 - \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^2\right).\end{aligned}$$

As $\text{erf}\left(\frac{a}{\sqrt{2q^*}}\right) < 1$ for $\frac{a}{\sqrt{2q^*}} < \infty$ then this is a contradiction. Therefore

$$\left|\frac{\mu_2}{\mu_1^2} - 1\right| \leq \text{erf}\left(\frac{a}{\sqrt{2q^*}}\right)^{-2} - 1.$$

As erf is monotonically increasing it suffices to lower bound the quantity $\frac{a}{\sqrt{q^*}}$. The result claimed is then recovered by applying the lower bound on $\frac{a}{\sqrt{q^*}}$ derived in Lemma 4.2.5. □

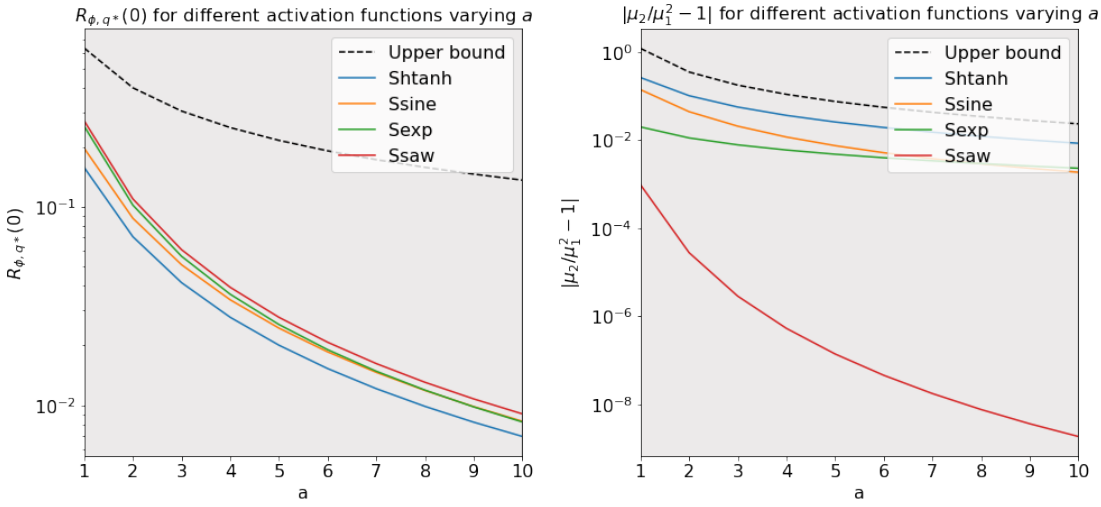


Figure 4.6: The left and right plots display the correlation and moment ratio bounds, given in Equations (4.13) and (4.14) respectively, vs. the equivalent numerically computed quantities for a variety of different activation functions $\phi \in \Omega$, as shown in Figure 4.3. We note that the right hand curve for hard saw is accurate only up to an a of 6 due to numerical precision issues arising from numerical integration steps.

4.2.2 Discussion and practical takeaways

Equation (4.13) implies, by choosing a sufficiently large relative to σ_b^2, that problems arising as a result of limits on the depth of information propagation as well as network sensitivity can be mitigated without the need for $q^*, \sigma_b^2 \to 0$. Equation (4.14) likewise implies, as long as orthogonal initialisation is used, that increasing a also mitigates the problem of model degeneracy, again without the need for $q^*, \sigma_b^2 \to 0$. Our numerics support these conclusions: in Figure 4.6 the bounds in (4.13) and (4.14) are plotted numerically with σ_b^2 fixed, and converge to 0 as a increases. Likewise, Figure 4.7 shows that increasing a moves R_{ϕ,q^*} closer to the identity. We remark that (4.2) and (4.4) where derived in the context of Gaussian initialisation. As a result one

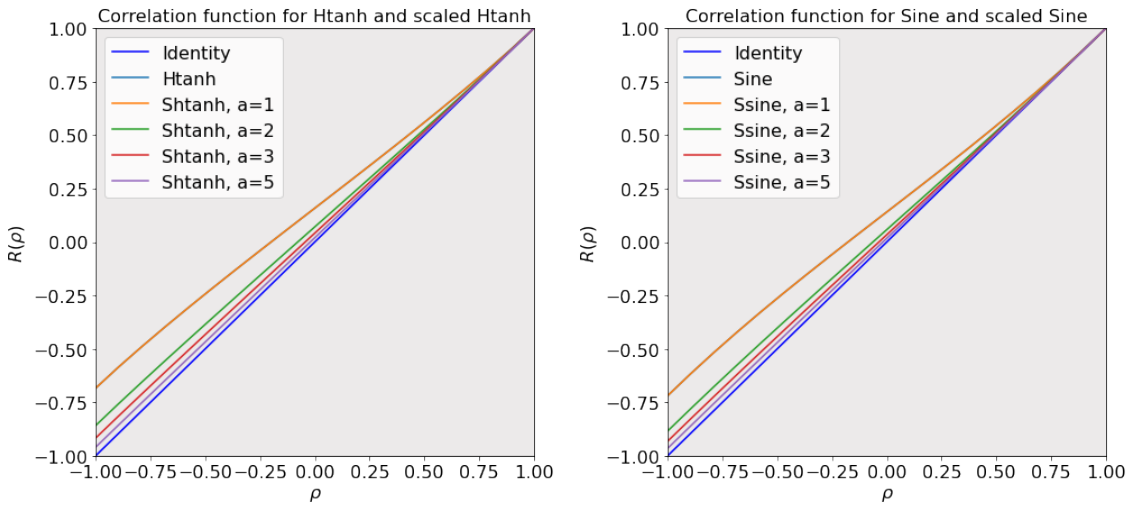


Figure 4.7: impact of scale parameter a on correlation function of scaled (or adapted) htanh (left-hand plot) and sinusoid (right-hand plot).

might question whether (4.13) is immediately applicable to orthogonally initialised networks. Indeed, although referenced to in [100, 101], the correspondence between orthogonally initialised networks and Gaussian processes is to our knowledge yet to be rigorously established. However, it seems highly likely that the same correspondence holds, an assertion supported both by empirical observation and the fact that large random orthogonal matrices are well approximated by Gaussian matrices (see e.g., [86]). We defer a detailed study of this correspondence to later work. We also note that the uniform bound we provide holds for non-negative correlations only, this is due to the fact that the upper bound on this interval, σ_b^2/q^* as per Lemma 4.2.4, is relatively easy to analyse. However, we suspect for scaled-bounded activations that letting $\sigma_b^2/a^2 \to 0$ results in a uniform convergence of the correlation function to the identity for $\rho \in [-1, 1]$. This hypothesis is supported by Figure 4.7.

We conclude this section by considering the relevance and importance of each of the four properties of scaled-bounded activation functions in regard to achieving a good initialisation in practice. Property 1), ϕ being continuous, does not seem controversial, indeed most of the commonly used activation functions are continuous. Furthermore, it seems reasonable that continuous activation functions result in a loss landscape that is easier for an optimiser to navigate, compared with that induced by non-continuous ones. Property 2), ϕ being odd, is beneficial in theory as it removes additional constant terms from (4.29) in Lemma 4.2.4. However, as per the proof of Lemma 4.2.4, it can be observed that these terms will decay exponentially fast as a grows due to the fact that the locations of the points in $\mathcal{D}$ all have magnitude at

least a. As for property 3), while the existence of a linear region around the origin is critical to our results, we suspect that boundedness is more an artefact of our proof than a necessity in practice. Assuming that ϕ is bounded simplifies certain pieces of analysis, and crucially allows us to formulate upper and lower bounds on $V_\phi(q)$, needed for the proof of Lemma 4.2.5. We hypothesise that this condition could be relaxed to $|\phi(z)| < |z|$. In regard to property 4) and contrary to the conclusion one might be inclined to draw from [59], ϕ being non-differentiable demonstrates that smoothness is not necessary for a good initialisation. Finally, the bound on the derivative of ϕ was introduced to allow us to to derive bounds on V_ϕ.

4.3 Experiments

In this section we explore, via experiments, the tension between having a large enough linear region to ensure a good initialisation, while having a small enough linear region so as to ensure that the network is sufficiently nonlinear. I would like to express my thanks to my collaborator Vinayak Abrol for his invaluable experimental expertise and contributions to this section, as well as for allowing me to include Figures 4.7 and 4.8 in this thesis.

4.3.1 Experimental setup

Across all experiments we train networks with depths $L \in \{20, 50, 100, 200\}$, with each layer having a fixed width $N = 400$, for 100 epochs on CIFAR-10. The results are averaged over 10 trials. Variance hyperparameters (σ_w^2, σ_b^2) are selected to lie on the EOC, with $\sigma_b^2 \in \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and σ_w^2 computed in in order that $\chi_1 = 1$, see (4.6). The parameters of the network are initialised according to the following two schemes.

- **Gaussian initialisation:** all parameters are drawn mutually independent of one another. The weights in each layer are identically distributed with $w_{i,j}^{(l)} \sim \mathcal{N}(0, \sigma_w^2/N)$. The biases are all identically distributed with $b_{i,j}^{(l)} \sim \mathcal{N}(0, \sigma_b^2)$.
- **Orthogonal initialisation:** all weight matrices and bias parameters are drawn mutually independent of one another. The weight matrix at each layer is drawn according to the Haar measure, i.e., uniformly, over the orthogonal group of $N \times N$ matrices such that $(\mathbf{W}^{(l)})^T \mathbf{W}^{(l)} = \sigma_w^2 \mathbf{I}_N$. The biases are all identically distributed with $b_{i,j}^{(l)} \sim \mathcal{N}(0, \sigma_b^2)$.

Finally, optimisation is performed using SGD with a batch size of 64. A learning rate of 10^{-4} was found to be provide good results across all experiments.

4.3.2 Experimental validation of the results of Section 4.2

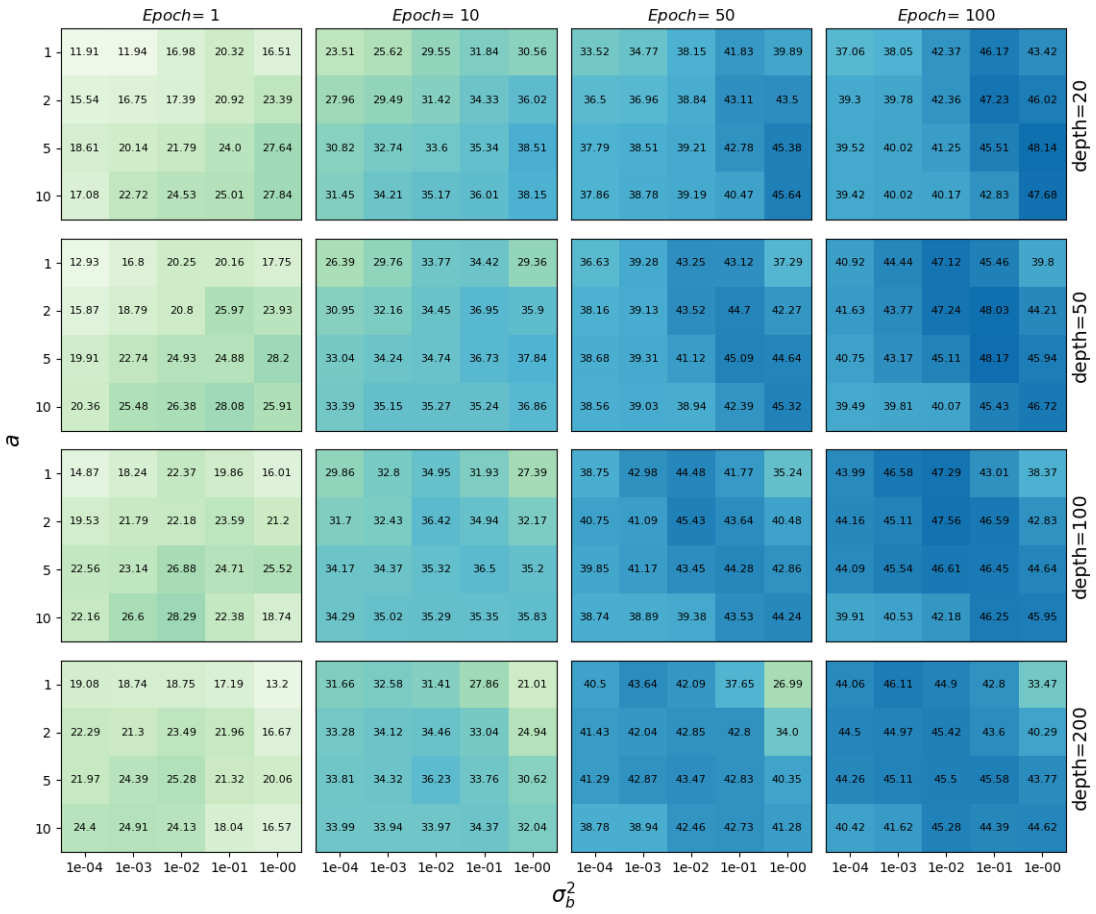
In order to test and validate the results of Section 4.2, we deploy the scaled Shtanh activation function, defined in (4.12), with $a \in \{1, 2, 5, 10\}$ and $k = 1$, and measure the test accuracy of the networks described in Section 4.3.1 at various stages of training. The results of these experiments are summarised in the heatmaps provided in Figure 4.7. As per the implications of Theorem 4.1.1, it is clear from Figure 4.7 that as σ_b^2 increases a larger but not necessarily maximal value of a gives the best results. This is illustrated by the ridges, indicating the highest test accuracy, running from top left to bottom right in the heatmap subplots. This therefore highlights the trade-off between activation functions which are linear enough to allow for a good initialisation, while not being overly linear that the network loses approximation power. We emphasise that these results are not unique to Shtanh and that the same conclusions can be drawn for other scaled-bounded activation functions [2]. As established in prior works, we also observe an advantage in using orthogonal over Gaussian initialisation. However, we also note that this performance gap is relatively small. We leave it to future work to investigate the impact of optimising over, or regularising with respect to, the set of orthogonal weight matrices so that orthogonality is preserved, at least approximately, throughout training.

We also provide some very preliminary results concerning a new training protocol, in which a scaled-bounded activation function is deployed and the value of a is decreased slowly over the first few epochs. To motivate this, observe in Figure 4.8 that, as per our theory, a large value of a always gives the best performance at initialisation. However, for reasons already discussed, an overly large value of a has a negative impact on training long term. This strategy can therefore be interpreted as computing a good initialisation for the final network. The purple curve in Figure 4.8 displays the outcome of this strategy, which appears promising over a wide range of depths and hyperparameter choices.

4.4 Conclusion and avenues for future work

In this chapter we considered the role of the activation function in avoiding certain problems at initialisation, namely limited information propagation with depth, high

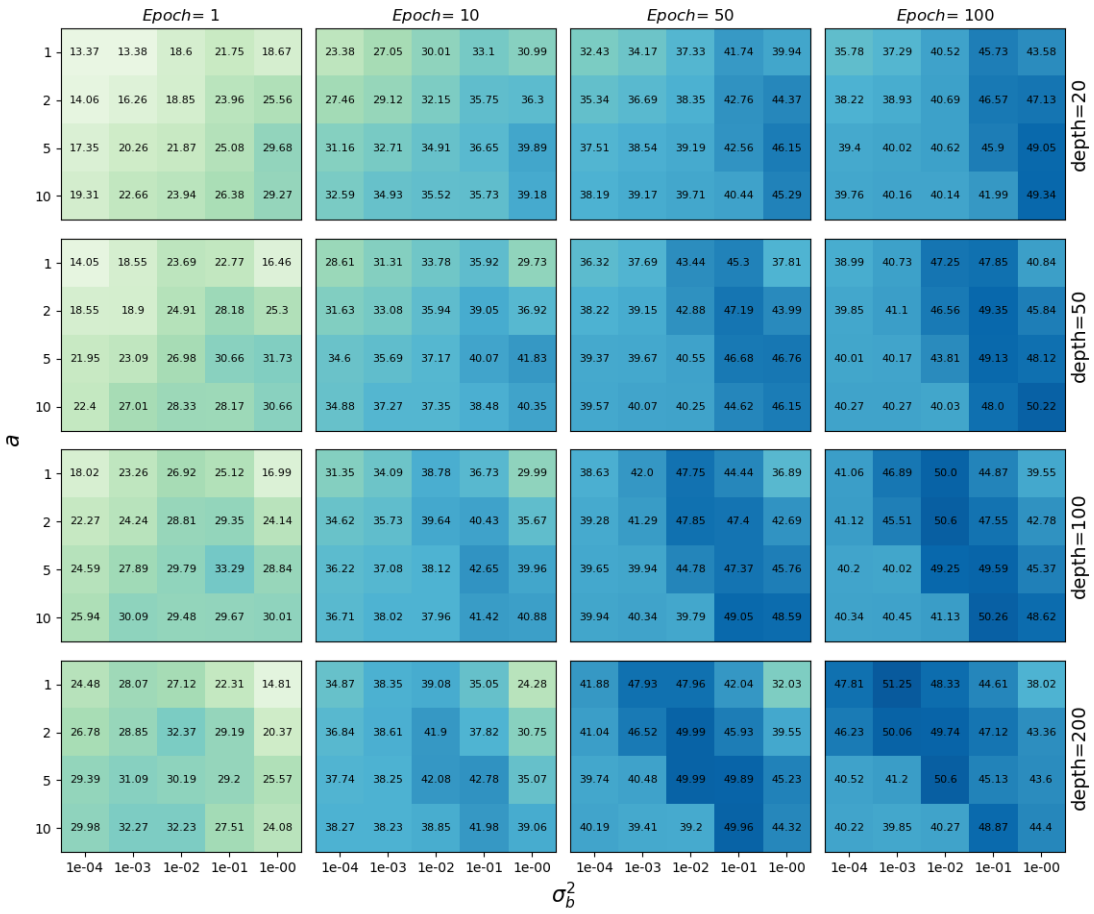
[2] Implementation and additional results can be found at https://github.com/Cross-Caps/AFLI



(a) Shtanh with Gaussian initialisation

sensitivity or insensitivity to perturbations of the input, and vanishing and exploding gradients. The first two of these we investigated by studying the dynamics of the preactivation correlations, and the third by analysing the spectrum of the input-output Jacobian of the network. Previously, these problems have been ameliorated by shrinking σ_b^2 in relation to the depth of the network. This is unsatisfactory for two reasons: firstly it results in the expected euclidean length of the activations shrinking towards 0 with depth, and second it places constraints on the initialisation regime, which can result in suboptimal training outcomes. Our theory and experiments clarify that shrinking σ_b^2 is not necessary, instead, it is possible to avoid these problems at initialisation by ensuring that the activation function deployed has a sufficiently large linear region around the origin. This work therefore provides a rigorous explanation for the observation that activation functions which approximate the identity near the origin perform well, particularly at initialisation.

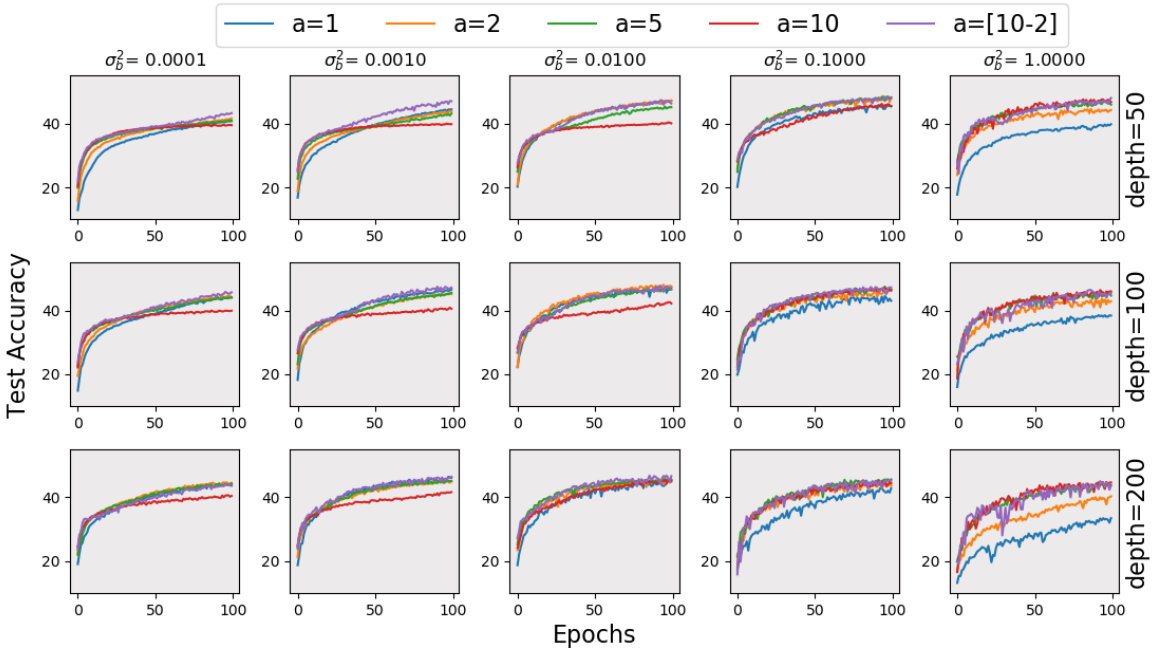
Avenues for future work include characterising more precisely how large the linear



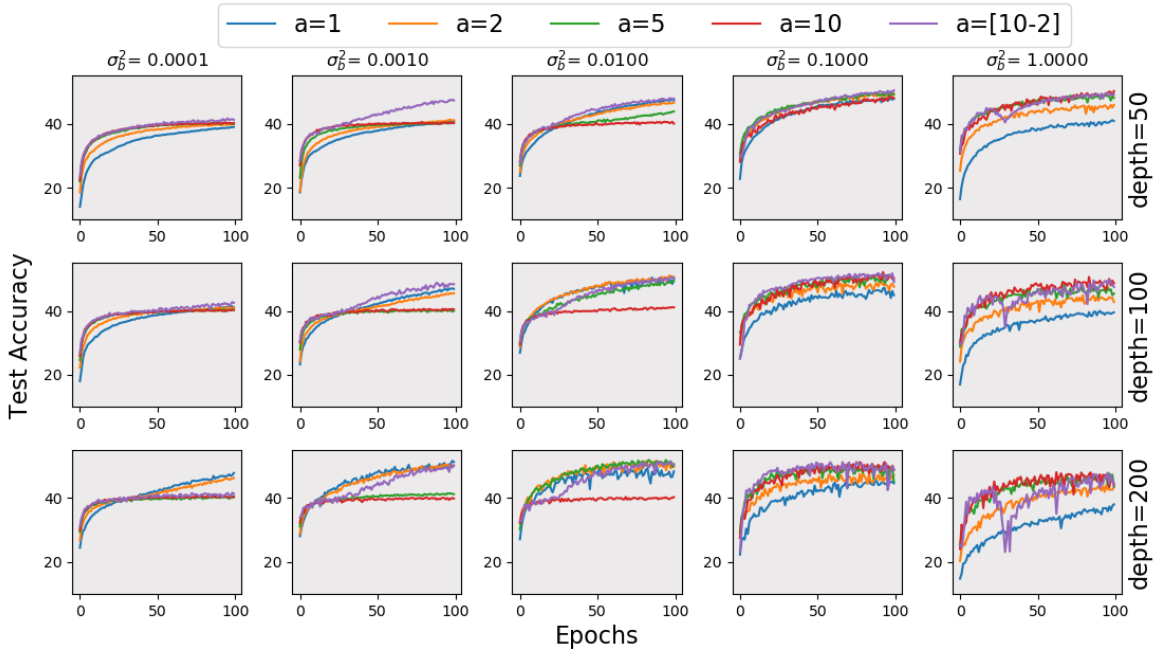
(b) Shtanh with orthogonal initialisation

Figure 4.7: test accuracy on CIFAR-10 at different stages of training. Note that all sub-heatmap plots share the same colour scale.

region needs to be for a given depth, a more comprehensive investigation as to the potential benefits of a more affine initialisation, and an analysis of the approximation capabilities of a neural network whose parameters are constrained to lie in some neighbourhood of a given initialisation point. In addition, a quantitative description of how the preactivation correlation dynamics impact training is desirable.



(a) Shtanh with Gaussian initialisation



(b) Shtanh with orthogonal initialisation

Figure 4.8: test accuracy during training on CIFAR-10. Networks are trained either with a fixed value of a or with a linearly decreasing from 10→2 over the first 30 epochs.

Bibliography

[1] Alekh Agarwal, Animashree Anandkumar, and Praneeth Netrapalli. A clustering approach to learning sparsely used overcomplete dictionaries. *IEEE Transactions on Information Theory*, 63(1):575–592, 2017.

[2] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[3] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, June 1986.

[4] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. More algorithms for provable dictionary learning. *CoRR*, abs/1401.0579, 2014.

[5] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 584–592. PMLR, June 2014.

[6] Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcomplete dictionaries. In *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *Proceedings of Machine Learning Research*, pages 779–806. PMLR, Jun 2014.

[7] Devansh Arpit, Yingbo Zhou, Hung Ngo, and Venu Govindaraju. Why regularized auto-encoders learn sparse representation? In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 136–144. PMLR, June 2016.

[8] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[9] Bubacarr Bah and Jared Tanner. On the construction of sparse matrices from expander graphs. *Frontiers in Applied Mathematics and Statistics*, 4:39, 2018.

[10] Boaz Barak, Jonathan A. Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 143–151, 2015.

[11] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28:253–263, 12 2008.

[12] L. A. Bassalygo and M. S. Pinsker. On the complexity of an optimal nonblocking commutation scheme without reorganization. *Problems Information Transmission*, 9:64–66, 1973.

[13] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, January 2009.

[14] R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, and M. J. Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 798–805, Sep. 2008.

[15] R. Berinde, P. Indyk, and M. Ruzic. Practical near-optimal sparse recovery in the l1 norm. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 198–205, 2008.

[16] Radu Berinde and Piotr Indyk. Sequential sparse matching pursuit. In *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing*, Allerton '09, page 36–43. IEEE Press, 2009.

[17] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[18] Jeffrey Blanchard, Coralia Cartis, and Jared Tanner. Compressed sensing: how sharp is the restricted isometry property? *SIAM Review*, 53, 04 2010.

[19] 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 models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

[20] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[21] E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[22] E.J. Candes and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51:4203 – 4215, 01 2006.

[23] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3), June 2011.

[24] Emmanuel J. Candes and Justin Romberg. Quantitative robust uncertainty principles and optimally sparse decompositions. *Foundations of Computational Mathematics*, 6(2):227–254, 2006.

[25] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

[26] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 659–668. ACM, 2002.

[27] J. Chen and X. Huo. Theoretical results on sparse representations of multiple-measurement vectors. *IEEE Transactions on Signal Processing*, 54(12):4634–4643, 2006.

[28] Albert Cohen, Wolfgang Dahmen, and Ronald Devore. Compressed sensing and best k -term approximation. *Journal of the American Mathematical Society*, 22(1):211–231, January 2009.

[29] James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[30] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the lambertw function. *Advances in Computational Mathematics*, 5(1):329–359, 1996.

[31] S. F. Cotter, B. D. Rao, Kjersti Engan, and K. Kreutz-Delgado. Sparse solutions to linear inverse problems with multiple measurement vectors. *IEEE Transactions on Signal Processing*, 53(7):2477–2488, 2005.

[32] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[33] Alexander G. de G. Matthews, Mark Rowland, Jiri Hron, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks, 2018.

[34] Benjamin Doerr. *Probabilistic Tools for the Analysis of Randomized Optimization Heuristics*, pages 1–87. Springer International Publishing, 2020.

[35] D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

[36] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2004.

[37] David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[38] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4829–4837, 2016.

[39] Aristides Doumas and Vassilis Papanicolaou. The coupon collector's problem revisited: generalizing the double dixie cup problem of Newman and Shepp. *ESAIM: Probability and Statistics*, 20, June 2016.

[40] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, 2008.

[41] David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 202–210, April 2014.

[42] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[43] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 907–940. PMLR, June 2016.

[44] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797, 2009.

[45] Marco Ferrante and Alessia Tagliavini. On the coupon-collector's problem with several parallel collections. *arXiv e-prints*, September 2016.

[46] G. B. Folland. *Real analysis : modern techniques and their applications*. Wiley, New York, 1999.

[47] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[48] A. C. Gilbert, Y. Zhang, K. Lee, Y. Zhang, and H. Lee. Towards Understanding the Invertibility of Convolutional Neural Networks. *ArXiv e-prints*, May 2017.

[49] R. Giryes, G. Sapiro, and A. M. Bronstein. Deep Neural Networks with Random Gaussian Weights: A Universal Classification Strategy? *IEEE Transactions on Signal Processing*, 64:3444–3457, July 2016.

[50] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, May 2010.

[51] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 9:249–256, 13–15 May 2010.

[52] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pages 315–323, April 2011.

[53] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Matrix Computations. Johns Hopkins University Press, 2012.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[55] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *J. ACM*, 56(4), July 2009.

[56] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, Dec 2017.

[57] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[58] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 571–581, December 2018.

[59] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. In *International Conference on Machine Learning (ICML)*, pages 2672–2680, June 2019.

[60] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[62] L. A. Hendricks, O. Wang, E. Shechtman, J. Sivic, T. Darrell, and B. Russell. Localizing moments in video with natural language. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5804–5813, 2017.

[63] Geoffrey E. Hinton and Zoubin Ghahramani. Geoffrey e. hinton and zoubin ghahramani, generative models for discovering sparse distributed representations, philosophical transactions of the royal society of london 352: 1177-1190.

[64] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. Master's thesis, TU Munich, July 1991.

[65] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[66] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computing*, 9(8):1735–1780, November 1997.

[67] Tom Høholdt and Heeralal Janwa. Eigenvalues and expansion of bipartite graphs. *Designs, Codes and Cryptography*, 65(3):259–273, 2012.

[68] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, March 1991.

[69] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[70] Sina Jafarpour, Weiyu Xu, Babak Hassibi, and A. Robert Calderbank. Efficient and robust compressed sensing using optimized expander graphs. *IEEE Transactions on Information Theory*, 55:4299–4308, 2009.

[71] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[72] G. K. Karagiannidis and A. S. Lioumpas. An improved approximation for the gaussian q-function. *IEEE Communications Letters*, 11(8):644–646, 2007.

[73] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[74] J. F. Kolen and S. C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. Wiley, 2001.

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105, December 2012.

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[77] M. Ledoux and M. Talagrand. *Probability in Banach Spaces*. Classics in Mathematics. Springer-Verlag Berlin Heidelberg, 2011.

[78] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, volume 19, 2007.

[79] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations (ICLR)*, pages 1–11, April 2018.

[80] Tao Li and Cha-charis Ding. *'Nonnegative Matrix Factorizations for Clustering: A Survey' in Aggarwal, Charu C. and Reddy, Chandan K. 'Data Clustering: Algorithms and Applications'*, chapter 7, pages 149–176.

[81] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196, 2015.

[82] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'08, page 1033–1040, 2008.

[83] S. Mallat and I. Waldspurger. Deep Learning by Scattering. *ArXiv e-prints*, June 2013.

[84] Stphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, Inc., USA, 3rd edition, 2008.

[85] J. L. McClelland, D. E. Rumelhart, and G. E. Hinton. Parallel distributed processing: explorations in the microstructure of cognition, Vol. 1. chapter The Appeal of Parallel Distributed Processing, pages 3–44. MIT Press, Cambridge, MA, USA, 1986.

[86] Elizabeth S. Meckes. *The Random Matrix Theory of the Classical Compact Groups*. Cambridge Tracts in Mathematics. Cambridge University Press, 2019.

[87] R. Mendoza-Smith, J. W. Tanner, and F. Wechsung. A robust parallel algorithm for combinatorial compressed sensing. *IEEE Transactions on Signal Processing*, 66(8):2167–2177, 2018.

[88] Rodrigo Mendoza-Smith and Jared Tanner. Expander 0-decoding. *Applied and Computational Harmonic Analysis*, 45(3):642 – 667, 2018.

[89] Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. When and why are deep networks better than shallow ones? *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017.

[90] M. Mishali and Y. C. Eldar. Reduce and boost: Recovering arbitrary sets of jointly sparse vectors. *IEEE Transactions on Signal Processing*, 56(10):4692–4702, 2008.

[91] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996.

[92] L. T. Nguyen, J. Kim, and B. Shim. Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237, 2019.

[93] Bruno A. Olshausen and David J. Fieldt. Sparse coding with an overcomplete basis set: a strategy employed by v1. *Vision Research*, 37:3311–3325, 1997.

[94] Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. In *International Conference on Learning Representations (ICLR)*, April 2018.

[95] V. Papyan, J. Sulam, and M. Elad. Working Locally Thinking Globally: Theoretical Guarantees for Convolutional Sparse Coding. *IEEE Transactions on Signal Processing*, 65:5687–5701, November 2017.

[96] Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *Journal of Machine Learning Research*, 18(83):1–52, 2017.

[97] Vardan Papyan, Yaniv Romano, Michael Elad, and Jeremias Sulam. Convolutional dictionary learning via local processing. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5306–5314, 2017.

[98] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6:90–105, 2004.

[99] Jeffrey Pennington and Yasaman Bahri. Geometry of neural network loss surfaces via random matrix theory. In *International Conference on Machine Learning (ICML)*, pages 2798–2806, 2017.

[100] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1924–1932, April 2018.

[101] Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: Theory and practice. In *International Conference on Neural Information Processing Systems (NeurIPS)*, page 4788–4798, December 2017.

[102] Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[103] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 3368–3376, 2016.

[104] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Chang, and T. Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019.

[105] L R. Welch. Lower bounds on the maximum cross correlation of signals. *IEEE Transactions on Information Theory*, 20:397 – 399, 06 1974.

[106] Marc'aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems 20*, pages 1185–1192. 2008.

[107] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[108] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, April 2014.

[109] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

[110] K. Schnass and P. Vandergheynst. Average performance analysis for thresholding. *IEEE Signal Processing Letters*, 14(11):828–831, Nov 2007.

[111] Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *International Conference on Learning Representations (ICLR)*, April 2017.

[112] Daniel Spielman, Huan Wang, and John Wright. Exact recovery of sparsely-used dictionaries. *IJCAI International Joint Conference on Artificial Intelligence*, 23, June 2012.

[113] Rupesh Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2377–2385, 2015.

[114] Wolfgang Stadje. The collector's problem with group drawings. *Advances in Applied Probability*, 22, 12 1990.

[115] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere I: Overview and the geometric picture. *IEEE Transactions on Information Theory*, 63(2):853–884, 2017.

[116] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere II: Recovery by riemannian trust-region method. *IEEE Transactions on Information Theory*, 63(2):885–914, 2017.

[117] Terence Tao and Van H. Vu. *Additive Combinatorics*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2006.

[118] Matus Telgarsky. benefits of depth in neural networks. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539. PMLR, June 2016.

[119] A. M. Tillmann and M. E. Pfetsch. The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing. *IEEE Transactions on Information Theory*, 60(2):1248–1259, 2014.

[120] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. *A Deeper Look at Dataset Bias in Domain Adaptation in Computer Vision Applications*, pages 37–55. 2017.

[121] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, December 2017.

[122] René Vidal. A tutorial on subspace clustering.

[123] W. Xu and B. Hassibi. Efficient compressive sensing with deterministic guarantees using expander graphs. In *2007 IEEE Information Theory Workshop*, pages 414–419, 2007.

[124] Jong Chul Ye. Compressed sensing mri: a review from signal processing perspective. *BMC Biomedical Engineering*, 1(1):8, 2019.

[125] Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *ICML Workshop on Deep Learning*.

[126] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019.

[127] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. L. Yoshua Bengio, and A. Courville. Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks. *ArXiv e-prints*, January 2017.

[128] Yuting Zhang, Kibok Lee, and Honglak Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 612–621. PMLR, June 2016.

[129] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: Algorithms and applications. *IEEE Access*, 3:490–530, 2015.

Appendices

Appendix A

Supporting theorems and results

A.1 Chapter 3

The following Rademacher concentration inequality plays an important role in the proof of Theorem 3.3.1.

Theorem A.1.1 (Rademacher concentration [77])**.** *Let α be an arbitrary real vector and ε a random vector whose elements are independent Rademacher random variables. Then for all $t \in \mathbb{R}_{>0}$*

$$P\left(|\sum_i \varepsilon_i \alpha_i| > t\right) \leq 2\exp\left(-\frac{t^2}{2\|\alpha\|_2^2}\right). \tag{A.1}$$

A.2 Chapter 4

For the sake of clarity and completeness we recall here two well known Lemmas concerning Lebesgue integrals. Both can be proved using Lebesgue's dominated convergence theorem and the mean value theorem (See e.g., Chapter 2 of [46]).

Lemma A.2.1. *Let $(X, \mathcal{F}, \mu)$ be a measure space and $Y \subset \mathbb{R}$ an open interval. Consider a function $f : X \times Y \to \mathbb{R}$ such that the following are true.*

1. *For each $y \in Y$ then the function $f_y : X \to \mathbb{R}$ with $f_y(x) := f(x, y)$ satisfies $f_y(x) \in L^1(X, \mathcal{F}, \mu)$.*
2. *For each $y \in Y$ then $\lim_{y' \to y} f(x, y') = f(x, y)$ almost everywhere in X.*
3. *For each $y \in Y$ there exists an open interval K, with $y \in K$, and a $g_K(x) \in L^1(X, \mathcal{F}, \mu)$ such that*

$$|f(x, y')| \leq g_K(x)$$

for all $y' \in K$.

Then the function $F : Y \to \mathbb{R}$ *with* $F(y) := \int_X f(x,y)d\mu$ *is continuous on* Y.

Lemma A.2.2. *Let* $(X, \mathcal{F}, \mu)$ *be a measure space and* $Y \subset \mathbb{R}$ *an open interval. Consider a function* $f : X \times Y \to \mathbb{R}$ *such that the following are true.*

1. *For each* $y \in Y$ *then the function* $f_y : X \to \mathbb{R}$ *with* $f_y(x) := f(x,y)$ *satisfies* $f_y(x) \in L^1(X, \mathcal{F}, \mu)$.

2. *For each* $y \in Y$ *then* $\frac{\partial f}{\partial y}(x,y)$ *exists almost everywhere in* X.

3. *For each* $y \in Y$ *there exists an open interval* K, *with* $y \in K$, *and a* $g_K(x) \in L^1(X, \mathcal{F}, \mu)$ *such that*
$$|\frac{\partial f}{\partial y}(x,y')| \leq g_K(x)$$
for all $y' \in K$.

Then the function $F : Y \to \mathbb{R}$ *with* $F(y) := \int_X f(x,y)dx$ *is differentiable on* Y *with*

$$F'(y) = \int_X \frac{\partial f}{\partial y}(x,y)d\mu. \tag{A.2}$$