Underspecification in Deep Learning

by

Mary Phuong

May, 2021

A thesis submitted to the
Graduate School
of the
Institute of Science and Technology Austria
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Committee in charge:

Jozsef Csicsvari, Chair
Christoph H. Lampert
Dan Alistarh
Novi Quadrianto
Xiaojin Zhu

Institute of Science and Technology

The thesis of Mary Phuong, titled *Underspecification in Deep Learning*, is approved by:

Supervisor: Christoph H. Lampert, IST Austria, Klosterneuburg, Austria

Signature: _____

Committee Member: Dan Alistarh, IST Austria, Klosterneuburg, Austria

Signature: _____

Committee Member: Novi Quadrianto, University of Sussex, Brighton, UK

Signature: _____

Committee Member: Xiaojin Zhu, University of Wisconsin-Madison, Madison, USA

Signature: _____

Defense Chair: Jozsef Csicsvari, IST Austria, Klosterneuburg, Austria

Signature: _____

Signed page is on file

IST Austria Thesis, ISSN: 2663-337X

I hereby declare that this thesis is my own work and that it does not contain other people's work without this being so stated; this thesis does not contain my previous work without this being stated, and the bibliography contains all the literature that I used in writing the dissertation.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee, and that this thesis has not been submitted for a higher degree to any other university or institution.

I certify that any republication of materials presented in this thesis has been approved by the relevant publishers and co-authors.

Signature: _____

Mary Phuong
May, 2021

Signed page is on file

Abstract

Deep learning is best known for its empirical success across a wide range of applications spanning computer vision, natural language processing and speech. Of equal significance, though perhaps less known, are its ramifications for learning theory: deep networks have been observed to perform surprisingly well in the high-capacity regime, aka the overfitting or underspecified regime. Classically, this regime on the far right of the bias-variance curve is associated with poor generalisation; however, recent experiments with deep networks challenge this view.

This thesis is devoted to investigating various aspects of underspecification in deep learning. First, we argue that deep learning models are underspecified on two levels: a) any given training dataset can be fit by many different functions, and b) any given function can be expressed by many different parameter configurations. We refer to the second kind of underspecification as *parameterisation redundancy* and we precisely characterise its extent. Second, we characterise the implicit criteria (the *inductive bias*) that guide learning in the underspecified regime. Specifically, we consider a nonlinear but tractable classification setting, and show that given the choice, neural networks learn classifiers with a large margin. Third, we consider learning scenarios where the inductive bias is not by itself sufficient to deal with underspecification. We then study different ways of 'tightening the specification': i) In the setting of representation learning with variational autoencoders, we propose a hand-crafted regulariser based on mutual information. ii) In the setting of binary classification, we consider *soft-label* (real-valued) supervision. We derive a generalisation bound for linear networks supervised in this way and verify that soft labels facilitate fast learning. Finally, we explore an application of soft-label supervision to the training of multi-exit models.

Acknowledgements

First and foremost, I want to thank Christoph, for a lot of things: for his patient, empathetic and kind mentorship, for providing the space and encouragement I needed to explore and pursue my interests, for his understanding and support in difficult times. I deeply appreciate his readiness to help whenever I needed help (even when I didn't know I did). This work would not have been possible without him.

I am grateful to my committee members Dan, Jerry and Novi for providing valuable feedback and advice at the qualifying exam, throughout my PhD, as well as on earlier drafts of the thesis. I additionally thank Dan for organising the deep learning theory seminar, which helped shape my research interests, and Novi for an opportunity to discuss my work with his group.

Parts of this thesis are based on work I did as an intern at Microsoft Research Cambridge. I want to thank Ryota, Nate and in particular Sebastian for hosting and guiding me. I thank Christoph for making the internship possible.

I specially want to thank (ex-)members of the MLCV group and other friends of third floor science, for companionship and encouragement on the PhD journey. I have learnt a lot from their feedback and perspectives. My thanks go to Alex, Alex, Alex, Amélie, Asya, Bernd, Elias, Jan, Jonny, Mirco, Niko, Paul, Viktor and Wiktor.

I am also grateful to my 'adoptive group' on the second floor, namely Christian, Honza, Isma, Kuba, Pepa and Rai, for companionship during the pandemic and for many stimulating lunch discussions.

I thank Vlad, Elisabeth, Astrid and Ksenja for admin support, and Janos and Stephan for accommodating my wildest wishes in the realm of scientific computing.

This work was in parts funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no. 308036.

$***$

A na záver ďakujem JeFovi, ktorý ma v mojich pseudo-intelektuálnych výpravách vždy naplno podporuje, či už po ľudskej alebo kulinárskej stránke.

Tento stotridsať-stranový skvost je venovaný jemu.

About the Author

Mary Phuong completed a BSc in applied mathematics at Comenius University, Bratislava, and an MSc in quantitative finance at the Vienna University of Economics and Business. She has worked in bio-statistics and medical image analysis before joining IST Austria in September 2016. At IST, she was lucky to be supervised by Christoph Lampert. In her research, she developed new methods for supervised and unsupervised learning, and later on she strove to understand existing methods theoretically. Her current area of interest is the theory of deep learning.

List of Publications

1. Mary Phuong, Max Welling, Nate Kushman, Ryota Tomioka, and Sebastian Nowozin. The mutual autoencoder: Controlling information in latent code representations. 2018.

2. Mary Phuong and Christoph H. Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learing (ICML)*, 2019a.

3. Mary Phuong and Christoph H. Lampert. Distillation-based training for multi-exit architectures. In *International Conference on Computer Vision (ICCV)*, 2019b.

4. Mary Phuong and Christoph H. Lampert. Functional vs. parametric equivalence of ReLU networks. In *International Conference on Learning Representations (ICLR)*, 2020.

5. Mary Phuong and Christoph H. Lampert. The inductive bias of ReLU networks on orthogonally separable data. In *International Conference on Learning Representations (ICLR)*, 2021.

Author contributions

This section specifies the contributions of each author to the publications that make up this thesis (listed above). Authors are referred to by their initials.

For [Phuong et al., 2018]:

- SN proposed the problem, conceived the method and designed the experiments.
- MP implemented the method and performed the experiments. She identified the method's shortcomings and developed it.
- Throughout, SN, NK and RT advised MP on potential paths forward.
- MP and SN wrote the paper.

For [Phuong and Lampert, 2019b]:

- MP proposed the problem and conceived the method.
- CHL designed the experiments.
- MP implemented the method and performed the experiments. She identified the method's shortcomings and developed it.
- Throughout, CHL advised MP on potential paths forward.
- MP and CHL wrote the paper.

For [Phuong and Lampert, 2019a, 2020, 2021]:

- MP proposed the problem and conceived the analysis. She formulated and proved the theorems. Where relevant, she designed additional experiments and performed them.
- Throughout, CHL advised MP on potential paths forward. He suggested important related work.
- MP wrote the paper and CHL edited it.

Table of Contents

List of Abbreviations

DL Deep learning. 1, 3

ERM Empirical risk minimisation. 1–3

MAE Mutual autoencoder. 30

MI Mutual information. 30

MLE Maximum likelihood estimation. 29

MSDNet Multi-scale dense-net. 55

NAS Neural architecture search. 55

VAE Variational autoencoder. 29

CHAPTER 1

Introduction

This is a thesis about (certain aspects of) *statistical learning*, or learning by observation. A *statistical learner* aims to master some general class of problems, but is only given solutions to a few instances of the problem, the *training data*. The learner's key concern is therefore how to *generalise*, i.e. how to use the solutions to previously encountered problems to correctly infer the solution to a new problem. Many real life situations involve statistical learning, for example estimating how late the bus will be based on our past experience, designing a new drug based on the properties of existing drugs, or learning to anticipate a friend's sentences based on past conversations.

The majority of natural and artificial statistical learners instantiate an approach to learning called *empirical risk minimisation (ERM)* [Vapnik, 1998, Shalev-Shwartz and Ben-David, 2014]. An *ERM algorithm* consists of a set of potential problem-solution mappings (*hypotheses*), and a criterion (*loss function*) for evaluating how well a given hypothesis performs on a given data point. Given a database of problem-solution pairs (*training data*), an ERM algorithm returns the hypothesis with the best performance (lowest loss), averaged over the data. Many classical and contemporary learning methods perform (approximate) ERM, and the framework is backed by well-established theory.

In this thesis, we are specifically interested in *deep learning (DL)*, a form of ERM where the hypothesis class is given by large neural networks and the search for low-risk hypotheses is carried out by local descent. Deep learning is interesting, because its behaviour often eludes common sense and theoretical understanding: single-hidden-layer networks suffice to approximate any continuous function [Cybenko, 1989, Hornik et al., 1989], yet in practice deeper is better; the risk landscape is highly non-convex, yet it can be minimised by local-descent methods; the capacity of the model class is immense, yet deep networks tend not to overfit [Zhang et al., 2017].

We focus here on the surprising generalisation performance of deep nets in the high-capacity regime [Belkin et al., 2019, Neyshabur et al., 2019, Novak et al., 2018]. Classical theory decomposes the generalisation error into *approximation error* and *statistical error*. While the approximation error becomes vanishingly small for large hypothesis classes (they more likely contain a hypothesis close to the truth), the statistical error of a large hypothesis class more than makes up for it, resulting in the famous U-shaped bias-variance curve. In this view, it is impossible to learn large hypothesis classes, essentially because of too many 'spurious fits' – hypotheses that by chance work well on the training data but poorly

outside of it. This unfortunate place on the bias-variance curve is called the *overfitting* or *underspecified* regime.

Underspecification is inherent in deep learning. It is common that there are multiple hypotheses with zero empirical risk, yet differing markedly in their predictions outside the training data. In such a setting, any choice seems arbitrary, but in practice deep nets can systematically 'guess' a hypothesis that performs well. This is often explained in terms of a favourable *inductive bias*, an implicit criterion for choosing among equal-risk hypotheses.

The work presented in this thesis addresses various aspects of underspecification in deep learning, be it understanding its extent (Chapter 2), or characterising the implicit bias in a tractable setting (Chapter 3). Although the implicit bias is often favourable, it is not always so and one may wish to override it. Hence, we also discuss other ways of dealing with underspecification: hand-crafting a regulariser (Chapter 4), or using more informative supervision (Chapter 5). We consider an independent application of strong supervision in Chapter 6. The remaining sections of this chapter are devoted to introducing common background and notation.

1.1 Supervised learning

This section formalises *supervised learning*, the most common form of statistical learning. We assume this setting in the entire thesis with the exception of Chapter 4.

Learning. Denote by $\mathcal{X}$ the set of potential *inputs* and by $\mathcal{Y}$ the set of potential *outputs*. We assume there is a 'true' correspondence between inputs and outputs, represented by a probability distribution P on $\mathcal{X} \times \mathcal{Y}$, which is unknown to us but which we wish to approximate. Usually this is done by searching over input-output mappings, or *hypotheses* $h : \mathcal{X} \to \mathcal{Y}$, for one that matches the data well.

Formally, a *supervised learning problem* consists of the distribution P over $\mathcal{X} \times \mathcal{Y}$, and an evaluation criterion r that maps a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ and an *example* $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ to a real value $r(h, \mathbf{x}, y)$ measuring how well h performs on the example $(\mathbf{x}, y)$. The goal of learning is to find a hypothesis h that minimises the *test risk*,

$$\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim P} [r(h, \mathbf{x}, y)], \tag{1.1}$$

where it is assumed that r is known to the learner but P is not. In order to learn about the distribution P, the learner has access to a *training dataset* $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ sampled iid from P. The learner can then use the dataset to inform its choice of h.

Empirical risk minimisation. If the learner considers only members of some *hypothesis class* $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ and makes its choice by minimising the average risk over the data,

$$\arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} r(h, \mathbf{x}_i, y_i), \tag{1.2}$$

we say it is an *empirical risk minimiser* (ERM).

In practice, few algorithms perform the minimisation globally or on the empirical risk per se; it is more common to employ iterative local optimisation techniques on some well-behaved (continuous, differentiable) proxy of the risk called a *loss function*. We consider such algorithms as instantiations of ERM as well.

1.2 Deep learning

Deep learning is a form of ERM where the hypothesis class consists of large neural networks. In this thesis, we study *fully-connected ReLU networks* as a more tractable model of deep nets in general. We denote by ρ the ReLU function, the most commonly used nonlinearity: $\rho(\mathbf{u})_i = \max\{0, u_i\}$ for $i \in [\dim(\mathbf{u})]$, where the subscript index denotes the corresponding vector element.

ReLU network. Let $\mathcal{X} \subseteq \mathbb{R}^{d_0}$ with $d_0 \geq 2$ be a nonempty open set, and let $\boldsymbol{\theta} \triangleq (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, \mathbf{b}_L)$ be the network's parameters, with *weights* $\mathbf{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$, *biases* $\mathbf{b}_l \in \mathbb{R}^{d_l}$, and $d_L = 1$. We denote the corresponding ReLU network by $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}$, where

$$h_{\boldsymbol{\theta}} \triangleq h^L_{\boldsymbol{\theta}} \circ \rho \circ h^{L-1}_{\boldsymbol{\theta}} \circ \cdots \circ \rho \circ h^1_{\boldsymbol{\theta}}, \tag{1.3}$$

and $h^l_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{W}_l \mathbf{x} + \mathbf{b}_l$. For $1 \leq l \leq k \leq L$, we also introduce notation for truncations of the network,

$$h^{l:k}_{\boldsymbol{\theta}} \triangleq h^k_{\boldsymbol{\theta}} \circ \rho \circ h^{k-1}_{\boldsymbol{\theta}} \cdots \circ \rho \circ h^l_{\boldsymbol{\theta}}. \tag{1.4}$$

We will omit the subscript $\boldsymbol{\theta}$ when it is clear from the context.

In some situations, it is more convenient to consider ReLU networks without biases, in which case we set the biases to zero, $\mathbf{b}_l = \mathbf{0}$ for all $l \in [L]$, and do not consider them as parameters, $\boldsymbol{\theta} \triangleq (\mathbf{W}_1, \ldots, \mathbf{W}_L)$.

Training. In deep learning, ERM is typically performed by numerical optimisation of some differentiable loss function $\ell : \boldsymbol{\Theta} \to \mathbb{R}$ of the form

$$\ell(\boldsymbol{\theta}) \triangleq \sum_{i=1}^{n} \ell_i(h_{\boldsymbol{\theta}}(\mathbf{x}_i)), \tag{1.5}$$

where $\ell_i(u)$ measures how well the prediction u matches the data i and varies based on the problem. We specify the details of ℓ_i in the individual chapters.

Popular numerical optimisers in DL are gradient-based methods, notably (stochastic) gradient descent and variants based on adaptive normalisation and/or acceleration. In our theoretical analyses we assume optimisation by *gradient flow*, the continuous idealisation of gradient descent (its limit as the step size goes to zero),

$$\frac{\partial \boldsymbol{\theta}(t)}{\partial t} = -\nabla \ell(\boldsymbol{\theta}(t)), \tag{1.6}$$

where $t \in [0, \infty)$ is the time variable indexing the evolution of the parameters $\boldsymbol{\theta}$ as training progresses. The details may differ from chapter to chapter and we point them out where necessary.

CHAPTER 2

Functional vs. parametric equivalence of ReLU networks

As we alluded to in the introduction, deep learning differs from preceding machine learning paradigms mainly in the extent of underspecification, i.e. in the number of redundant degrees of freedom. First, a typical DL hypothesis class is so large that any given dataset can be fit in many different ways; in other words, it is compatible with many different members of the hypothesis class. Second, the parameterisation of deep networks is such that, additionally, any given member of the hypothesis class can be expressed by many different parameter configurations. We refer to the latter phenomenon as *parameterisation redundancy* and study it in depth in this chapter.

Specifically, consider feed-forward ReLU networks with weight matrices $\mathbf{W}_1, \dots, \mathbf{W}_L$, and biases $\mathbf{b}_1, \dots, \mathbf{b}_L$. For any given parameter vector $\boldsymbol{\theta} \triangleq (\mathbf{W}_1, \mathbf{b}_1, \dots, \mathbf{W}_L, \mathbf{b}_L)$, we wish to identify all the other parameter vectors that represent the same function. We formalise these as transformations of the parameter space which preserve the output behaviour of the network $h(\mathbf{x}) = \mathbf{W}_L \rho(\mathbf{W}_{L-1} \rho(\dots \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1 \dots) + \mathbf{b}_{L-1}) + \mathbf{b}_L$ for all inputs $\mathbf{x}$ in some domain $\mathcal{X}$. Two such transformations are known for feed-forward ReLU architectures:

1. Permutation of units (neurons) within a layer, i.e. for some permutation matrix $\mathbf{P}$,

$$\mathbf{W}_l \leftarrow \mathbf{P}\mathbf{W}_l, \qquad \mathbf{b}_l \leftarrow \mathbf{P}\mathbf{b}_l, \tag{2.1}$$

$$\mathbf{W}_{l+1} \leftarrow \mathbf{W}_{l+1}\mathbf{P}^{-1}. \tag{2.2}$$

2. Positive rescaling of all incoming weights of a unit coupled with inverse rescaling of its outgoing weights. Applied to a whole layer, with potentially different scaling factors arranged into a diagonal matrix $\mathbf{M}$, this can be written as

$$\mathbf{W}_l \leftarrow \mathbf{M}\mathbf{W}_l, \qquad \mathbf{b}_l \leftarrow \mathbf{M}\mathbf{b}_l, \tag{2.3}$$

$$\mathbf{W}_{l+1} \leftarrow \mathbf{W}_{l+1}\mathbf{M}^{-1}. \tag{2.4}$$

In this chapter, we will show (for architectures with non-increasing widths) that permutation and rescaling are in fact *the only* function-preserving weight transformations. For any eligible architecture, we explicitly construct a neural network such that any other network that implements the same function can be obtained from the original one by the application of permutation and rescaling. Stated formally:

Theorem 2.1. *Consider a bounded open nonempty domain $\mathcal{X} \subseteq \mathbb{R}^{d_0}$ and any architecture $(d_0, \dots, d_L)$ with $d_0 \geq d_1 \geq \cdots \geq d_{L-1} \geq 2$, $d_L = 1$. For this architecture, there exists a ReLU network $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}$, or equivalently a setting of the weights $\boldsymbol{\theta} \triangleq (\mathbf{W}_1, \mathbf{b}_1, \dots, \mathbf{W}_L, \mathbf{b}_L)$, such that for any 'general' ReLU network $h_{\boldsymbol{\eta}} : \mathcal{X} \to \mathbb{R}$ (with the same architecture) satisfying $h_{\boldsymbol{\theta}}(\mathbf{x}) = h_{\boldsymbol{\eta}}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$, there exist permutation matrices $\mathbf{P}_1, \dots \mathbf{P}_{L-1}$, and positive diagonal matrices $\mathbf{M}_1, \dots, \mathbf{M}_{L-1}$, such that*

$$\begin{aligned} \mathbf{W}_1 &= \mathbf{M}_1 \mathbf{P}_1 \mathbf{W}'_1, & \mathbf{b}_1 &= \mathbf{M}_1 \mathbf{P}_1 \mathbf{b}'_1, & & \\ \mathbf{W}_l &= \mathbf{M}_l \mathbf{P}_l \mathbf{W}'_l \mathbf{P}_{l-1}^{-1} \mathbf{M}_{l-1}^{-1}, & \mathbf{b}_l &= \mathbf{M}_l \mathbf{P}_l \mathbf{b}'_l, & l &\in \{2, \dots, L-1\}, \\ \mathbf{W}_L &= \mathbf{W}'_L \mathbf{P}_{L-1}^{-1} \mathbf{M}_{L-1}^{-1}, & \mathbf{b}_L &= \mathbf{b}'_L, & & \end{aligned} \tag{2.5}$$

where $\boldsymbol{\eta} \triangleq (\mathbf{W}'_1, \mathbf{b}'_1, \dots, \mathbf{W}'_L, \mathbf{b}'_L)$ are the parameters of $h_{\boldsymbol{\eta}}$.

In the above, 'general' networks is a class of networks meant to exclude degenerate cases. We give a more precise definition in Section 2.2; for now it suffices to note that almost all networks are general.

The result shows there are only two ways in which parameterisation redundancy can systematically arise.

In the following sections, we delve deeper into the proof, which relies on a geometric understanding of prediction surfaces of ReLU networks. These surfaces are piece-wise linear functions, with non-differentiabilities or 'folds' between linear regions. It turns out that folds carry a lot of information about the parameters of a network, so much in fact, that some networks are uniquely identified (up to permutation and rescaling) by the function they implement. In the following sections, we introduce in more detail the concept of a fold-set and describe its geometric structure for a subclass of ReLU networks. The chapter culminates in a proof sketch of the main result. The full proof, including proofs of intermediate results, is included in Appendix A.

2.1 Related work

The functional equivalence of neural networks is a well-researched topic in classical connectionist literature. The problem was first posed by Hecht-Nielsen [1990], and soon resolved for feed-forward networks with the tanh activation function by Chen et al. [1993], who showed that any smooth transformation of the weight space that preserves the function of all neural networks is necessarily a composition of permutations and sign flips. For the same class of networks, Fefferman and Markel [1994] showed a somewhat stronger result: knowledge of the input-output mapping of a neural network determines both its architecture and its weights, up to permutations and sign flips. Similar results have been proven for single-layer networks with a saturating activation function such as sigmoid or RBF [Kůrková and Kainen, 1994], as well as single-layer recurrent networks with a smooth activation function [Albertini and Sontag, 1993a,b].

To the best of our knowledge, no such theoretical results exist for networks with the ReLU activation, which is non-saturating, asymmetric and non-smooth. Broadly related is the recent work by Petersen et al. [2020] and Berner et al. [2019] who study whether two neural networks (ReLU or otherwise) that are close in the functional space have parameterisations that are close in the weight space. This is called inverse stability. In contrast, we are

interested in ReLU networks that are functionally identical, and ask about all their possible parameterisations.

In terms of proof technique, our approach is based on the geometry of piece-wise linear functions, specifically the boundaries between linear regions. The intuition for this kind of analysis has previously been presented by Raghu et al. [2017] and Serra et al. [2018], and somewhat similar proof techniques to ours have been used by Hanin and Rolnick [2019] in the context of counting the number of linear decision regions.

Finally, the sets of equivalent parametrisations can be viewed as symmetries in the weight space, with implications for optimisation. Multiple authors, including e.g. Neyshabur et al. [2015], Badrinarayanan et al. [2016], Stock et al. [2019], have observed that the naive loss gradient is sensitive to reparametrisation by scaling, and proposed alternative, scaling-invariant optimisation procedures.

2.2 General and transparent networks

This section introduces two important classes of ReLU networks that we refer to throughout the chapter.

General ReLU network. In this chapter, we restrict our attention to so-called *general* ReLU networks. Intuitively, a general network is one that satisfies a number of non-degeneracy properties, such as all weight matrices having non-zero entries and full rank, no two network neurons exactly cancelling each other out, etc. It can be shown[1] that *almost all* ReLU networks are general. In other words, a sufficient condition for a ReLU network to be general with probability one is that its weights are sampled from a distribution with a density.

More formally, a general ReLU network is one that satisfies the following three conditions.

1. For any neuron (l, i), the local optima of $h_i^{1:l}$ do *not* have value exactly zero.
2. For all $k \leq l$ and all diagonal matrices $(\mathbf{I}_k, \ldots, \mathbf{I}_l)$ with entries in $\{0, 1\}$,
$$\operatorname{rank}\left(\mathbf{I}_l \mathbf{W}_l \mathbf{I}_{l-1} \cdots \mathbf{I}_k \mathbf{W}_k\right) = \min\left\{d_{k-1}, \operatorname{rank}(\mathbf{I}_k), \ldots, \operatorname{rank}(\mathbf{I}_{l-1}), \operatorname{rank}(\mathbf{I}_l)\right\}. \quad (2.6)$$
3. For any two neurons (l, i), (k, j), any linear region $\mathcal{R}_1 \subseteq \mathcal{X}$ of $h_i^{1:l}$, and any linear region $\mathcal{R}_2 \subseteq \mathcal{X}$ of $h_j^{1:k}$, the linear functions implemented by $h_i^{1:l}$ on $\mathcal{R}_1$ and $h_j^{1:k}$ on $\mathcal{R}_2$ are *not* multiples of each other.

General networks are convenient to study, as they exclude many degenerate special cases.

The second important class of ReLU networks are so-called *transparent* networks. Their significance as well as their name will become clear in the next section. For now, we state the definition.

Transparent ReLU network. A ReLU network $h : \mathcal{X} \to \mathbb{R}$ is called transparent if for all $\mathbf{x} \in \mathcal{X}$ and $l \in [L-1]$, there exists $i \in [d_l]$ such that $h_i^{1:l}(\mathbf{x}) \geq 0$. In words, we require that for any input, at least one neuron on each layer is active.

[1] See Appendix, Lemmas A.10, A.11 and A.12.

Figure 2.1: A piece-wise linear surface of order one, two and three.[3]

2.3 Fold-sets

We now introduce the concept of *fold-sets*, which is key to our understanding of ReLU networks and their prediction surfaces. Since ReLU networks are piece-wise linear functions, a great deal about them is revealed by the boundaries between individual linear regions. A network's fold-set is simply the union of all these boundaries.

More formally, if $\mathcal{X}$ is an open set, and $f : \mathcal{X} \to \mathbb{R}$ is any continuous, piece-wise linear function, we define the *fold-set of* f, denoted by $\mathcal{F}(f)$, as the set of all points at which f is non-differentiable.

It turns out there is a class of networks whose fold-sets are especially easy to understand; these are the ones we have termed *transparent*. For transparent networks, we have the following characterisation of the fold-set (which also motivates the name 'transparent').

Lemma 2.1. *If $h : \mathcal{X} \to \mathbb{R}$ is a general and transparent ReLU network, then*

$$\mathcal{F}(h) = \bigcup_{l,i} \left\{ \mathbf{x} \,|\, h_i^{1:l}(\mathbf{x}) = 0 \right\}. \tag{2.7}$$

To appreciate the significance of the lemma, suppose we are given some transparent ReLU network function h and we want to infer its parameters. This lemma shows that the knowledge of the *end-to-end mapping* $h \triangleq h^{1:L}$ in fact gives us information about the network's *hidden units* $h_i^{1:l}$ (hence 'transparent'). Moreover, this information is very explicit: we observe the units' zero-level sets, which in the case of a linear unit on a full-dimensional space already determines the unit's parameters up to rescaling[2]. Of course, dealing with piece-wise linearity and disambiguating the union into its constituent zero-level sets remains a challenge, which we deal with in upcoming sections.

2.4 Piece-wise linear surfaces

In this section, we provide a geometric description of fold-sets of transparent networks. Intuitively, the fold-sets look like the sets shown in Figure 2.1. The first-layer neurons of a network are linear functions, so the component $\bigcup_i \{\mathbf{x} \,|\, h_i^{1:1}(\mathbf{x}) = 0\}$ of the fold-set (2.7) is a union of hyperplanes, illustrated by the blue lines in Figure 2.1. These hyperplanes partition the input space into a number of regions that each correspond to a different *activation*

[2] See Appendix, Lemma A.19.

[3] A similar figure has appeared in the work of Raghu et al. [2017].

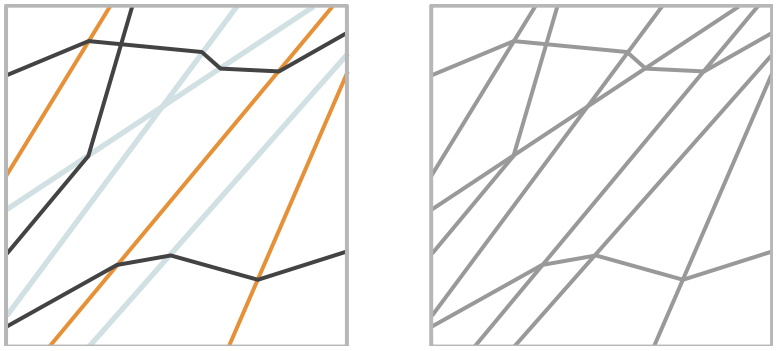


Figure 2.2: A piece-wise linear surface with few intersections between piece-wise hyperplanes. From the fold-set alone (right) it is not possible to determine if a hyperplane emerged from the first layer (left, blue) or from the second one (left, orange).

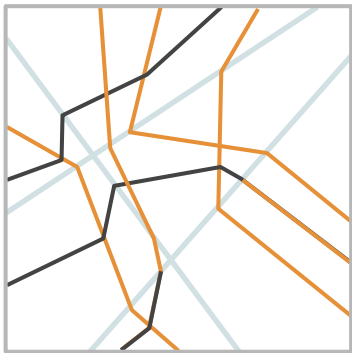


Figure 2.3: Greedy layer assignment to the piece-wise linear surface in Figure 2.1.

pattern. For a fixed activation pattern, or equivalently on each region, the second-layer neurons are linear, so their zero-level sets $\bigcup_i \{\mathbf{x} \,|\, h_i^{1:2}(\mathbf{x}) = 0\}$ are composed of piece-wise hyperplanes on the partition induced by the first-layer neurons. This is shown by the orange lines in Figure 2.1. More generally, the l^{th}-layer zero-level sets $\bigcup_i \left\{\mathbf{x} \,|\, h_i^{1:l}(\mathbf{x}) = 0\right\}$ consist of piece-wise hyperplanes on the partition induced by all lower-layer neurons. This yields a fold-set that looks like the set in the right pane of Figure 2.1, but potentially much more complicated.

We now define these concepts more precisely.

Piece-wise hyperplane. Let $\mathcal{P}$ be a partition of $\mathcal{X}$. We say $\mathcal{H} \subseteq \mathcal{X}$ is a *piece-wise hyperplane* with respect to partition $\mathcal{P}$, if $\mathcal{H}$ is nonempty and there exist $(\mathbf{w}, b) \neq (\mathbf{0}, 0)$ and $P \in \mathcal{P}$ such that $\mathcal{H} = \{\mathbf{x} \in P \,|\, \mathbf{w}^\intercal \mathbf{x} + b = 0\}$.

Piece-wise linear surface. A set $\mathcal{S} \subseteq \mathcal{X}$ is called a *piece-wise linear surface on $\mathcal{X}$ of order κ* if it has a representation of the form $\mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l$, where each $\mathcal{H}_i^l$ is a piece-wise hyperplane with respect to the partition induced by $\bigcup_{k \in [l-1], j \in [n_k]} \mathcal{H}_j^k$, and no number smaller than κ admits such a representation.

Using these definitions, the following lemma formalises the intuition behind Figure 2.1.

Lemma 2.2. *If h is a general and transparent ReLU network, then its fold-set is a piece-wise linear surface of order at most $L - 1$.*

The final ingredient we will need to be able to reason about the parameterisation of ReLU networks is a more precise characterisation of the fold-set, in particular, the *dependence structure* between individual piece-wise hyperplanes. For example, consider the piece-wise linear surface in Figure 2.1 and compare it to the one in Figure 2.2. Suppose as before that the blue hyperplanes come from first-layer neurons, the orange hyperplanes come from second-layer neurons, and the black hyperplanes come from third-layer neurons. The difference between Figure 2.1 and Figure 2.2 is that if we observe only the fold-set, i.e. only the union of the zero-level sets over all layers (as shown in the right pane of Figure 2.2), then in the case of Figure 2.2, it is impossible to know which folds come from which layers. For instance, the blue folds and the orange folds could be assigned to the first and second

layer almost arbitrarily; there is not enough information (i.e. intersection) in the fold-set to tell which is which. In contrast, the piece-wise linear surface in the rightmost pane of Figure 2.1 could in principle be disambiguated into first-, second- and third- layer folds. Here is a possible disambiguation procedure:

1. Take the largest possible union of hyperplanes that is a subset of the fold-set, and assign the hyperplanes to layer one.
2. Take all piece-wise hyperplanes with respect to the partition induced by the first-layer folds, and assign them to layer two.
3. Take all piece-wise hyperplanes with respect to the partition induced by the first- and second- layer folds, and assign them to layer three.

This procedure is not guaranteed to assign all folds to their original layers because it ignores how piece-wise hyperplanes are connected; for example for the piece-wise linear surface in Figure 2.1, the procedure yields the layer assignment shown in Figure 2.3. However, it is sufficient for our purposes, and it is easier to work with mathematically.

Formally, for a piece-wise linear surface $\mathcal{S}$, we denote

$$\square_k \mathcal{S} := \bigcup \{\mathcal{S}' \subseteq \mathcal{S} \mid \mathcal{S}' \text{ is a piece-wise linear surface of order at most } k\}. \tag{2.8}$$

One can show[4] that $\square_k \mathcal{S}$ is itself a piece-wise linear surface of order at most k, so one can think of $\square_k \mathcal{S}$ as the 'largest possible' subset of $\mathcal{S}$ that is a piece-wise linear surface of order at most k. For the piece-wise linear surface in Figure 2.3, the set $\square_1 \mathcal{S}$ consists of the blue hyperplanes, $\square_2 \mathcal{S}$ consists of the blue and the orange (piece-wise) hyperplanes, and $\square_3 \mathcal{S} = \mathcal{S}$.

This definition allows us to uniquely decompose $\mathcal{S}$ into its piece-wise hyperplanes. Let $\mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l$ be any representation of $\mathcal{S}$ in terms of its piece-wise hyperplanes. We say the representation is *canonical* if each $\mathcal{H}_i^l$ is distinct and $\bigcup_{l \in [k], i \in [n_l]} \mathcal{H}_i^l = \square_k \mathcal{S}$ for all $k \in [\kappa]$. One can show[5] that such a representation exists and is unique up to subscript indexing. Importantly, it assigns a unique 'layer' to each piece-wise hyperplane, its superscript.

The *dependency graph* (see also Figure 2.4) is a way to formally describe the dependencies between piece-wise hyperplanes.

Dependency graph. Let $\mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l$ be the canonical representation of $\mathcal{S}$. The *dependency graph of* $\mathcal{S}$ is the directed graph that has the piece-wise hyperplanes $\left\{\mathcal{H}_i^l\right\}_{l,i}$ as vertices, and has an edge $\mathcal{H}_i^l \to \mathcal{H}_j^k$ iff $l < k$ and $\operatorname{relint} \mathcal{H}_i^l \cap \operatorname{cl} \mathcal{H}_j^k \neq \emptyset$. That is, there is and edge $\mathcal{H}_i^l \to \mathcal{H}_j^k$ if $\mathcal{H}_j^k$ 'depends on' or 'bends at' $\mathcal{H}_i^l$.

2.5 Main result

With all the necessary concepts in place, we now put the pieces together and explain the proof idea behind the main result. We restate the theorem here for the reader's convenience.

[4] See Appendix, Lemma A.1.
[5] See Appendix, Lemmas A.5 and A.6.

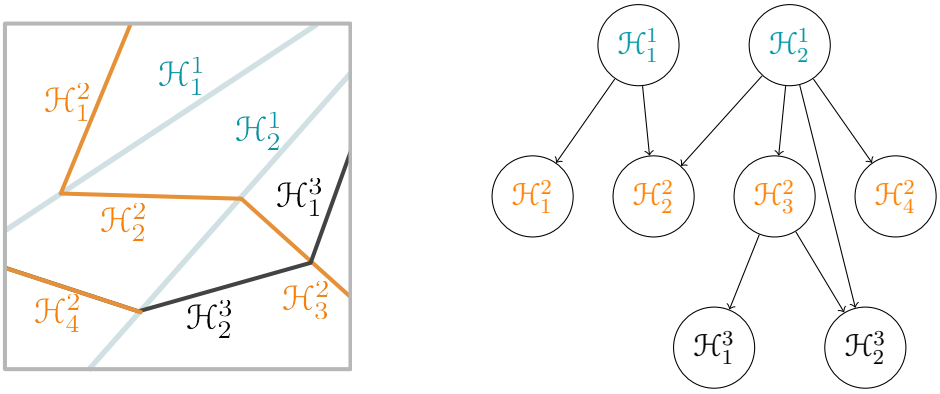


Figure 2.4: A piece-wise linear surface in canonical form and its dependency graph.

Theorem 2.1. *Consider a bounded open nonempty domain $\mathcal{X} \subseteq \mathbb{R}^{d_0}$ and any architecture $(d_0, \dots, d_L)$ with $d_0 \geq d_1 \geq \cdots \geq d_{L-1} \geq 2$, $d_L = 1$. For this architecture, there exists a ReLU network $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}$ such that for any general ReLU network $h_{\boldsymbol{\eta}} : \mathcal{X} \to \mathbb{R}$ (with the same architecture) satisfying $h_{\boldsymbol{\theta}}(\mathbf{x}) = h_{\boldsymbol{\eta}}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$, there exist permutation matrices $\mathbf{P}_1, \dots \mathbf{P}_{L-1}$, and positive diagonal matrices $\mathbf{M}_1, \dots, \mathbf{M}_{L-1}$, such that*

$$\begin{aligned} \mathbf{W}_1 &= \mathbf{M}_1 \mathbf{P}_1 \mathbf{W}_1', & \mathbf{b}_1 &= \mathbf{M}_1 \mathbf{P}_1 \mathbf{b}_1', \\ \mathbf{W}_l &= \mathbf{M}_l \mathbf{P}_l \mathbf{W}_l' \mathbf{P}_{l-1}^{-1} \mathbf{M}_{l-1}^{-1}, & \mathbf{b}_l &= \mathbf{M}_l \mathbf{P}_l \mathbf{b}_l', \qquad l \in \{2, \dots, L-1\}, \\ \mathbf{W}_L &= \mathbf{W}_L' \mathbf{P}_{L-1}^{-1} \mathbf{M}_{L-1}^{-1}, & \mathbf{b}_L &= \mathbf{b}_L', \end{aligned} \tag{2.9}$$

where $(\mathbf{W}_1, \mathbf{b}_1, \dots, \mathbf{W}_L, \mathbf{b}_L)$ are the parameters of $h_{\boldsymbol{\theta}}$, and $(\mathbf{W}_1', \mathbf{b}_1', \dots, \mathbf{W}_L', \mathbf{b}_L')$ are the parameters of $h_{\boldsymbol{\eta}}$.

In other words, for architectures with non-increasing widths, there exists a ReLU network h such that knowledge of the input-output mapping h determines the network's parameters uniquely up to permutation and scaling.

The idea behind the proof is as follows. Suppose we are given the function h. Then we also know its fold-set $\mathcal{F}(h)$, and if h is general and transparent, the fold-set is a piece-wise linear surface (by Lemma 2.2) of the form $\mathcal{F}(h) = \bigcup_{l,i} \left\{ \mathbf{x} \,|\, h_i^{1:l}(\mathbf{x}) = 0 \right\}$. As we have mentioned earlier, this union of zero-level sets contains a lot of information about the network's parameters, provided one can disambiguate the union to obtain the zero-level sets of individual neurons.

This disambiguation of the union is crucial, but is impossible in general. To see why, consider the first-layer neurons: given $\mathcal{F}(h)$, we want to identify $\bigcup_i \{\mathbf{x} \,|\, h_i^{1:1}(\mathbf{x}) = 0\}$. We know that $\bigcup_i \{\mathbf{x} \,|\, h_i^{1:1}(\mathbf{x}) = 0\}$ is a union of d_1 hyperplanes that is a subset of $\square_1 \mathcal{F}(h)$, so if $\square_1 \mathcal{F}(h)$ is a union of d_1 hyperplanes, we are done. In general however, $\mathcal{F}(h)$ may contain more than d_1 hyperplanes, such as for example in Figure 2.2. In such a setting it is impossible to tell which hyperplanes come from the first layer.

The key insight here is the following: even though, say, a last-layer neuron can create a fold that looks like a hyperplane, this hyperplane cannot have any dependencies, or descendants in the dependency graph. This follows from the fact that the layer is the last. More generally, if a (piece-wise) hyperplane has a chain of descendants of length m, it must come from a layer that is at least m layers below the last one. Formally, we have the following lemma.

Lemma 2.3. *Let $h : \mathcal{X} \to \mathbb{R}$ be a general ReLU network. Denote $\mathcal{S} := \bigcup_{l\in[\lambda],i\in[d_l]} \left\{\mathbf{x} \,|\, h_i^{1:l}(\mathbf{x}) = 0\right\}$ and let $\mathcal{S} = \bigcup_{k\in[\kappa],j\in[n_k]} \mathcal{H}_j^k$ be the canonical representation of $\mathcal{S}$. Then for all $\mathcal{H}_j^k$ there exists a neuron (l, i) with $l \geq k$ such that $\mathcal{H}_j^k \subseteq \left\{\mathbf{x} \,|\, h_i^{1:l}(\mathbf{x}) = 0\right\}$. Moreover, if the dependency graph of $\mathcal{S}$ contains a directed path of length m starting at $\mathcal{H}_j^k$, then $l \leq \lambda - m$.*

Main proof idea. This lemma motivates the main idea of the proof. We explicitly construct a network h such that the dependency graph of its fold-set is well connected. More precisely, we ensure that each of the hyperplanes corresponding to first-layer neurons has a chain of descendants of length $L - 2$. This implies by Lemma 2.3 that the first-layer hyperplanes can be identified as such, using only the information contained in the fold-set. One can show that this is sufficient to recover the parameters $\mathbf{W}_1, \mathbf{b}_1$, up to permutation and scaling. To extend the argument to higher-layers, we then consider the truncated network $h^{l:L}$. In $h^{l:L}$, layer l becomes the first layer, and we apply the same reasoning as above to recover $\mathbf{W}_l, \mathbf{b}_l$.

The next lemma shows that a network with a 'well connected' dependency graph exists. In what follows, $f|_{\mathcal{A}}$ denotes the restriction of a function f to a domain $\mathcal{A}$, and $\mathcal{X}_{\boldsymbol{\theta}}^l \triangleq \left\{\rho(h_{\boldsymbol{\theta}}^{1:l}(\mathbf{x})) \,|\, \mathbf{x} \in \mathcal{X}\right\}$ is the set of all possible inputs to the truncated network $h^{l:L}$. For notational convenience, we define $\mathcal{X}_{\boldsymbol{\theta}}^0 \triangleq \mathcal{X}$.

Lemma 2.4. *For a bounded open nonempty domain $\mathcal{X}$ and architecture $(d_0, \ldots, d_L)$ with $d_0 \geq d_1 \geq \cdots \geq d_{L-1} \geq 2$, $d_L = 1$, there exists a general transparent ReLU network $h : \mathcal{X} \to \mathbb{R}$ such that for $l \in [L - 1]$, the fold-set $\mathcal{F}(h^{l:L}|_{\operatorname{int}\mathcal{X}^{l-1}})$ is a piece-wise linear surface whose dependency graph contains d_l directed paths of length $(L - 1 - l)$ with distinct starting vertices.*

Theorem 2.1 then follows by the inductive argument outlined above.

Proof sketch of Theorem 2.1. Let $h_{\boldsymbol{\theta}}$ be the network from Lemma 2.4. One can show that if $h_{\boldsymbol{\theta}}$ is transparent, and $h_{\boldsymbol{\eta}}(\mathbf{x}) = h_{\boldsymbol{\theta}}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$, then also $h_{\boldsymbol{\eta}}$ is transparent, and all the truncated networks $h_{\boldsymbol{\theta}}^{l:L}$, $h_{\boldsymbol{\eta}}^{l:L}$ are transparent.

We proceed by induction. Let $l = 1$. Then we have

$$h_{\boldsymbol{\theta}}^{l:L}|_{\operatorname{int}\mathcal{X}_{\boldsymbol{\theta}}^{l-1}} \equiv h_{\boldsymbol{\theta}} \equiv h_{\boldsymbol{\eta}} \equiv h_{\boldsymbol{\eta}}^{l:L}|_{\operatorname{int}\mathcal{X}_{\boldsymbol{\theta}}^{l-1}} \tag{2.10}$$

which implies $\mathcal{F}(h_{\boldsymbol{\theta}}^{l:L}|_{\operatorname{int}\mathcal{X}_{\boldsymbol{\theta}}^{l-1}}) = \mathcal{F}(h_{\boldsymbol{\eta}}^{l:L}|_{\operatorname{int}\mathcal{X}_{\boldsymbol{\theta}}^{l-1}})$. (For notational convenience, we will omit the domain restriction for now.) Because both networks are general and transparent, the fold-sets are representable as unions of the respective zero-level sets, and we obtain

$$\bigcup_{k\in[L-l],j\in[d_k]} \left\{\mathbf{x} \,|\, h_{\boldsymbol{\theta}}^{l:l-1+k}[j](\mathbf{x}) = 0\right\} = \bigcup_{k\in[L-l],j\in[d_k]} \left\{\mathbf{x} \,|\, h_{\boldsymbol{\eta}}^{l:l-1+k}[j](\mathbf{x}) = 0\right\} \tag{2.11}$$

This is a piece-wise linear surface, whose dependency graph by Lemma 2.4 contains d_l directed paths of length $(L - 1 - l)$ with distinct starting vertices. Denote these vertices $\mathcal{H}_1, \ldots, \mathcal{H}_{d_l}$. By Lemma 2.3, $\mathcal{H}_i \subseteq \left\{\mathbf{x} \,|\, h_{\boldsymbol{\theta}}^{l:l-1+\lambda}[\iota](\mathbf{x}) = 0\right\}$ for some (λ, ι) with $\lambda \leq (L - l) - (L - 1 - l) = 1$. We thus obtain $\bigcup_{i\in[d_l]} \mathcal{H}_i \subseteq \bigcup_{i\in[d_l]} \left\{\mathbf{x} \,|\, h_{\boldsymbol{\theta}}^{l}[\iota](\mathbf{x}) = 0\right\}$, where on the left-hand side we have a union of d_l hyperplanes, and on the right-hand side we have a

union of at most d_l hyperplanes. It follows that the two sides are equal, and by applying the same argument to $h_{\boldsymbol{\eta}}$, we get

$$\bigcup_{i\in[d_l]} \Big\{\mathbf{x} \,|\, h^l_{\boldsymbol{\theta}}[i](\mathbf{x}) = 0\Big\} = \bigcup_{i\in[d_l]} \Big\{\mathbf{x} \,|\, h^l_{\boldsymbol{\eta}}[i](\mathbf{x}) = 0\Big\}. \tag{2.12}$$

Therefore there must exist a permutation $\pi : [d_l] \to [d_l]$ such that

$$\Big\{\mathbf{x} \,|\, h^l_{\boldsymbol{\theta}}[i](\mathbf{x}) = 0\Big\} = \Big\{\mathbf{x} \,|\, h^l_{\boldsymbol{\eta}}[\pi(i)](\mathbf{x}) = 0\Big\} \tag{2.13}$$

for all i. One can show[6] that this implies the existence of scalars $m_1, \dots m_{d_l}$, such that

$$(\mathbf{W}_l[i,:], b_l[i]) = m_i(\mathbf{W}'_l[\pi(i),:], b'_l[\pi(i)]). \tag{2.14}$$

We know that $m_i \neq 0$ because the folds $\Big\{\mathbf{x} \,|\, h^l_{\boldsymbol{\theta}}[i](\mathbf{x}) = 0\Big\}, \Big\{\mathbf{x} \,|\, h^l_{\boldsymbol{\eta}}[i](\mathbf{x}) = 0\Big\}$, are nonempty; otherwise $\bigcup_{i\in[d_l]} \mathcal{H}_i$ could not be a union of d_l hyperplanes. We have thus shown that there exists a permutation matrix $\mathbf{P}_l \in \mathbb{R}^{d_l \times d_l}$ and a nonzero-entry diagonal matrix $\mathbf{M}_l \in \mathbb{R}^{d_l \times d_l}$ such that $\mathbf{W}_l = \mathbf{M}_l\mathbf{P}_l\mathbf{W}'_l$ and $\mathbf{b}_l = \mathbf{M}_l\mathbf{P}_l\mathbf{b}'_l$. One can also show that the scalars m_i are positive.[7]

For the inductive step, let $l \in \{2, \dots, L-1\}$, and assume that there exist permutation matrices $\mathbf{P}_1, \dots, \mathbf{P}_{l-1}$, and positive-entry diagonal matrices $\mathbf{M}_1, \dots, \mathbf{M}_{l-1}$, such that (2.9) holds up to layer $l-1$. Then $h^{1:l-1}_{\boldsymbol{\theta}} \equiv \mathbf{M}_{l-1}\mathbf{P}_{l-1}h^{1:l-1}_{\boldsymbol{\eta}}$. Since the end-to-end mappings are the same, $h^{1:L}_{\boldsymbol{\theta}} \equiv h^{1:L}_{\boldsymbol{\eta}}$, it follows that the truncated mappings satisfy

$$h^{l:L}_{\boldsymbol{\theta}}|_{\operatorname{int} \mathcal{X}^{l-1}_{\boldsymbol{\theta}}} \equiv \Big(h^{l:L}_{\boldsymbol{\eta}} \circ \mathbf{P}^{-1}_{l-1}\mathbf{M}^{-1}_{l-1}\Big)\Big|_{\operatorname{int} \mathcal{X}^{l-1}_{\boldsymbol{\theta}}} \equiv h^{l:L}_{\tilde{\boldsymbol{\eta}}}|_{\operatorname{int} \mathcal{X}^{l-1}_{\boldsymbol{\theta}}}, \tag{2.15}$$

where $\tilde{\boldsymbol{\eta}} := (\mathbf{W}'_l\mathbf{P}^{-1}_{l-1}\mathbf{M}^{-1}_{l-1}, \mathbf{b}'_l, \mathbf{W}'_{l+1}, \mathbf{b}'_{l+1}, \dots, \mathbf{W}'_L, \mathbf{b}'_L)$. We therefore apply the same argument to $h^{l:L}_{\boldsymbol{\theta}}|_{\operatorname{int} \mathcal{X}^{l-1}_{\boldsymbol{\theta}}}$ and $h^{l:L}_{\tilde{\boldsymbol{\eta}}}|_{\operatorname{int} \mathcal{X}^{l-1}_{\boldsymbol{\theta}}}$ as we presented above for the case $l = 1$. We obtain that there exists a permutation matrix $\mathbf{P}_l \in \mathbb{R}^{d_l \times d_l}$ and a positive-entry diagonal matrix $\mathbf{M}_l \in \mathbb{R}^{d_l \times d_l}$ such that

$$\mathbf{W}_l = \mathbf{M}_l\mathbf{P}_l\mathbf{W}'_l\mathbf{P}^{-1}_{l-1}\mathbf{M}^{-1}_{l-1}, \qquad \mathbf{b}_l = \mathbf{M}_l\mathbf{P}_l\mathbf{b}'_l. \tag{2.16}$$

Finally, consider the last layer. We know that $h^{1:L-1}_{\boldsymbol{\theta}} \equiv \mathbf{M}_{L-1}\mathbf{P}_{L-1}h^{1:L-1}_{\boldsymbol{\eta}}$, which implies $h^L_{\boldsymbol{\theta}} \equiv h^L_{\boldsymbol{\eta}} \circ \mathbf{P}^{-1}_{L-1}\mathbf{M}^{-1}_{L-1}$, i.e. $h^L_{\boldsymbol{\theta}}$ and $h^L_{\boldsymbol{\eta}} \circ \mathbf{P}^{-1}_{L-1}\mathbf{M}^{-1}_{L-1}$ are identical linear functions supported on the full-dimensional domain $\mathcal{X}^{L-1}_{\boldsymbol{\theta}}$. It follows that $\mathbf{W}_L = \mathbf{W}'_L\mathbf{P}^{-1}_{L-1}\mathbf{M}^{-1}_{L-1}$ and $\mathbf{b}_L = \mathbf{b}'_L$. □

Discussion of assumptions. Most of the theorem's assumptions have their origin in Lemma 2.4. The reason we restrict the domain of $h^{l:L}$ to the interior of $\mathcal{X}^{l-1}$ is that we want $h^{l:L}$ to be defined on an open set (otherwise fold-sets become unwieldy). For similar reasons, we study only architectures with non-increasing widths; otherwise $\operatorname{int} \mathcal{X}^{l-1}$ may be empty. We conjecture that the theorem does *not* hold for more general architectures. If it does, the proof will likely go beyond fold-sets.

To guarantee transparency, our construction is such that for each input $\mathbf{x} \in \mathcal{X}$ and layer $l \in [L-1]$, either $h^{1:l}_1(\mathbf{x}) > 0$ or $h^{1:l}_2(\mathbf{x}) > 0$. Transparency could in principle be achieved

[6] See Appendix, Lemma A.19.

[7] See Appendix, Theorem A.1.

with just a single neuron, but it would have to be positive everywhere. This is why we impose $d_l \geq 2$. Guaranteeing transparency for the first layer (whose inputs are not constrained to the positive quadrant) also necessitates boundedness of $\mathcal{X}$. Boundedness can be lifted if we consider a slightly modified definition of transparency; proofs become more complicated though and we do not consider this crucial.

Almost all of the proof carries over to the case of leaky ReLU activations (where ρ is defined as $\rho(\mathbf{u})_i = \max\{\alpha u_i, u_i\}$ for some small $\alpha > 0$). The part that does *not* carry over is our proof that $\mathbf{M}_l$ has only positive entries on the diagonal: In this part, we compare the slope of $h_{\boldsymbol{\theta}}^{l:L}$ for inputs on the positive and negative side of a given ReLU neuron, and notice that the negative-side slope is 'singular' in the sense that some basis directions have zero magnitude. This particular argument does not work for the leaky ReLU, though we cannot rule out that a simple workaround exists.

2.6 Discussion & future work

In this chapter, we have shown that for architectures with non-increasing widths, certain ReLU networks are almost uniquely identified by the function they implement. The result suggests that the function-equivalence classes of ReLU networks are surprisingly small, i.e. there may be only little redundancy in the way ReLU networks are parameterised, contrary to what is commonly believed.

This apparent contradiction could be explained in a number of ways:

- It could be the case that even though exact equivalence classes are small, approximate equivalence is much easier to achieve. That is, it could be that $\|h_{\boldsymbol{\theta}} - h_{\boldsymbol{\eta}}\| \leq \epsilon$ is satisfied by a disproportionately larger class of parameters $\boldsymbol{\eta}$ than $\|h_{\boldsymbol{\theta}} - h_{\boldsymbol{\eta}}\| = 0$. This issue is related to the so-called inverse stability of the realisation map of neural nets, which is not yet well understood.
- Another possibility is that the kind of networks we consider in this paper is not representative of networks typically encountered in practice, i.e. it could be that 'typical networks' do not have well connected dependency graphs, and are therefore not easily identifiable.
- Finally, we have considered only architectures with non-increasing widths, whereas some previous theoretical work has assumed much wider intermediate layers compared to the input dimension. It is possible that parameterisation redundancy is much larger in such a regime compared to ours. However, gains from over-parameterisation have also been observed in practical settings with architectures not unlike those considered here.

We consider these questions important directions for further research. We also hypothesise that our analysis could be extended to convolutional and recurrent networks, and to other piece-wise linear activation functions such as leaky ReLU.

CHAPTER 3

The inductive bias of ReLU networks on orthogonally separable data

We now turn to the problem of understanding the inductive bias of ReLU networks: from the many zero-risk hypotheses, how does the learning algorithm decide which one to return?

This problem is actively investigated in the community, and the implicit bias has already been worked out for many linear models.[1] Notably, Soudry et al. [2018] consider a logistic regression classifier trained on linearly separable data, and show that the normalised weight vector converges to the max-margin direction. Building on their work, Ji and Telgarsky [2019a] consider deep linear networks, also trained on linearly separable data, and show that the normalised end-to-end weight vector converges to the max-margin direction. They in fact show that all first-layer neurons converge to the same 'canonical neuron' (which points in the max-margin direction). Although such impressive progress on linear models has spurred attempts at nonlinear extensions, the problem is much harder and analogous nonlinear results have been elusive.

In this chapter, we provide the first such inductive-bias result for ReLU networks trained on 'easy' datasets. Specifically, we

- propose orthogonal separability of datasets as a stronger form of linear separability that facilitates the study of ReLU network training,
- prove that a two-layer ReLU network trained on an orthogonally separable dataset learns a function with two distinct groups of neurons, where all neurons in each group converge to the same 'canonical neuron',
- characterise the directions of the canonical neurons, which turn out to be the max-margin directions for the positive and the negative data subset.

The proof is based on the recently introduced concept of extremal sectors [Maennel et al., 2018] which govern the early phase of training. We prove a precise characterisation of extremal sectors for orthogonally separable datasets, and an invariance property which ensures that the network's activation pattern becomes fixed at some point during training. The latter allows us to treat ReLU networks late in training as ensembles of linear networks, which are much more tractable.

[1] A more thorough overview of related work can be found in Section 3.5.

3.1 Setting and assumptions

In this section, we introduce the learning scenario including the assumptions we make about the dataset, the model, and the training procedure. We consider binary classification. Denote the training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ for all $i \in [n]$. We denote by $\mathbf{X} \in \mathbb{R}^{d \times n}$ the matrix with $\{\mathbf{x}_i\}$ as columns and by $\mathbf{y} \in \mathbb{R}^n$ the vector with $\{y_i\}$ as entries.

Orthogonally separable data. A binary classification dataset $(\mathbf{X}, \mathbf{y})$ is called *orthogonally separable* if for all $i, j \in [n]$,

$$\begin{aligned} \mathbf{x}_i^\intercal \mathbf{x}_j &> 0, && \text{if } y_i = y_j, \\ \mathbf{x}_i^\intercal \mathbf{x}_j &\leq 0, && \text{if } y_i \neq y_j. \end{aligned} \tag{3.1}$$

In other words, a dataset is orthogonally separable iff it is linearly separable, and any training example $\mathbf{x}_i$ can serve as a linear separator. Geometrically, this means that examples with $y_i = 1$ ('positive examples') and examples with $y_i = -1$ ('negative examples') lie in opposite orthants.

Two-layer ReLU network. We consider a two-layer width-p fully-connected ReLU network without biases, parameterised by $\boldsymbol{\theta} \triangleq (\mathbf{W}_1, \mathbf{W}_2) \triangleq (\mathbf{W}, \mathbf{a}^\intercal)$,

$$\begin{aligned} h_{\boldsymbol{\theta}} &: \mathbb{R}^d \to \mathbb{R}, \\ h_{\boldsymbol{\theta}}(\mathbf{x}) &\triangleq \mathbf{a}^\intercal \rho(\mathbf{W}\mathbf{x}), \end{aligned} \tag{3.2}$$

where $\mathbf{W} \triangleq [\mathbf{w}_1, \dots \mathbf{w}_p]^\intercal \in \mathbb{R}^{p \times d}$ and $\mathbf{a} \triangleq [a_1, \dots, a_p]^\intercal \in \mathbb{R}^p$ are the first- and second-layer weights of the network, and ρ is the element-wise ReLU function, $\rho(\mathbf{u})_i = \max\{0, u_i\}$. We will often view the network as a collection of neurons, $\{(a_j, \mathbf{w}_j)\}_{j=1}^p$.

Cross-entropy loss. We assume a training loss of the form

$$\ell(\boldsymbol{\theta}) \triangleq \sum_{i=1}^n \ell_i(h_{\boldsymbol{\theta}}(\mathbf{x}_i)), \qquad \ell_i(u) \triangleq \log(1 + \exp(-y_i u)); \tag{3.3}$$

this is the standard empirical cross-entropy loss. More generally, our results hold when the loss is differentiable, ℓ_i' is bounded and Lipschitz continuous, and satisfies $-y_i \ell_i'(u) > 0$ for all $u \in \mathbb{R}$.

Gradient flow training. We assume the loss is optimised by gradient descent with infinitesimally small step size, also known as gradient flow. Under the gradient flow dynamics, the parameter trajectory is an absolutely continuous curve $\{\boldsymbol{\theta}(t) \,|\, t \geq 0\}$ satisfying the differential inclusion

$$\frac{\partial \boldsymbol{\theta}(t)}{\partial t} \in -\partial \ell(\boldsymbol{\theta}(t)), \qquad \text{for almost all } t \in [0, \infty), \tag{3.4}$$

where $\partial \ell$ denotes the Clarke subdifferential [Clarke, 1975, Clarke et al., 2008] of ℓ, an extension of the gradient to not-everywhere differentiable functions,

$$\partial \ell(\boldsymbol{\theta}) \triangleq \operatorname{conv}\left\{ \lim_{k \to \infty} \nabla \ell(\boldsymbol{\theta}_k) \,\middle|\, \boldsymbol{\theta}_k \to \boldsymbol{\theta} \right\}. \tag{3.5}$$

$\boldsymbol{\theta}(t)$ is the value of the parameters at time t, and we will use the suffix (t) more generally to denote the value of some function of $\boldsymbol{\theta}$ at time t.

Near-zero balanced initialisation. We assume that the neurons $\{\mathbf{w}_j\}$ are initialised iid from the Gaussian distribution and then rescaled such that $\|\mathbf{w}_j\| \leq \lambda$, where $\lambda > 0$ is a small constant. That is, $\mathbf{w}_j = \lambda_j \mathbf{v}_j / \|\mathbf{v}_j\|$ for $\mathbf{v}_j \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$ and arbitrary λ_j satisfying $\lambda_j \in (0, \lambda]$. We also assume that $a_j \in \{\pm\lambda_j\}$. These technical conditions ensure that the neurons are balanced and small in size, $\|\mathbf{w}_j\| = |a_j| \approx 0$, which simplifies the calculations involved in the analysis of gradient flow.

Support examples span the full space. We assume that the support examples of the positive data subset $\{\mathbf{x}_i \,|\, y_i = 1\}$ span the entire $\mathbb{R}^d$, and similarly that the support examples of the negative data subset $\{\mathbf{x}_i \,|\, y_i = -1\}$ span $\mathbb{R}^d$. (We formally define support examples after introducing some more notation below.)

3.2 Main result

Under the assumptions of Section 3.1, the network converges to a linear combination of two max-margin neurons. Specifically, given a dataset $(\mathbf{X}, \mathbf{y})$, define the *positive and the negative max-margin vectors* $\mathbf{w}_+, \mathbf{w}_- \in \mathbb{R}^d$ as

$$\mathbf{w}_+ = \arg\min_{\mathbf{w}} \|\mathbf{w}\|^2 \quad \text{subject to} \quad \mathbf{w}^\intercal \mathbf{x}_i \geq 1 \;\text{ for }\; i : y_i = 1, \tag{3.6}$$

$$\mathbf{w}_- = \arg\min_{\mathbf{w}} \|\mathbf{w}\|^2 \quad \text{subject to} \quad \mathbf{w}^\intercal \mathbf{x}_i \geq 1 \;\text{ for }\; i : y_i = -1. \tag{3.7}$$

We call examples which attain equality in eqs. (3.6) and (3.7) *positive support examples* and *negative support examples* respectively. We now state the main result.

Theorem 3.1. *Let $h_{\boldsymbol{\theta}}$ be a two-layer width-p ReLU network trained by gradient flow with the cross-entropy loss, initialised near-zero and balanced. Consider an orthogonally separable dataset $(\mathbf{X}, \mathbf{y})$ such that its positive support examples span $\mathbb{R}^d$, and its negative support examples also span $\mathbb{R}^d$. For almost all such datasets[2] and with probability $1 - 1/2^p$ over the random initialisation,*

$$\left\| \frac{\mathbf{W}(t)}{\|\mathbf{W}(t)\|_F} - (\mathbf{u}\mathbf{w}_+^\intercal + \mathbf{z}\mathbf{w}_-^\intercal) \right\|_F \to 0, \qquad \text{as } t \to \infty, \tag{3.8}$$

for some $\mathbf{u}, \mathbf{z} \in \mathbb{R}^p_+$ such that either $u_i = 0$ or $z_i = 0$ for all $i \in [p]$. Also,

$$\left\| \frac{\mathbf{a}(t)}{\|\mathbf{a}(t)\|} - (\mathbf{u}\|\mathbf{w}_+\| - \mathbf{z}\|\mathbf{w}_-\|) \right\| \to 0, \qquad \text{as } t \to \infty. \tag{3.9}$$

The theorem says that each neuron (row of $\mathbf{W}$), properly normalised, converges either to a scalar multiple of the positive max-margin direction, $u_i \mathbf{w}_+$, or to a scalar multiple of the negative max-margin direction, $z_i \mathbf{w}_-$. In other words, there are asymptotically only two distinct types of neurons, and the network could in principle be pruned down to a width of just two. These two 'canonical neurons' moreover have an explicit characterisation, given by eqs. (3.6) and (3.7).

As for the second-layer weights, the magnitude of each a_j equals the norm of the respective $\mathbf{w}_j$, and the sign of a_j is $+1$ if $\mathbf{w}_j$ approaches $\mathbf{w}_+$ and -1 if $\mathbf{w}_j$ approaches $\mathbf{w}_-$.

[2] Formally, this means that if $\{\mathbf{x}_i\}$ are sampled from any distribution with a density wrt. the Lebesgue measure, then the theorem (treated as an implication) holds with probability one wrt. the data.

The following corollary summarises the above in terms of the function learnt by the network.

Corollary 3.1. *Under the conditions of Theorem 3.1, there exist constants* $u, z \geq 0$ *such that*

$$\frac{h_{\boldsymbol{\theta}(t)}(\mathbf{x})}{\|\boldsymbol{\theta}(t)\|^2} \to u\rho(\mathbf{w}_+^\intercal \mathbf{x}) - z\rho(\mathbf{w}_-^\intercal \mathbf{x}), \qquad \text{as } t \to \infty. \tag{3.10}$$

3.2.1 Discussion of assumptions

Many of our assumptions are technical, serving to simplify the analysis while detracting little from the result's relevance[3]. These include infinitesimal step size (gradient flow), balancedness at initialisation and the condition on support span. The first two could potentially be relaxed to their approximate counterparts, i.e. gradient descent with a small constant step size and approximate balancedness [Arora et al., 2019]. The assumption that support vectors span $\mathbb{R}^d$ comes from Ji and Telgarsky [2019a], Soudry et al. [2018]. It seems to us that it could be lifted, though we have not investigated this possibility in depth.

Two assumptions that deserve more attention are near-zero initialisation and orthogonal separability; both are crucial for the result to hold. Near-zero initialisation grants neurons high directional mobility early in training, allowing them to cluster close to the canonical directions. Orthogonal separability ensures that the canonical directions are 'easy to find' by local descent. In prior work, which considered linear networks, this role is fulfilled by linear separability. The reason we need a stronger condition is that ReLU updates are more local compared to linear updates: a linear neuron takes into account all examples in the training set, whereas a ReLU neuron updates only on examples in its positive half-plane (its active examples). ReLU neurons therefore easily get stuck in a variety of directions, unless the data is highly structured.

3.3 Proof sketch

In the analysis, we distinguish between two phases of training. The first phase takes place close to the origin, $\|\boldsymbol{\theta}\| \approx 0$. In this phase, while neurons move little in the absolute sense, they converge in direction to certain regions of the weight space called *extremal sectors*.

3.3.1 Convergence to extremal sectors

(All definitions and results in this subsection come from the prior work of Maennel et al. [2018]. We will need them later on.)

Sectors are regions in weight space corresponding to different activation patterns. They are important for understanding neuron dynamics: roughly speaking, neurons in the same sector move in the same direction.

Definition 3.1 (Sector)**.** *The* sector *associated to a sequence of signs* $\boldsymbol{\sigma} \in \{-1, 0, 1\}^n$ *is the region in input space defined as*

$$\mathcal{S}_{\boldsymbol{\sigma}} \triangleq \{\mathbf{w} \in \mathbb{R}^d \mid \operatorname{sign} \mathbf{w}^\intercal \mathbf{x}_i = \sigma_i,\ i \in [n]\}. \tag{3.11}$$

We may also refer to the sign sequence $\boldsymbol{\sigma}$ *itself as a sector.*

[3] We verify experimentally in Section 3.4.1 that these assumptions are indeed not crucial.

Some sectors are attractors early in training, i.e. neurons tend to converge to them. Such attracting sectors are called *extremal sectors*. To give a formal definition, we first introduce the function $G : \mathbb{S}^{d-1} \to \mathbb{R}$,

$$G(\mathbf{w}) \triangleq -\sum_{i=1}^{n} \ell_i'(0) \cdot \rho(\mathbf{w}^\intercal \mathbf{x}_i). \tag{3.12}$$

Intuitively, (normalised) neurons early in training behave as if they were locally optimising G, they therefore tend to cluster around the local optima of G. We formally define extremal sectors as sectors containing these local optima.

Definition 3.2 (Extremal directions and sectors)**.** *We say that $\mathbf{w} \in \mathbb{S}^{d-1}$ is a* positive extremal direction*, if it is a strict local maximum of G. We say that $\mathbf{w}$ is a* negative extremal direction *if it is a strict local minimum of G. A sector is called (positive/negative) extremal, if it contains a (positive/negative) extremal direction.*

The following lemma [Maennel et al., 2018, Lemma 5] shows that all neurons either turn off, i.e. become deactivated for all training examples and stop updating, or converge to extremal sectors.

Lemma 3.1. *Let a two-layer ReLU network $h_{\boldsymbol{\theta}}$ be balanced at initialisation and trained by gradient flow. Assume that the loss derivative ℓ_i' is Lipschitz continuous. Then, for almost all datasets and almost all initialisations with λ small enough, there exists a time T such that each neuron satisfies one of these three conditions:*

- $\mathbf{w}_j(T) \in \mathcal{S}_{\boldsymbol{\sigma}}$ *where* $\boldsymbol{\sigma} \leq \mathbf{0}$ *and so* $\mathbf{w}_j$ *remains constant for* $t \geq T$*, or*
- $a_j(T) > 0$ *and* $\mathbf{w}_j(T) \in \mathcal{S}_{\boldsymbol{\sigma}}$ *where* $\boldsymbol{\sigma}$ *is a positive extremal sector, or*
- $a_j(T) < 0$ *and* $\mathbf{w}_j(T) \in \mathcal{S}_{\boldsymbol{\sigma}}$ *where* $\boldsymbol{\sigma}$ *is a negative extremal sector.*

3.3.2 Orthogonal separability: Two absorbing extremal sectors

Lemma 3.1 shows that by the end of the early phase of training, neurons have converged to extremal sectors. Although eq. (3.12) shows that the number of extremal sectors depends only on the data (i.e. is independent of model expressivity), it is a priori unclear how many extremal sectors there are for a given dataset, or what happens once neurons have converged to extremal sectors. We now answer both of these questions for orthogonally separable datasets.

First, we claim that for orthogonally separable datasets, there are only two extremal sectors, one corresponding to the positive data subset and one corresponding to the negative data subset. That is, by converging to an extremal sector, neurons 'choose' whether to activate for positive examples or for negative examples. They thus naturally form two groups of similar neurons.

Lemma 3.2. *In the setting of Theorem 3.1, there is exactly one positive extremal direction and exactly one negative extremal direction. The positive extremal sector $\boldsymbol{\sigma}^+$ is given by*

$$\sigma_j^+ = \begin{cases} 1, & \text{if } y_j = 1, \\ -1, & \text{if } y_j = -1 \text{ and } \mathbf{x}_j^\intercal \mathbf{x}_i < 0 \text{ for some } i \text{ with } y_i = 1, \\ 0, & \text{if } y_j = -1 \text{ and } \mathbf{x}_j^\intercal \mathbf{x}_i = 0 \text{ for all } i \text{ with } y_i = 1, \end{cases} \tag{3.13}$$

and the negative extremal sector $\boldsymbol{\sigma}^-$ is given by

$$\sigma_j^- = \begin{cases} 1, & \text{if } y_j = -1, \\ -1, & \text{if } y_j = 1 \text{ and } \mathbf{x}_j^\intercal \mathbf{x}_i < 0 \text{ for some } i \text{ with } y_i = -1, \\ 0, & \text{if } y_j = 1 \text{ and } \mathbf{x}_j^\intercal \mathbf{x}_i = 0 \text{ for all } i \text{ with } y_i = -1. \end{cases} \tag{3.14}$$

Second, we show that once a neuron reaches an extremal sector, it remains in the sector forever, i.e. its activation pattern remains fixed for the rest of training.

Lemma 3.3. *Assume the setting of Theorem 3.1. If at time T the neuron $(a_j, \mathbf{w}_j)$ satisfies $a_j(T) > 0$ and $\mathbf{w}_j(T) \in \mathcal{S}_{\boldsymbol{\sigma}}$, where $\boldsymbol{\sigma}$ is the positive extremal sector (eq.* (3.13)*), then for $t \geq T$, $\mathbf{w}_j(t) \in \mathcal{S}_{\boldsymbol{\sigma}}$. The same holds if $a_j(T) < 0$ and $\boldsymbol{\sigma}$ is the negative extremal sector (eq.* (3.14)*).*

3.3.3 Proof of Theorem 3.1

Once neurons enter their respective absorbing sectors, the second phase of training begins. In this phase, the network's activation patterns are fixed: some neurons are active and update on the positive examples, while the others are active and update on the negative examples. The network thus behaves like an ensemble of independent linear subnetworks trained on subsets of the data. Once this happens, it becomes possible to apply existing results for linear networks; in particular, each subnetwork converges to its respective max-margin classifier.

We give more details in the proof below.

Proof of Theorem 3.1. By Lemmas 3.1 and 3.2, there exists a time T such that each neuron satisfies either

- $\mathbf{w}_j(T) \in \mathcal{S}_{\boldsymbol{\sigma}}$ where $\boldsymbol{\sigma} \leq \mathbf{0}$ and $\mathbf{w}_j$ remains constant for $t \geq T$, or
- $a_j(T) > 0$ and $\mathbf{w}_j(T) \in \mathcal{S}_{\boldsymbol{\sigma}^+}$, or
- $a_j(T) < 0$ and $\mathbf{w}_j(T) \in \mathcal{S}_{\boldsymbol{\sigma}^-}$,

where $\boldsymbol{\sigma}^+, \boldsymbol{\sigma}^-$ are the unique positive and negative extremal sectors given by eqs. (3.13) and (3.14). Denote by $\mathcal{J}_0, \mathcal{J}_+, \mathcal{J}_-$, the sets of neurons satisfying the first, the second, and the third condition respectively. By Lemma 3.3, if $j \in \mathcal{J}_+$ then $\mathbf{w}_j(t) \in \mathcal{S}_{\boldsymbol{\sigma}^+}$ for all $t \geq T$, and if $j \in \mathcal{J}_-$ then $\mathbf{w}_j(t) \in \mathcal{S}_{\boldsymbol{\sigma}^-}$ for $t \geq T$. Hence, for $t \geq T$, if $\mathbf{x}_i$ is such that $y_i = 1$ then

$$h_{\boldsymbol{\theta}}(\mathbf{x}_i) \triangleq \sum_{j \in [p]} a_j \rho(\mathbf{w}_j^\intercal \mathbf{x}_i) = \sum_{j \in \mathcal{J}_+} a_j \mathbf{w}_j^\intercal \mathbf{x}_i. \tag{3.15}$$

Combined with Lemma B.3, this implies that for $k \in \mathcal{J}_+$,

$$\begin{aligned} \frac{\partial a_k}{\partial t} &= -\sum_{i:y_i=1} \ell_i' \left(\sum_{j \in \mathcal{J}_+} a_j \mathbf{w}_j^\intercal \mathbf{x}_i \right) \cdot \mathbf{w}_k^\intercal \mathbf{x}_i, \\ \frac{\partial \mathbf{w}_k}{\partial t} &= -\sum_{i:y_i=1} \ell_i' \left(\sum_{j \in \mathcal{J}_+} a_j \mathbf{w}_j^\intercal \mathbf{x}_i \right) \cdot a_k \mathbf{x}_i, \end{aligned} \tag{3.16}$$

(where we have used that $\mathbf{P}_{\mathbf{w}}\mathbf{x}_i = \mathbf{x}_i$ for i with $y_i = 1$ due to positive extremality). From eq. (A.26) it follows that the evolution of neurons in $\mathcal{J}_+$ depends only on positive examples and other neurons in $\mathcal{J}_+$. The neurons behave linearly on the positive data subset, while ignoring the negative subset. The same argument shows that the evolution of neurons in $\mathcal{J}_-$ depends only on other neurons in $\mathcal{J}_-$ and the negative data subset, on which the neurons act linearly. In other words, from time T onwards the ReLU network decomposes into a constant part and two independent linear networks, one trained on the positive data subset and the other trained on the negative data subset.

We can therefore apply existing max-margin convergence results for linear networks to each of the linear subnetworks. Denote by $\mathbf{W}^\intercal = [\mathbf{W}_0^\intercal, \mathbf{W}_+^\intercal, \mathbf{W}_-^\intercal]$ the three parts of the weight matrix. Then by [Ji and Telgarsky, 2019a, Theorems 2.2 and 2.8] and [Ji and Telgarsky, 2020, Theorem 3.1], there exist vectors $\bar{\mathbf{u}}, \bar{\mathbf{z}}$, such that

$$\left\| \frac{\mathbf{W}_+(t)}{\|\mathbf{W}_+(t)\|_F} - \bar{\mathbf{u}}\mathbf{w}_+^\intercal \right\|_F \to 0, \qquad \text{as } t \to \infty, \tag{3.17}$$

$$\left\| \frac{\mathbf{W}_-(t)}{\|\mathbf{W}_-(t)\|_F} - \bar{\mathbf{z}}\mathbf{w}_-^\intercal \right\|_F \to 0, \qquad \text{as } t \to \infty. \tag{3.18}$$

(We allow $\bar{\mathbf{u}}, \bar{\mathbf{z}} \in \mathbb{R}^0$ to account for the fact that $\mathcal{J}_+, \mathcal{J}_-$ may be empty). We now need to relate $\|\mathbf{W}_+\|_F$ and $\|\mathbf{W}_-\|_F$ to $\|\mathbf{W}\|_F$. In particular, it will be useful to show that $\|\mathbf{W}_+(t)\|_F^2 / \log t$ has a limit as $t \to \infty$; the same is true for $\|\mathbf{W}_-(t)\|_F^2 / \log t$ (by the same argument). If $\mathcal{J}_+$ or $\mathcal{J}_-$ is empty, this is trivially true and the limit is 0. Otherwise, consider the learning of the positive linear subnetwork, whose objective is effectively $\ell^+(\boldsymbol{\theta}) := \sum_{i:y_i=1} \ell_i(h_{\boldsymbol{\theta}}(\mathbf{x}_i))$. By [Ji and Telgarsky, 2019a, Theorem 2.2], we know that $\ell^+(\boldsymbol{\theta}(t)) \to 0$ as $t \to \infty$. Following [Lyu and Li, 2020, Definition A.3], define

$$\tilde{\gamma}(\boldsymbol{\theta}) \triangleq \frac{g(\log 1/\ell^+(\boldsymbol{\theta}))}{2\|\mathbf{W}_+\|_F^2}, \tag{3.19}$$

where $g(q) := -\log\left(\exp(\exp(-q)) - 1\right)$ for the cross-entropy loss. Then

$$\frac{\|\mathbf{W}_+(t)\|_F^2}{\log t} = \frac{g(\log 1/\ell^+(\boldsymbol{\theta}(t)))}{2\tilde{\gamma}(t)\log t} = \frac{-\log\left(\exp(\ell^+(\boldsymbol{\theta}(t))) - 1\right)}{2\tilde{\gamma}(t)\log t}. \tag{3.20}$$

Using the Taylor expansion $\exp(u) = 1 + \Theta(u)$ for $u \to 0$ and [Lyu and Li, 2020, Corollary A.11], we obtain

$$\frac{\|\mathbf{W}_+(t)\|_F^2}{\log t} = \frac{-\log \Theta(\ell^+(\boldsymbol{\theta}(t)))}{2\tilde{\gamma}(t)\log t} = \frac{\log \Theta(t\log t)}{2\tilde{\gamma}(t)\log t} = \frac{1}{2\tilde{\gamma}(t)}\left(\frac{\Theta(1) + \log\log t}{\log t} + 1\right). \tag{3.21}$$

By [Lyu and Li, 2020, Theorem A.7:1], $\tilde{\gamma}$ is increasing in t and hence converges; it follows that $\|\mathbf{W}_+(t)\|_F^2 / \log t$ has a limit. By [Lyu and Li, 2020, Corollary A.11], $\|\mathbf{W}_+(t)\|_F^2 = \Theta(\log t)$, implying that the limit is finite and strictly positive. We will denote it by ν_+ and the analogous quantity for $\mathbf{W}_-$ by ν_-.

We now return to the main thread of the proof. We analyse the convergence of $\mathbf{W}(t)/\|\mathbf{W}(t)\|_F$ by analysing $\mathbf{W}_0/\|\mathbf{W}(t)\|_F$, $\mathbf{W}_+(t)/\|\mathbf{W}(t)\|_F$ and $\mathbf{W}_-(t)/\|\mathbf{W}(t)\|_F$ in turn. Since $\|\mathbf{W}(t)\|_F^2 = \|\mathbf{W}_0\|_F^2 + \|\mathbf{W}_+(t)\|_F^2 + \|\mathbf{W}_-(t)\|_F^2$,

$$\lim_{t\to\infty} \frac{\|\mathbf{W}(t)\|_F^2}{\log t} = \nu_+ + \nu_-. \tag{3.22}$$

Now observe that with probability at least $1-1/2^p$ over the random initialisation, $\nu_+ + \nu_- > 0$ (or equivalently, $\mathcal{J}_+ \cup \mathcal{J}_- \neq \emptyset$). To prove this, let $\mathbf{x}_{i+}$ be any training example with $y_{i+} = 1$ and let $\mathbf{x}_{i-}$ be any training example with $y_{i-} = -1$. Then by Lemma B.5, if a neuron $(a_j, \mathbf{w}_j)$ is initialised such that $a_j(0) > 0$ and $\mathbf{w}_j(0)^\intercal \mathbf{x}_{i+} > 0$ then for $t \geq 0$, $\mathbf{w}_j(t)^\intercal \mathbf{x}_{i+} > 0$. This holds in particular at time T. The neuron j thus cannot be in $\mathcal{J}_0$ nor $\mathcal{J}_-$, implying $j \in \mathcal{J}_+$. Similarly, if the neuron is initialised such that $a_j(0) < 0$ and $\mathbf{w}_j(0)^\intercal \mathbf{x}_{i-} > 0$, then $j \in \mathcal{J}_-$. The probability that one of the two initialisations occurs for a single neuron j is $1/2$, as $\mathbb{P}_{\mathbf{w}_j}\left[\, \mathbf{w}_j(0)^\intercal \mathbf{x} > 0 \,\right] = 1/2$ for any fixed $\mathbf{x}$. Hence, the probability that $j \in \mathcal{J}_0$ is at most $1 - 1/2 = 1/2$, and the probability that $[p] \subseteq \mathcal{J}_0$ is at most $1/2^p$.

It follows that with probability at least $1 - 1/2^p$,

$$\frac{\mathbf{W}_0}{\|\mathbf{W}(t)\|_F} \to \mathbf{0}, \qquad \text{as } t \to \infty. \tag{3.23}$$

Also, by eqs. (3.17) and (3.22),

$$\frac{\mathbf{W}_+(t)}{\|\mathbf{W}(t)\|_F} = \frac{\mathbf{W}_+(t)}{\|\mathbf{W}_+(t)\|_F} \cdot \frac{\|\mathbf{W}_+(t)\|_F / \sqrt{\log t}}{\|\mathbf{W}(t)\|_F / \sqrt{\log t}} \to \frac{\sqrt{\nu_+}}{\sqrt{\nu_+ + \nu_-}}\, \bar{\mathbf{u}} \mathbf{w}_+^\intercal, \tag{3.24}$$

and similarly

$$\frac{\mathbf{W}_-(t)}{\|\mathbf{W}(t)\|_F} \to \frac{\sqrt{\nu_-}}{\sqrt{\nu_+ + \nu_-}}\, \bar{\mathbf{z}} \mathbf{w}_-^\intercal. \tag{3.25}$$

For $j \in \mathcal{J}_+$ and $t \geq T$ we moreover know that if $y_i = 1$ then $\mathbf{w}_j(t)^\intercal \mathbf{x}_i > 0$ because $\mathbf{w}_j(t) \in \mathcal{S}_{\boldsymbol{\sigma}^+}$. As the same property holds for $\mathbf{w}_+$, it follows that $\bar{u}_j \geq 0$. By a similar argument, $\bar{z}_j \geq 0$. Combining the last three equations then proves eq. (3.8).

As for eq. (3.9), we know by Lemma B.4 that $a_j(t) = s_j \|\mathbf{w}_j(t)\|$ for some $s_j \in \{\pm 1\}$, implying $\|\mathbf{a}(t)\| = \|\mathbf{W}(t)\|_F$. Hence, for $j \in \mathcal{J}_+$,

$$\frac{a_j(t)}{\|\mathbf{a}(t)\|} = \frac{s_j \|\mathbf{w}_j(t)\|}{\|\mathbf{W}(t)\|_F} \to s_j \|u_j \mathbf{w}_+^\intercal\| \tag{3.26}$$

by eq. (3.24), where $u_j \geq 0$. For $j \in \mathcal{J}_+$ we also know that $a_j(t) \geq 0$, so $s_j = 1$ and

$$\frac{a_j(t)}{\|\mathbf{a}(t)\|} \to u_j \|\mathbf{w}_+\|. \tag{3.27}$$

By a similar argument, we obtain that for $j \in \mathcal{J}_-$,

$$\frac{a_j(t)}{\|\mathbf{a}(t)\|} \to -z_j \|\mathbf{w}_-\|. \tag{3.28}$$

Finally, for $j \in \mathcal{J}_0$, $a_j(t)$ is constant and so

$$\frac{a_j(t)}{\|\mathbf{a}(t)\|} \to 0. \tag{3.29}$$

□

3.4 Experiments

In this section, we first verify that the theoretical result (Theorem 3.1) is predictive of experimental outcomes, even when some technical assumptions are violated. Second, we present evidence that a similar result may hold for deeper networks as well, although this goes beyond Theorem 3.1. Third, we explore the regime beyond orthogonal separability.

3.4.1 Two-layer networks

To see how well the theory holds up, we train a two-layer ReLU network with 100 neurons on a synthetic orthogonally separable dataset consisting of 500 examples in $\mathbb{R}^{20}$. The dataset is constructed from an iid Gaussian dataset by filtering, to ensure orthogonal separability and $\mathbf{w}_+ \not\propto -\mathbf{w}_-$ (for visualisation purposes). Specifically, let $\mathbf{z} := [1, -1, \ldots, 1, -1]$. A Gaussian-sampled point $\mathbf{x}$ is included with label $+1$ if it lies in the first orthant and $\mathbf{x}^\intercal \mathbf{z} \geq 0$, included with label -1 if it lies in the orthant opposite to the first and $\mathbf{x}^\intercal \mathbf{z} \geq 0$, and discarded otherwise.

We train by stochastic gradient descent with batch size 50 and a learning rate of 0.1 for 500 epochs. At initialisation, we multiply all weights by 0.05. This reflects a setting where both key assumptions of Theorem 3.1 – orthogonal separability and small initialisation – hold, while the other assumptions are relaxed to approach real-life practice.

Figure 3.1 shows the results. Figure 3.1a shows the top 10 singular values of the first-layer weight matrix $\mathbf{W} \in \mathbb{R}^{100\times 20}$ after training. We see that despite its size, the matrix has rank only two: all singular values except the first two are effectively zero. This is exactly as predicted by the theorem. Furthermore, when we project the neurons on the positive-variance dimensions (Figure 3.1b), we see that they align along two main directions. To see how well these directions align with the predicted max-margin directions, we compute the correlation (normalised inner product) of each neuron with its respective max-margin direction. Figure 3.1c shows the histogram of these correlations. We see that the correlation is generally high, above 0.9 for most neurons. Overall we find very good agreement with theory.

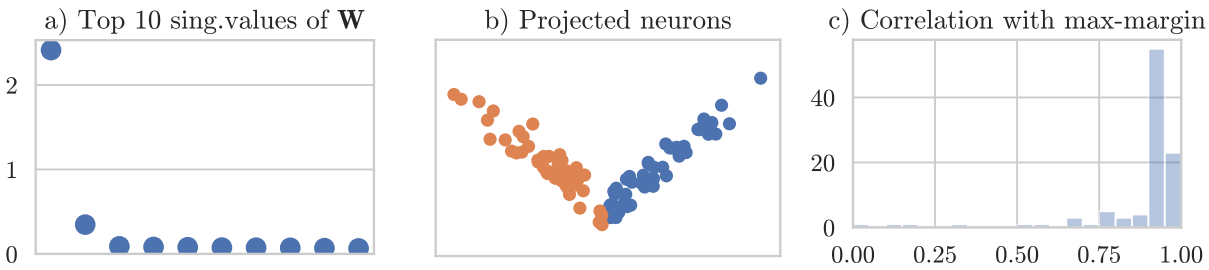


Figure 3.1: **a)** The 10 largest singular values of the first-layer weight matrix $\mathbf{W}$ after training. Each dot represents one singular value. **b)** Neurons (rows of $\mathbf{W}$) projected on the top two singular dimensions. Orange (or blue) dots represent neurons with $a_j > 0$ (or $a_j < 0$). **c)** Histogram of correlations between each neuron and its respective max-margin direction. (There are 100 neurons in total).

3.4.2 Deeper networks

We now explore the behaviour of deeper networks on orthogonally separable data. We train a residual network rather than a fully-connected one. The reason for this is that fully-connected networks with small initialisation are hard to train: early in training, the gradients are vanishingly small but then grow very quickly. We therefore found setting a numerically stable learning rate rather delicate.

We consider a residual network $h_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}$ parameterised by $\boldsymbol{\theta} \triangleq \{\mathbf{W}_1, \ldots, \mathbf{W}_L\}$, of the

form

$$
\begin{aligned}
h_{\boldsymbol{\theta}}^{1:1}(\mathbf{x}) &= \mathbf{W}_1\mathbf{x}, \\
h_{\boldsymbol{\theta}}^{1:l}(\mathbf{x}) &= h_{\boldsymbol{\theta}}^{1:l-1}(\mathbf{x}) + \mathbf{W}_l\rho(h_{\boldsymbol{\theta}}^{1:l-1}(\mathbf{x})), \qquad \text{for } l \in [2, L-1], \\
h_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbf{W}_L\rho(h_{\boldsymbol{\theta}}^{1:L-1}(\mathbf{x})),
\end{aligned}
\tag{3.30}
$$

where p is the network's width, and $\mathbf{W}_1 \in \mathbb{R}^{p\times d}$, $\mathbf{W}_l \in \mathbb{R}^{p\times p}$ and $\mathbf{W}_L \in \mathbb{R}^{1\times p}$ are its weights.

We train such a four-layer residual net with width 100 on the same dataset and using the same optimiser and hyper-parameters as in Section 3.4.1. Figure 3.2 shows the results. The results are very similar to what we observe for two-layer nets: the weight matrices are all rank two (Figure 3.2a-c), and the weight matrices' rows align in two main directions (Figure 3.2d-f). It is unclear what these directions are for the intermediate layers of the network, but for the first layer, we conjecture it is again the max-margin directions, as suggested by Figure 3.2g.

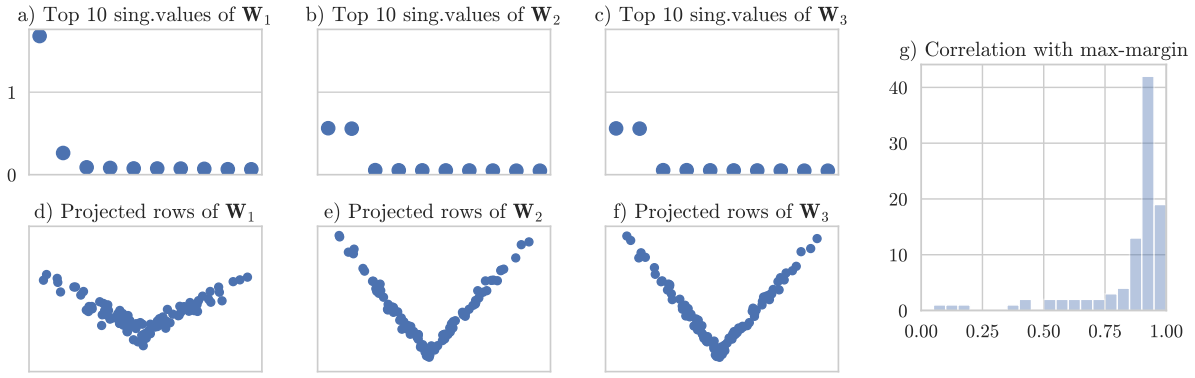


Figure 3.2: **a-c)** The 10 largest singular values of the first-, second- and third-layer weight matrix $\mathbf{W}_l$ after training. Each dot represents one singular value. **d-f)** Neurons (rows of $\mathbf{W}_l$) projected on the respective top two singular dimensions. **g)** Histogram of correlations between each first-layer neuron and the closest max-margin direction. (There are 100 neurons in total).

3.4.3 Beyond orthogonal separability

In this section we explore the applicability of our result to real-world datasets and architectures (which lie outside the scope formally covered by our assumptions). We experiment on the MNIST dataset subsetted to two classes, the digit `0` and the digit `1`.

We train a network consisting of six convolutional layers followed by two fully-connected layers. We view the six convolutional layers as a 'feature extractor' and the two fully-connected layers as a two-layer fully-connected network of the kind we analyse in this paper. The details of the architecture are given in Table 3.1. We train the network by Adam with the binary cross-entropy loss and a batch size of 128. We train for 50 epochs. Prior to training, we multiply the weights of the fully-connected layers by 0.05, to approximate the small-norm initialisation assumed by theory.

We conduct two sub-studies. First, we demonstrate that the network learns orthogonally separable representations all by itself, in the course of training. This is shown in Figure 3.3. The first subplot shows three distributions: The blue distribution is the distribution of $\mathbf{x}_i^\intercal\mathbf{x}_j$ where $\mathbf{x}_i$ is sampled from class `0` and $\mathbf{x}_j$ is sampled from class `1`. The orange (or green)

Layer	Type
1	`conv(32, 3, 1, 1)`
2	`conv(32, 3, 1, 1)`
3	`conv(32, 5, 2, 2)`
4	`conv(64, 3, 1, 1)`
5	`conv(64, 3, 1, 1)`
6	`conv(64, 5, 2, 2)`
7	`fc(3136, 128)`
8	`fc(128, 1)`

Table 3.1: Architecture of the studied network. By `conv(n, k, s, p)` we denote a convolutional layer with n kernels of size $k \times k$ with stride s, where the input to the layer is padded by p rows or columns on each margin. By `fc(n, o)` we denote a fully-connected layer whose input dimension is n and whose output dimension is o.

distribution is the distribution of $\mathbf{x}_i^\intercal \mathbf{x}_j$ where both $\mathbf{x}_i, \mathbf{x}_j$ are sampled from class `0` (or class `1`). The other subplots show analogous inner-product distributions for the intermediate representations or learned features of the data, i.e. $h_{\boldsymbol{\theta}}^{1:l}(\mathbf{x}_i)^\intercal h_{\boldsymbol{\theta}}^{1:l}(\mathbf{x}_j)$ instead of $\mathbf{x}_i^\intercal \mathbf{x}_j$.

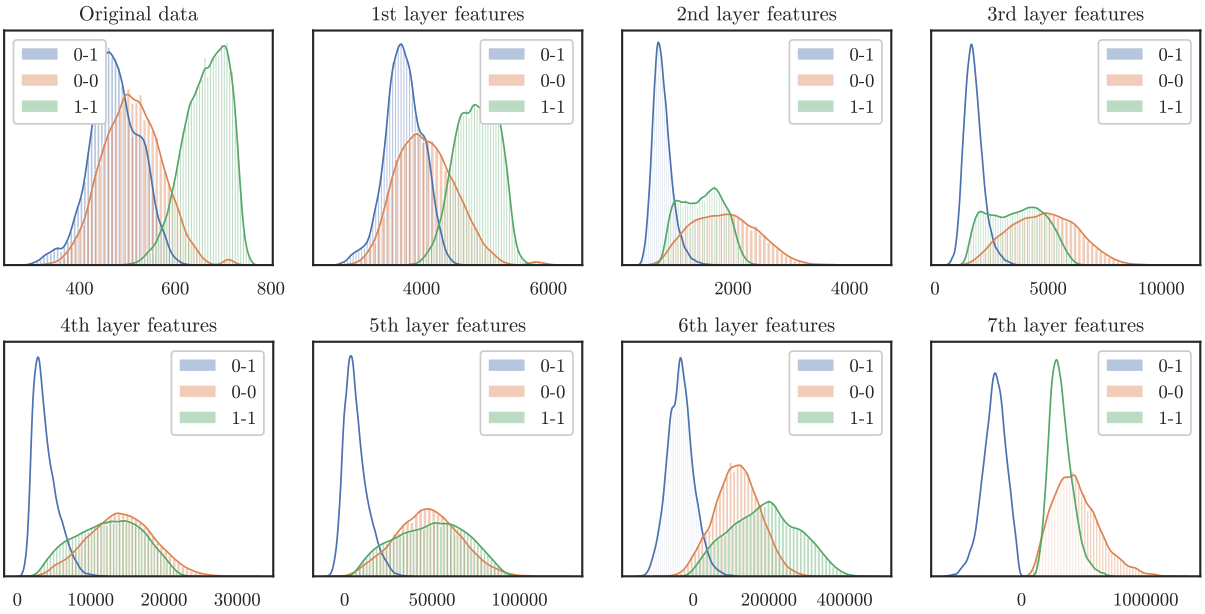


Figure 3.3: Distributions of feature similarity, where examples are sampled from the specified classes. Specifically, The l-th subplot shows the distribution of $h_{\boldsymbol{\theta}}^{1:l}(\mathbf{x}_i)^\intercal h_{\boldsymbol{\theta}}^{1:l}(\mathbf{x}_j)$ for $\mathbf{x}_i, \mathbf{x}_j$ sampled from different classes (blue) or both from class `0` (orange) or both from class `1` (green).

What we see is that the network learns representations such that examples of the same class are more similar to each other than examples of different classes – the orange and green distributions are generally more to the right compared to the blue distribution. Moreover, higher-layer representations are generally more strongly separated – as we move up the layer hierarchy, the orange and green distributions keep shifting rightward, whereas the blue distribution shifts leftward. Remarkably, the 7th layer representations are orthogonally separable.

In the second sub-study, we explore properties of the weight matrix learnt by the first linear

layer of the network, in analogy to the first-layer weight matrix in a two-layer net. Figure 3.4a shows the top ten singular values of the weight matrix $\mathbf{W}_7 \in \mathbb{R}^{128\times 3136}$. We see that despite its size, it has very few (perhaps five or ten) significantly non-zero singular values. This is similar to what we observed for synthetic data in Sections 3.4.1 and 3.4.2, though the separation between small and large singular values is less sharp and there are more than two non-zero values. Figure 3.4b shows the rows (neurons) of $\mathbf{W}_7$ projected onto the top two singular dimensions (note that unlike in Sections 3.4.1 and 3.4.2, the projection is lossy). The neurons roughly form three clusters: a mixed cluster close to the origin and two clusters corresponding to positive and negative outer-layer weights. Compared to our observations from Sections 3.4.1 and 3.4.2, there is less variation in the neurons' norms, leading to them forming clusters rather than rays. This deviation from the theoretical prediction could be due to a number reasons, e.g. the use of biases, convolutional layers, or the large dimensionality of the layer. We leave a detailed investigation of this question to future work.

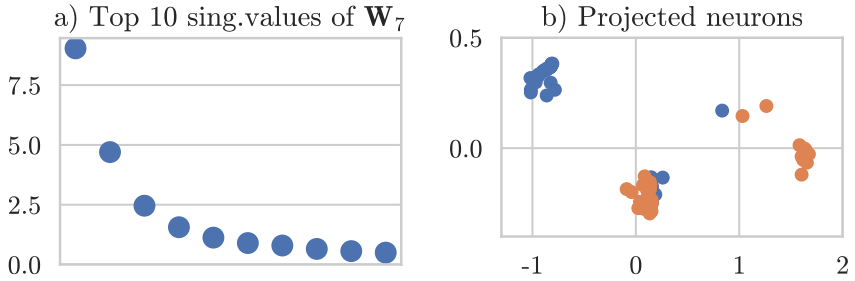


Figure 3.4: **a)** The 10 largest singular values of the first linear layer's weight matrix $\mathbf{W}_7$ after training. Each dot represents one singular value. **b)** Neurons (rows of $\mathbf{W}_7$) projected on the top two singular dimensions. Orange (or blue) dots represent neurons with $W_8[j] > 0$ (or $W_8[j] < 0$).

3.5 Related work

There is a lot of prior work on the implicit bias of gradient descent for various linear models. For logistic regression, Soudry et al. [2018] show that assuming an exponentially-tailed loss and linearly separable data, the normalised weight vector converges to the max-margin direction. Ji and Telgarsky [2019b] extend this result to non-separable data, Nacson et al. [2019] extend it to super-polynomially-tailed losses, and Gunasekar et al. [2018a] considers different optimisation algorithms. For deep linear networks, Ji and Telgarsky [2019a] show that the end-to-end weight matrix converges to the max-margin solution and consecutive weight matrices align. Gunasekar et al. [2018b] consider linear convolutional nets and prove convergence to a predictor related to the $\ell_{2/L}$ bridge penalty.

A few papers have started addressing the implicit bias problem for nonlinear (homogeneous or ReLU) networks. The problem is much harder and hence requires stronger assumptions. Lyu and Li [2020] and Ji and Telgarsky [2020] assume that at some point during training, the network attains perfect classification accuracy. Training from this point onward, Ji and Telgarsky [2020] show that the network parameters converge in direction. Lyu and Li [2020] show that this direction is a critical KKT point of the (nonlinear) max-margin problem. A complementary approach is taken by Maennel et al. [2018] who analyse the very early phase of training, when the weights are close to the origin. For two-layer networks, they show convergence of neurons to extremal sectors. Our work can be seen as a first step towards bridging the very early and the very late phase of training.

Zooming out a bit, there is also work motivated by similar questions, but taking a different approach. For example, Li and Liang [2018] show that two-layer ReLU nets trained on structured data converge to a solution that generalises well. Like ours, their analysis requires that the network's activation patterns change little, but they achieve it by containing training in the neighbourhood of the (relatively large) initialisation (this is the standard lazy training argument Chizat et al. [2019]). In contrast, we initialise much closer to zero, allowing the neurons to move more. Another related paper is Chizat and Bach [2020]. Using a mean-field analysis, the authors show that infinite-width two-layer ReLU nets converge to max-margin classifiers in a certain non-Hilbertian function space.

3.6 Conclusion

In this chapter, we have characterised the inductive bias of training ReLU networks in a simple setting: we proved that two-layer ReLU nets trained by gradient flow on orthogonally separable data converge to a combination of the positive and the negative max-margin classifier. Moreover, our experiments suggest that the same is true of deeper networks.

Such an implicit bias seems favourable – indeed, large margin has a long history in machine learning as an indicator of good generalisation. That deep networks implicitly prefer large margin classifiers, at least for orthogonally separable datasets, is reassuring. It remains a topic for future work to investigate whether similar results extend to more complicated datasets as well.

CHAPTER 4

The Mutual Autoencoder: Controlling information in latent code representations

In the previous chapter, we have seen an example of favourable inductive bias: in the face of underspecification, the DL algorithm chooses a hypothesis with a large margin, which can reasonably be expected to favour generalisation. In this chapter, we consider a setting where the default inductive bias is not enough.

Specifically, we explore representation learning with variational autoencoders (VAE) [Kingma and Welling, 2014, Rezende et al., 2014]. Variational autoencoders learn a latent variable model of the observable $\mathbf{x} \in \mathcal{X}$,

$$p_\theta(\mathbf{x}) = \int_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}, \tag{4.1}$$

by performing approximate maximum likelihood estimation (MLE),

$$\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i), \tag{4.2}$$

specifically, VAEs maximise a tractable lower bound of eq. (4.2). Beyond providing a good density model, a VAE model assigns to each data instance $\mathbf{x}$ a *latent code* $\mathbf{z}$. In many applications, this latent code provides a useful high-level summary of the observation. Because (4.1) can be learned from unlabelled data, we can use VAEs for unsupervised representation learning, an important building block in artificial intelligence systems.

However, it has been observed that the VAE may fail to learn a useful representation when the model class is very expressive. We argue that this is due to underspecification: the maximum likelihood criterion does not explicitly encourage useful representations and the latent variable is used only if it helps model the marginal distribution.[1] In particular, the amount of information stored in $\mathbf{z}$ depends on the expressiveness of the model $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with respect to the true data distribution [Chen et al., 2017]. In practice this has made the VAE approach difficult to work with in important applications such as natural language processing and for modelling discrete data.

[1] See also the article [Huszár, 2017].

To address this issue, we propose a method for explicitly controlling the amount of information stored in the latent code. The idea is illustrated in Figure 4.1: optimising the MLE criterion (eq. (4.2)) as in VAEs will lead to a model $\hat{p}$ which achieves a certain mutual information (MI) between $\mathbf{x}$ and $\mathbf{z}$ but this amount of information is difficult to predict and in fact may be zero [Chen et al., 2017, Zhao et al., 2019]. The proposed *mutual autoencoder* (MAE) approach forces information flow by ensuring that the estimated model $\hat{p}_m$ achieves a user-specified mutual information $M = I_{\hat{p}_m}$. Therefore, we precisely control the amount of bits stored in the representation but leave the organisation and use of this information to be learned.

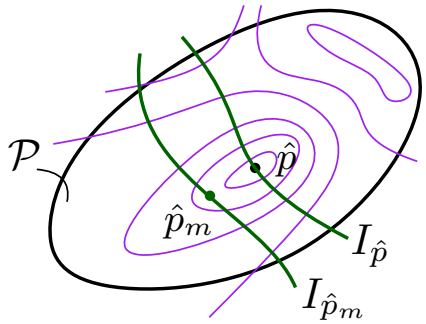


Figure 4.1: The mutual autoencoder maximises the likelihood $\log p_{\boldsymbol{\theta}}(x)$ while constraining model family $\mathcal{P}$ by ensuring that the latent variable $\mathbf{z}$ and the observable $\mathbf{x}$ share a desired mutual information $M = I_{\hat{p}_m}$. The purple lines denote level sets of the log-likelihood $\sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i)$.

The method can learn codes ranging from independent to nearly deterministic, while benefiting from model capacity. Thus, we decouple the choice of model capacity and the latent code dimensionality from concerns about underspecification.

To control information in the described way requires novel algorithms and the rest of this chapter discusses the procedure in more detail.

4.1 Background: Variational autoencoders

Consider a latent variable model with a data variable $\mathbf{x} \in \mathcal{X}$ and a latent variable $\mathbf{z} \in \mathcal{Z}$, $p_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Given the data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we would like to train the model by maximising the marginal log-likelihood,

$$\ell(\boldsymbol{\theta}) \triangleq \mathop{\mathbb{E}}_{\mathbf{x}\sim q} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] \triangleq \mathop{\mathbb{E}}_{\mathbf{x}\sim q} \left[\log \int_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}\right], \tag{4.3}$$

where $q(\mathbf{x})$ denotes the empirical distribution of $\mathbf{x}$, $q(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \delta_{\mathbf{x}_i}(\mathbf{x})$. However, the integral in (4.3) is intractable in many applications of interest. The idea behind variational methods is to instead maximise a lower bound $L(p_{\boldsymbol{\theta}}, q)$ to the log-likelihood, where

$$L(p, q) \triangleq \mathop{\mathbb{E}}_{\mathbf{x}\sim q} \left[\mathop{\mathbb{E}}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))\right], \tag{ELBO}$$

where D_{KL} denotes the KL divergence. Any choice of $q(\mathbf{z}|\mathbf{x})$ gives a valid lower bound $L(p_{\boldsymbol{\theta}}, q) \leq \ell(\boldsymbol{\theta})$, with a gap of $\mathbb{E}_{\mathbf{x}\sim q}\left[D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x}))\right]$.

VAEs [Kingma and Welling, 2014, Rezende et al., 2014] replace the per-instance posteriors $q(\mathbf{z}|\mathbf{x})$ by an inference network $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ that is trained together with $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ to jointly maximise $L(p_{\boldsymbol{\theta}}, q_{\boldsymbol{\theta}})$. For the inference network $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, this is equivalent to minimising $D_{\mathrm{KL}}(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$, so one can think of $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ as an approximation of $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

Being a stochastic mapping of data $\mathbf{x}$ to a latent code $\mathbf{z}$, the inference network $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is sometimes called an *encoder* and, by the same analogy, the generator network $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is called a *decoder*.

4.2 VAE objective is insufficient for representation learning

A major appeal of the VAE framework is the ability to learn meaningful latent codes $\mathbf{z}$ from unlabelled or only weakly labelled data. Despite numerous promising results on image and video datasets (e.g. Higgins et al. [2017], Bouchacourt et al. [2018], or Goyal et al. [2017]), the application of VAEs to text has proven challenging. Specifically, Bowman et al. [2016] found that a VAE with an LSTM-based decoder fails to learn a useful latent code when trained naively – the approximate posterior collapses to the prior, $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$ for all inputs $\mathbf{x}$, leading to a near independent relationship between $\mathbf{x}$ and $\mathbf{z}$.

A number of works have addressed this problem, mostly treating it as an optimisation issue [Sønderby et al., 2016, Kingma et al., 2016, Yeung et al., 2017]. However, as Chen et al. [2017] point out, even if one could optimise exactly, the model would still learn trivial latent codes when using a high-capacity decoder such as an LSTM.

The reason for this is underspecification. Note that the log-likelihood (4.3) is only a function of the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$, whereas representation is an aspect of the joint distribution $p_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{x})$. That is, we approximately optimise a marginal quantity in the hope of producing the desired effect on the joint distribution. This approach is unreliable, although it works when the marginal imposes strong constraints on the joint distribution, such as when the decoder has limited capacity and the model is forced to use the latent structure to reach a high likelihood. However, in the high-capacity regime, e.g. when $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = q(\mathbf{x})$ is close to achievable, the task of density estimation does not constrain the model enough toward representation learning [Huszár, 2017].

We propose an alternative optimisation problem that better specifies the representation objective. The idea is to explicitly control the amount of information stored in the latent code, as measured by the mutual information. In the following sections, we derive the procedure in detail and show that it enables representation learning with powerful decoders.

4.3 Controlling information in latent code representations

Our goal is to learn a deep latent variable model $p_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, while precisely controlling the coupling between the latent code $\mathbf{z}$ and the output $\mathbf{x}$. We formalise our goal in the following constrained optimisation problem,

$$\max_{\boldsymbol{\theta}} \quad \mathop{\mathbb{E}}_{\mathbf{x}\sim q(\mathbf{x})}\left[\log \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}\right], \tag{4.4}$$

$$\text{s.t.} \quad I_{p_{\boldsymbol{\theta}}}(\mathbf{z}, \mathbf{x}) = M, \tag{4.5}$$

where $M \geq 0$ is a scalar constant denoting the desired *mutual information*,

$$I_{p_{\boldsymbol{\theta}}}(\mathbf{z}, \mathbf{x}) = \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p_{\boldsymbol{\theta}}} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{x})}{p(\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{x})} \right], \tag{4.6}$$

between $\mathbf{x}$ and $\mathbf{z}$ as encoded in the model $p_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{x})$. When M is close to zero, the code $\mathbf{z}$ is uninformative about $\mathbf{x}$, whereas a large value of M approaching the entropy $H_{p_{\boldsymbol{\theta}}}(\mathbf{x})$, the maximum possible value, should lead to a deterministic relation between $\mathbf{x}$ and $\mathbf{z}$. Intermediate values of M will lead to lossy codes $\mathbf{z}$ which recover a compressed representation of structure in $q(\mathbf{x})$ that can be most efficiently captured by $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.

4.4 The Mutual Autoencoder (MAE)

We now describe our idea for approximately solving the problem (4.4–4.5). We call our approach the *mutual autoencoder*.

The method relies on two results from the literature: 1) exact penalty functions [Zangwill, 1967] to accommodate the equality constraint (4.5) into the estimation problem; 2) the variational infomax bound, [Barber and Agakov, 2003], to approximate the intractable mutual information $I_{p_{\boldsymbol{\theta}}}(\mathbf{z}, \mathbf{x})$.

We now briefly describe each of them and their role in the mutual autoencoder.

4.4.1 Exact penalty functions

To approximately solve (4.4–4.5), we resort to a classic method for constrained nonlinear optimisation, the *penalty function method*. First proposed by Zangwill [1967], this method uses a positive penalty constant $C > 0$ to relax (4.4–4.5) to the unconstrained problem

$$\max_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\mathbf{x}\sim q(\mathbf{x})} \left[\log \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z} \right] - C\, |I_{p_{\boldsymbol{\theta}}}(\mathbf{z}, \mathbf{x}) - M|. \tag{4.7}$$

In (4.7), any deviation of $I_{p_{\boldsymbol{\theta}}}(\mathbf{z}, \mathbf{x})$ from M is penalised linearly. For values of C large enough, an unconstrained maximum of (4.7) recovers a feasible local maximum of (4.4–4.5). In particular, we do not need to take C all the way to infinity: a finite value is sufficient [Han and Mangasarian, 1979] and the magnitude of this value is determined by the unknown optimal Lagrange multiplier for (4.5) (see Bertsekas [1999, Proposition 4.3.1]). In practice, we observe that values of C in the range $[0.1, 10]$ work best.

We will show experimentally that the above penalty approach is highly effective. However, to leverage the approach, we also need to approximate the intractable mutual information $I_{p_{\boldsymbol{\theta}}}(\mathbf{z}, \mathbf{x})$.

4.4.2 Variational information maximisation – infomax inequality

To approximate the mutual information, we leverage the variational infomax inequality of Barber and Agakov [2003]:

Theorem 4.1. *For any two random variables $\mathbf{z}$ and $\mathbf{x}$ distributed according to the joint distribution with a density $p(\mathbf{z}, \mathbf{x})$ and for any conditional density function $r(\mathbf{z}|\mathbf{x})$ we have*

$$I_p(\mathbf{z}, \mathbf{x}) \geq H_p(\mathbf{z}) + \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p} [\log r(\mathbf{z}|\mathbf{x})]. \tag{4.8}$$

A proof is provided in Appendix C.1.

Equality in (4.8) is attained for $r(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$, so we can write

$$I_p(\mathbf{z}, \mathbf{x}) = H_p(\mathbf{z}) + \max_r \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r(\mathbf{z}|\mathbf{x})]. \tag{4.9}$$

In our method, we constrain the inner search over r to a parametric class, yielding the following lower bound approximation $\hat{I}_p(\mathbf{z}, \mathbf{x}) \leq I_p(\mathbf{z}, \mathbf{x})$, to the mutual information:

$$\hat{I}_p(\mathbf{z}, \mathbf{x}) = H_p(\mathbf{z}) + \max_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})]. \tag{4.10}$$

4.4.3 The Mutual Autoencoder

We are now in the position to combine the variational infomax approximation (4.10) with the penalty function formulation (4.7) to define the following *mutual autoencoder objective* $L_m(p, q)$ to be maximised over the distributions p and q.

$$\begin{aligned} L_m(p, q) &\triangleq L(p, q) - C\left|\hat{I}_p(\mathbf{z}, \mathbf{x}) - M\right| \\ &= L(p, q) - C\left|H_p(\mathbf{z}) + \max_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})] - M\right|. \end{aligned} \tag{4.11}$$

We now discuss how to estimate the gradient of the MAE objective, to facilitate standard gradient-based training. The first term in eq. (4.11) corresponds to the VAE objective and its gradient can be estimated by methods from the literature. For example, the reparametrisation trick [Kingma and Welling, 2014, Rezende et al., 2014] provides unbiased and low-variance gradient estimates when the latent variable $\mathbf{z}$ is continuous.

In the second term in eq. (4.11), the quantity $H_p(\mathbf{z})$ is typically a constant because we fix the prior $p(\mathbf{z})$ to a simple distribution such as a multivariate normal distribution. The difficulty instead lies in differentiating $\max_{\boldsymbol{\omega}} \mathbb{E}_{(\mathbf{z},\mathbf{x})\sim p}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})]$ with respect to p. Denoting by r_p^* the optimal $r_{\boldsymbol{\omega}}$ corresponding to p, one can derive by *REINFORCE*-style reasoning (see Appendix C.2) that

$$\begin{aligned} \nabla_p \max_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})] &= \nabla_p \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r_p^*(\mathbf{z}|\mathbf{x})] \\ &= \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}\left[(\nabla_p \log p(\mathbf{z}, \mathbf{x})) \cdot \log r_p^*(\mathbf{z}|\mathbf{x}) + \nabla_p \log r_p^*(\mathbf{z}|\mathbf{x})\right]. \end{aligned} \tag{4.12}$$

There are two difficulties here: 1) The last term in (4.12), which arises from the dependence of r_p^* on p, is difficult to compute. 2) Evaluating r_p^* requires solving a separate optimisation program in $\boldsymbol{\omega}$, which is too expensive to do for each gradient evaluation.

To address the first issue, we note that in the non-parametric limit, as $r_p^*(\mathbf{z}|\mathbf{x}) \approx p(\mathbf{z}|\mathbf{x})$, the problematic term vanishes (see Appendix C.2). We therefore ignore it, which is equivalent to treating r_p^* as independent of p during back-propagation. For the second issue, we keep a running estimate of the optimal r_p^* and perform a single gradient update to it whenever p is updated. This gives rise to the practical procedure described in Algorithm 4.1.

Note that unlike (4.12), the term $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{z},\mathbf{x})\sim p_{\boldsymbol{\theta}}}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})]$ in Algorithm 4.1, line 14, can be efficiently approximated via the reparametrisation trick if $\mathbf{x}$ is continuous or *REINFORCE* [Williams, 1992] if $\mathbf{x}$ is discrete. We provide a derivation of the necessary gradients in Appendix C.3.

Algorithm 4.1 Mutual Autoencoder training

procedure TRAINMAE($\boldsymbol{\theta}, \boldsymbol{\omega}, B, C, M, N$) // Batch size, penalty factor, MI target, opt. steps.
  for $i = 1, \ldots, N$ **do**
    UPDATEMODEL($\boldsymbol{\theta}, \boldsymbol{\omega}, B, C, M$) // We simultaneously optimise the model...
    UPDATEMIESTIMATE($\boldsymbol{\omega}, \boldsymbol{\theta}, B$) // ...and the mutual information estimate.
  end for
end procedure

procedure UPDATEMIESTIMATE($\boldsymbol{\omega}, \boldsymbol{\theta}, B$)
  Sample $(\mathbf{z}_i, \mathbf{x}_i) \sim p_{\boldsymbol{\theta}}$ for $i = 1, \ldots, B$
  $g \leftarrow \frac{1}{B} \sum_{i=1}^{B} \nabla_{\boldsymbol{\omega}} \log r_{\boldsymbol{\omega}}(\mathbf{z}_i|\mathbf{x}_i)$ // Gradient estimate of the infomax bound.
  $\boldsymbol{\omega} \leftarrow$ Update$(\boldsymbol{\omega}, g)$
end procedure

procedure UPDATEMODEL($\boldsymbol{\theta}, \boldsymbol{\omega}, B, C, M$)
  $g_{\text{ELBO}} \leftarrow$ EstimateElboGradient$(\boldsymbol{\theta})$
  $g_{\text{MI}} \leftarrow$ Estimate of $\nabla_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{z},\mathbf{x})\sim p_{\boldsymbol{\theta}}}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})]$ // Using reparametrisation trick or *REINFORCE*.
  Sample $(\mathbf{z}_i, \mathbf{x}_i) \sim p_{\boldsymbol{\theta}}$ for $i = 1, \ldots, B$
  $m \leftarrow H_p(\mathbf{z}) + \frac{1}{B} \sum_{i=1}^{B} \log r_{\boldsymbol{\omega}}(\mathbf{z}_i|\mathbf{x}_i)$ // Mutual information estimate.
  $\boldsymbol{\theta} \leftarrow$ Update$(\boldsymbol{\theta}, g_{\text{ELBO}} - C \cdot \operatorname{sign}(m - M) \cdot g_{\text{MI}})$
end procedure

4.5 Discrete data requires flexible encoder distributions

Before presenting experimental results, we discuss the choice of encoder distribution, which we find to be critical for training a good MAE model. More generally, it turns out that flexible encoder distributions are essential for VAE representation learning of discrete data. Below we give the formal statement and discuss its practical implications.

Theorem 4.2. *Consider a VAE model (p, q) applied to discrete data. Assume that both the prior $p(\mathbf{z})$ and the encoder distribution $q(\mathbf{z}|\mathbf{x})$ are Gaussian and that the decoder $p(\mathbf{x}|\mathbf{z})$ is powerful enough to model the true marginal $q(\mathbf{x})$. Then the independent configuration given by $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z}) = q(\mathbf{x})$ is* the only *global optimum of the VAE objective.*

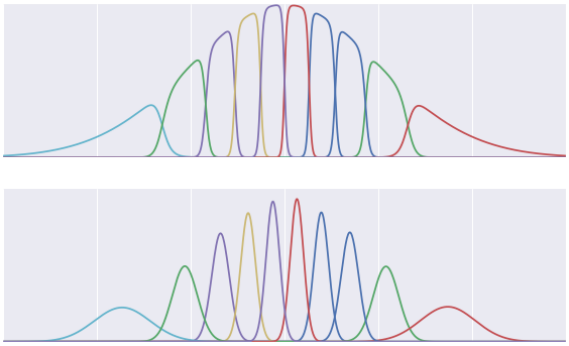
Proof. Maximising the VAE objective $L(p, q)$ is equivalent to minimising the Kullback-Leibler divergence between the joint distributions q and p, $D_{\text{KL}}(q(\mathbf{x})\, q(\mathbf{z}|\mathbf{x}) \| \, p(\mathbf{z})\, p(\mathbf{x}|\mathbf{z}))$. The VAE optimum is attained when the KL divergence is zero i.e. when $q(\mathbf{z}, \mathbf{x}) = p(\mathbf{z}, \mathbf{x})$ almost everywhere. The independent solution clearly satisfies this.

To show that all other configurations are suboptimal, let p, q be such that $q(\mathbf{z}|\mathbf{x}) \neq p(\mathbf{z})$ for some value of $\mathbf{x}$ with $q(\mathbf{x}) > 0$. Then the implied marginal $q(\mathbf{z}) = \sum_{\mathbf{x}\in\mathcal{X}} q(\mathbf{z}|\mathbf{x})\, q(\mathbf{x})$ is a finite Gaussian mixture with at least one nonstandard component, so it cannot equal $p(\mathbf{z})$ and the KL divergence is positive. □

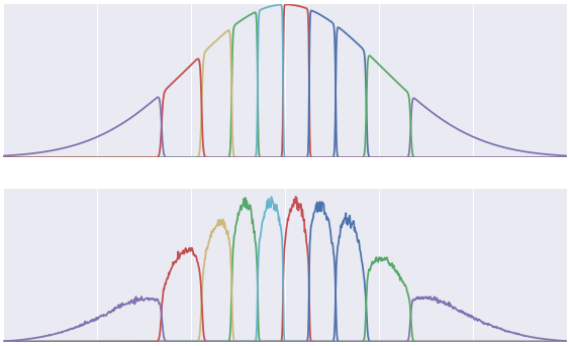
The theorem shows that not only is the VAE agnostic to representation learning, it in fact encourages the independent solution. The Gaussianity assumption introduces a gap due to $q(\mathbf{z}) \triangleq \sum_{\mathbf{x}\in\mathcal{X}} q(\mathbf{z}|\mathbf{x}) q(\mathbf{x})$ not being able to fit $p(\mathbf{z})$.

We now illustrate the effect on a toy task of modelling $\mathbf{x} \sim \text{Unif}[10]$ with a one-dimensional normal latent space. We train a MAE, setting the mutual information target M to the theoretical maximum, i.e. we aim to learn a deterministic code.

Figure 4.2 shows the difference between using the standard Gaussian encoder (left column) and a more complex resampling-based encoding distribution of Cremer et al. [2017] with



(a) MAE trained with a Gaussian encoder. Top: true model posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Bottom: posterior approximation $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Each curve corresponds to one possible value of $\mathbf{x}$.



(b) MAE trained with a resampling-based encoder. Top: true model posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Bottom: posterior approximation $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. Each curve corresponds to one possible value of $\mathbf{x}$.

Figure 4.2: The effect of encoder flexibility.

$k = 5$ samples (right column). The top plot in each column shows the exact model posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ and the bottom plot in each column shows the approximate posterior $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. It can be seen that not only does the resampling-based MAE learn better posterior approximations (the bottom plot is closer to the top plot), it also enables the decoder to train more effectively, as reflected in sharper, closer-to-deterministic model posteriors $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

We have observed this problem in larger-scale experiments as well; in fact, we find that the problem tends to get worse with the dimension of the latent space as well as the amount of encoded information M. In the experiments in Section 4.6.3 we mitigate the issue by employing the resampling-based encoder.

4.6 Experiments

We show that the mutual autoencoder can learn latent codes ranging from independent to nearly deterministic. First, we consider two toy examples (a continuous and a discrete one), where we can visualise important quantities. Then we show promising results on text data.

4.6.1 Splitting the normal

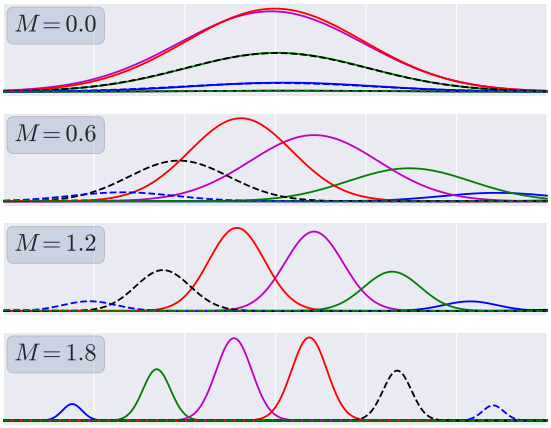
We compare our method to the variational autoencoder on the task of modeling the one-dimensional standard normal distribution, $q(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\, \mu = 0, \sigma = 1)$. Our goal here is to a) show in a minimal setting the issue of *representation collapse* in VAEs, and b) show that the MAE overcomes this problem and in fact can encode any pre-specified amount of information.

We now describe the experimental setup. The prior $p(\mathbf{z})$ is assumed to be a one-dimensional standard normal. Both the decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ and encoder $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ are modeled as normal, with means and log-variances parametrised by a three-layer fully connected network. We train a VAE and several instances of the MAE with different values of the mutual information target M.

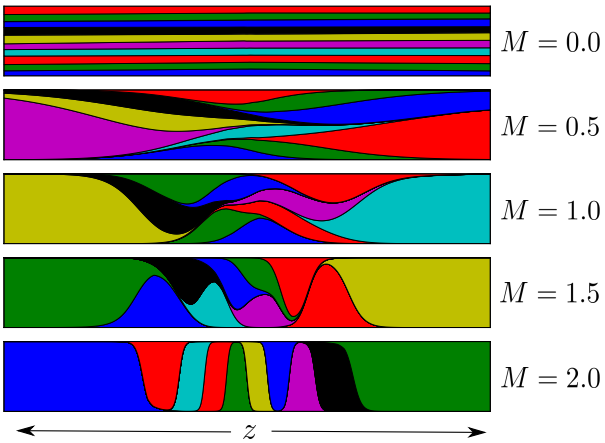
One can think of the task as splitting the normal $q(\mathbf{x})$ into an infinite mixture of normals $\int p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z}) \mathrm{d}\mathbf{z}$. There exists a trivial optimum of the VAE objective $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = q(\mathbf{x}) =$

$\mathcal{N}(\mathbf{x}|\,\mu = 0, \sigma = 1), q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}) = \mathcal{N}(\mathbf{x}|\,\mu = 0, \sigma = 1)$ that ignores the latent code. Indeed, this is the solution recovered by the VAE and the MAE with $M = 0$, corresponding to the top row of Figure 4.3a.

However, when we set $M > 0$, the MAE learns a non-trivial representation, as can be seen in Figure 4.3a. As we increase M, the conditionals $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ become more peaked and carry more information about $\mathbf{z}$.



(a) MAE splits the normal. Each row corresponds to a MAE model with the specified mutual information target M. Each curve corresponds to the rescaled conditional $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\,p(\mathbf{z})$ for $\mathbf{z}$ in the fixed set $\{-2.1, -1.3, -0.4, +0.4, +1.3, +2.1\}$.



(b) Categorical example, modeling the uniform distribution over 10 labels. Each row corresponds to a MAE model with the specified mutual information target M. Each colored patch corresponds to one possible outcome of $\mathbf{x}$. A vertical slice at position $\mathbf{z}$ consists of 10 line segments whose lengths (sometimes 0) indicate the probabilities $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.

Figure 4.3: The mutual autoencoder learns codes ranging from independent to deterministic.

4.6.2 Categorical example

VAEs are particularly prone to ignoring the latent code when modeling discrete data. Here we show that the MAE is an effective solution even in this setting, and is able to learn a wide range of codes.

We consider synthetic data drawn from $q(\mathbf{x}) = \text{Unif}[10]$. We let the latent variable again be a one-dimensional standard normal. The decoder is a three-layer fully connected network with a 10-way softmax output; the encoder is given by 10 separate normal distributions, one for each possible value of $\mathbf{x}$.

As in the continuous case, the decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ can easily reach an optimum of the VAE objective by outputting the marginal, $\text{Unif}[10]$, irrespective of $\mathbf{z}$. What makes the discrete case more difficult is that *any* distribution over $\mathcal{X}$ can be represented independently of the latent $\mathbf{z}$. Such a scenario is shown in the top row of Figure 4.3b; this is the model learned by the VAE and the MAE with $M = 0$.

For higher values of M, the MAE avoids this problem. In Figure 4.3b, we see that as M increases, the conditionals $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ learned by MAE move from independence towards determinism.

4.6.3 Movie reviews

We now demonstrate the effectiveness of the mutual autoencoder on real text data.

We consider a sentence modelling task using a subset of the IMDB movie review dataset [Diao et al., 2014]. We split each review into sentences and extract those of length 8 words or shorter. We train several instances of the MAE with different M, using a variant of the bidirectional LSTM [Schuster and Paliwal, 1997, Hochreiter and Schmidhuber, 1997] as the encoder and a standard LSTM as the decoder. The conditioned value of the latent variable $\mathbf{z}$ is fed to the decoder LSTM at each step.

To evaluate the information content of a given model's latent code, we perform a simple reconstruction experiment: we use the model to encode a random subset of the training data and decode the obtained representations back into sentence space. A model with a highly informative latent code should be able to approximately reconstruct the input. We consider two metrics: 1) the number of exactly reconstructed sentences, and 2) the number of matching words between the input sentence and the reconstruction.

Figure 4.4 shows the results for MAE models trained with different M. The graph shows a simple monotonic relation: as M increases, so does the amount of encoded information, which in turn is reflected by the model's ability to autoencode. The MAE provides a powerful way of controlling this behavior by setting M.

In Table 4.1 we show sample sentence reconstructions of a VAE and two MAE models. As expected, the VAE fails to learn a useful latent code, $q_{\boldsymbol{\theta}}^{\mathsf{VAE}}(\mathbf{z}|\mathbf{x}) \approx p(\mathbf{z})$, so its 'reconstructions' are random samples from the decoder model. At the other extreme, the MAE with a high value of the target mutual information (last row) learns a close-to-deterministic code and reconstructs the input sentence with high fidelity, at the expense of sample diversity. MAEs trained with intermediate values of M learn to 'paraphrase' input sentences.

To further inspect the learned representations, we interpolate between sentences in the latent space. An example is shown in Table 4.2. We see that the model picks up on syntactic elements such as specific word choice or sentence length and that the generated sentences are mostly grammatical (subject to limitations of the training data).

Input	there are many great scenes of course .
VAE	and he knows it too .
	terrific performances from all three stars .
	this movie could have been a classic .
MAE ($M = 5$)	there are things that i liked .
	as a whole it works pretty well .
	there were a few good performances too .
MAE ($M = 10$)	there were many scenes like that .
	there are many great scenes of course .
	there were many scenes like that .

Table 4.1: Sentence reconstructions. The input sentence is encoded and decoded by a given model, and the output is displayed. We show three reconstructions per model.

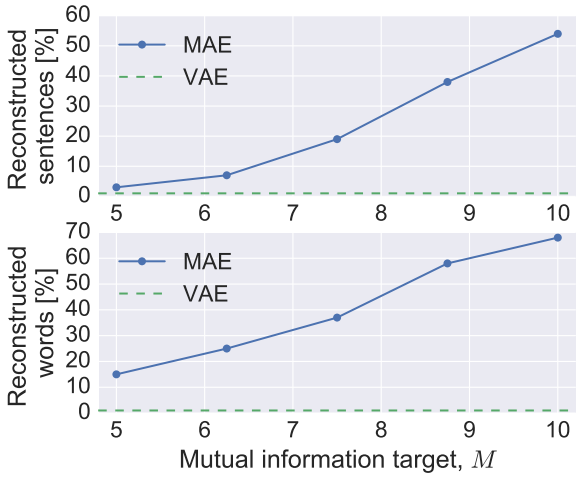


Figure 4.4: Reconstruction grows with M.

there are many great scenes of course .
there are other good people as well .
they so look real its incredible .
they like different , not good .
they like different , not good .
these people talk way too much .
die hard gets a few character .
i love these kind of movies .
i left out a character .
i love those films) .

Table 4.2: Sentence interpolation, $M = 10$.

4.7 Related Work

To our knowledge, the first work to notice the failures of VAE-type models when combined with high-capacity decoders was [Bowman et al., 2016]. Since then, a number of authors have reported similar difficulties and many fixes have been proposed.

Probably the most popular of these is 'KL cost annealing' [Bowman et al., 2016] or 'warm-up' [Sønderby et al., 2016], whereby the KL term in the VAE objective gets a multiplicative weight that gradually increases from zero to one during training. Another class of approaches is based on carefully limiting the capacity of the generative model class, as in [Chen et al., 2017], [Yang et al., 2017], or [Yeung et al., 2017]. Finally, some works make use of auxiliary objectives: Kingma et al. [2016] introduce a 'free bits' constraint on the minimum value of the KL term, whereas [Semeniuta et al., 2017] add a reconstruction term of an intermediate network layer. Zhao et al. [2019] propose a more drastic change to the objective, either replacing the KL term by one derived from a different divergence or removing it completely.

Our approach has close connections to the work of Chen et al. [2016], who also use the variational infomax bound to learn meaningful representations. Their work is based on the GAN framework of Goodfellow et al. [2014] and hence may be difficult to apply to text. The work of Hu et al. [2017] can also be seen as employing (an approximation of) mutual information to enforce coupling between the latent and output variable, although they do not state it explicitly.

4.8 Conclusion

In this chapter, we considered the problem of representation collapse in VAEs with high-capacity decoders. We argue that at the core of the issue is underspecification: the VAE objective is not sufficient to encourage learning of useful representations. To deal with this, we propose to augment the objective by a hand-crafted regulariser to better constrain the problem; in this case a term controlling the amount of information stored in the latent code. We have seen that the proposed approach works well in practice, being able to learn a continuum of representations with varying properties.

CHAPTER 5

Towards understanding knowledge distillation

In this chapter, we will explore another way of dealing with underspecification: strong supervision, or simply more informative data. We study the specific setting of binary classification with *soft labels*, i.e. labels taking values in some continuous interval, say $[0, 1]$, instead of the usual binary set $\{0, 1\}$. Such labels carry more information about the ground-truth hypothesis, lessening the curse of underspecification.

In deep learning, soft-label supervision is most common in the context of *knowledge distillation* [Hinton et al., 2014], a setting where one classifier (called the *student*) is trained on the real-valued outputs of another classifier (called the *teacher*). It has been observed that classifiers learn much faster and more reliably when trained with this kind of supervision rather than with ground-truth data, even when the teacher had originally been trained on exactly the same data. Distillation also makes optimisation easier: it reduces the need for regularisation and various optimisation tricks. Consequently, in several fields, distillation has become the standard technique for transferring information between classifiers with different architectures, such as from deep to shallow neural networks or from ensembles of classifiers to individual ones.

While the practical benefits of distillation are beyond doubt, its theoretical justification remains almost completely unclear. In this chapter, we study distillation in a simplified, analytically tractable setting: binary classification with linear teacher and linear student (either shallow or deep linear networks). For this setting, we prove a generalisation bound establishing fast convergence of the expected risk of a distillation-trained linear classifier. The result makes use of the interaction between strong supervision and the inductive bias to guarantee faster rates than classically achievable.

5.1 Related work

Ideas underpinning distillation have a long history dating back to the work of Ba and Caruana [2014], Buciluǎ et al. [2006], Craven and Shavlik [1996], Li et al. [2014], Liang et al. [2008]. In its current and most widely known form, it was introduced by Hinton et al. [2014] in the context of neural network compression.

Since then, distillation has quickly gained popularity among practitioners and established its place in deep learning folklore. It has been found to work well across a wide range of applications, including e.g. transferring from one architecture to another [Geras et al., 2016], compression [Howard et al., 2017, Polino et al., 2018], integration with first-order logic [Hu et al., 2016] or other prior knowledge [Yu et al., 2017], learning from noisy labels [Li et al., 2018], defending against adversarial attacks [Papernot et al., 2016], training stabilisation [Romero et al., 2015, Tang et al., 2016], few-shot learning [Kimura et al., 2018], distributed learning Polino et al. [2018], reinforcement learning Rusu et al. [2016] and data privacy Celik et al. [2017].

In contrast to the empirical success, the mathematical principles underlying distillation's effectiveness have largely remained a mystery. To our knowledge, Lopez-Paz et al. [2016] is the only work that examines distillation from a theoretical perspective. It casts distillation as a form of learning using privileged information (LUPI, Vapnik and Izmailov 2015), a learning setting in which additional per-instance information is available at training time but not at test time. However, even the LUPI view conceptually falls short of explaining the effectiveness of distillation. In particular, it concentrates on the aspect that the teacher's supervision to the student is noise-free. However, this argument does not suffice to explain the success of distillation even when the original problem is noise-free to start with.

A more distantly related topic is *machine teaching* Zhu [2015]. In machine teaching, a machine learning system is trained by a human teacher, whose goal is to hand-pick as small a training set as possible, while ensuring that the machine learns a desired hypothesis. Transferring knowledge via machine teaching techniques is extremely effective: perfect transfer is often possible from a small finite teaching set Zhu [2013], Liu and Zhu [2016]. However, the price for this radical reduction in sample complexity is the expensive training set construction. Our work shows that, at least in the linear setting, distillation achieves a similar effectiveness with a more practical form of supervision.

5.2 Background: Linear distillation

We formally introduce distillation in the context of binary classification. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the input space, $\mathcal{Y} = \{0, 1\}$ the label space, and $P_{\mathbf{x}}$ the probability distribution of inputs. We assume $P_{\mathbf{x}}$ has a density.

The *teacher* $h^* : \mathcal{X} \to \mathcal{Y}$ is a fixed linear classifier, i.e. $h^*(\mathbf{x}) = \mathbb{1}\{\mathbf{w}_*^\intercal \mathbf{x} \geq 0\}$ for some $\mathbf{w}_* \in \mathbb{R}^d \setminus \{\mathbf{0}\}$, where $\mathbb{1}\{.\}$ returns 1 if the argument is true and 0 otherwise. The *student* also is a linear classifier, $h(\mathbf{x}) = \mathbb{1}\{\mathbf{w}^\intercal \mathbf{x} \geq 0\}$.

We allow the setting where the weight vector is parameterised as a product of matrices, $\mathbf{w}^\intercal \triangleq \mathbf{W}_L \mathbf{W}_{L-1} \cdots \mathbf{W}_1$ for some $L \geq 1$. When $L \geq 2$, this parameterisation is known as a *deep linear network*. Although deep linear networks have no additional capacity compared to directly parameterised linear classifiers ($L = 1; \mathbf{w}^\intercal = \mathbf{W}_1$), they induce different gradient-descent dynamics, and are often studied as a first step towards understanding deep nonlinear networks Saxe et al. [2014], Kawaguchi [2016], Hardt and Ma [2017].

Distillation proceeds as follows. First, we collect a *transfer set* $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ consisting of inputs $\mathbf{x}_i$ sampled i.i.d. from $P_{\mathbf{x}}$, and *soft labels* $y_i = \sigma(\mathbf{w}_*^\intercal \mathbf{x}_i)$ provided by the teacher, where σ is the sigmoid function, $\sigma(u) = 1/(1 + \exp(-u))$. The soft (real-valued) labels can be thought of as a more informative version of the hard (0/1-valued) labels of the standard classification setting. We write $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ for the data matrix. Second, the

student is trained by minimizing the (normalised) empirical cross-entropy loss,

$$\ell^1(\mathbf{w}) \triangleq -\frac{1}{n}\sum_{i=1}^{n}\Big[y_i \log \sigma(\mathbf{w}^\intercal \mathbf{x}_i) + (1-y_i)\log(1-\sigma(\mathbf{w}^\intercal \mathbf{x}_i))\Big] - \ell^*, \tag{5.1}$$

where ℓ^* is a normalisation constant, such that the minimum of ℓ^1 is 0. It only serves the purpose of simplifying notation and has no effect on the optimisation.

The student observes the loss as a function of its parameters, i.e. the individual weight matrices,

$$\ell(\mathbf{W}_1, \ldots, \mathbf{W}_L) \triangleq \ell^1((\mathbf{W}_L \mathbf{W}_{L-1} \cdots \mathbf{W}_1)^\intercal), \tag{5.2}$$

and optimises it via gradient descent. For the theoretical analysis, we avoid the complications of discrete step sizes and assume *infinitesimal step size*, i.e. we formally assume *gradient flow* in place of gradient descent. We write $\mathbf{W}_l(\tau)$ for the value of the matrix $\mathbf{W}_l$ at time $\tau \in [0, \infty)$, with $\mathbf{W}_l(0)$ denoting the initial value, and $\mathbf{w}(\tau)^\intercal \triangleq \mathbf{W}_L(\tau) \cdots \mathbf{W}_1(\tau)$. Then, each $\mathbf{W}_l(\tau)$ for $l \in [L]$ evolves according to the following differential equation,

$$\frac{\partial \mathbf{W}_l(\tau)}{\partial \tau} = -\frac{\partial \ell}{\partial \mathbf{W}_l}(\mathbf{W}_1(\tau), \ldots, \mathbf{W}_L(\tau)). \tag{5.3}$$

The student is trained until convergence, i.e. $\tau \to \infty$. We measure the *transfer risk* of the trained student, defined as the probability that its prediction differs from that of the teacher,

$$R(h) \triangleq \mathop{\mathbb{P}}_{\mathbf{x}\sim P_{\mathbf{x}}}[h(\mathbf{x}) \neq h^*(\mathbf{x})]. \tag{5.4}$$

In the following sections, we derive a bound for the transfer risk and establish how rapidly it decreases as a function of dataset size n.

5.3 Generalisation properties of linear distillation

We now present our main technical results. First, in Section 5.3.1, we provide an explicit characterisation of the inductive bias of distillation-based training in the linear setting. In other words, we identify *what the student learns*. In particular, we prove that the student is able to perfectly identify the teacher's weight vector, if the number of training examples (n) is equal to the dimensionality of the data (d) or higher. If less data is available, under minor assumptions, the student finds the best approximation of the teacher's weight vector that is possible within the subspace spanned by the training data.

In Section 5.3.2 we use these results to study the generalisation properties of the student classifier, i.e. we characerise *how fast the student learns*. Specifically, we prove a generalisation bound with much more appealing properties than what is possible in the classic situation of learning from hard labels. As soon as enough training data is available ($n \geq d$), the student's risk is simply 0. Otherwise, the risk can be bounded explicitly in a distribution-dependent way that allows us to understand when distillation-based transfer is most effective.

5.3.1 What does the student learn?

In this section, we derive in closed form the asymptotic solution to the gradient flow (5.3) undergone by the student when trained by distillation. We state the results separately for

directly parameterised linear classifiers ($L = 1$) and deep linear networks ($L \geq 2$), as the settings require slightly different ways of initialising parameters. Namely, in the former case initialising $\mathbf{w}(0) = \mathbf{0}$ is valid, while in the latter case this would lead to vanishing gradients and we instead initialise with small random values.

Theorem 5.1. *Assume the student is a directly parameterised linear classifier* ($L = 1$) *initialised such that* $\mathbf{w}(0) = \mathbf{0}$*. Then, the student's weight vector fulfills almost surely*

$$\mathbf{w}(t) \to \hat{\mathbf{w}}, \tag{5.5}$$

for $t \to \infty$*, where*

$$\hat{\mathbf{w}} = \begin{cases} \mathbf{w}_*, & n \geq d, \\ \mathbf{X}(\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{w}_*, & n < d. \end{cases} \tag{5.6}$$

Theorem 5.1 shows a remarkable property of distillation-based training for linear classifiers: if sufficiently many (at least d) data points are available, the student exactly recovers the teacher's weight vector $\mathbf{w}_*$. This is a strong justification for distillation as a method of knowledge transfer between linear classifiers and the theorem establishes that the effect occurs not just in the infinite data limit ($n \to \infty$), as is common, but already in the finite sample regime ($n \geq d$).

When few data points are available ($n < d$), the weight vector learned by the student is simply the projection of the teacher's weight vector onto the data span (the subspace spanned by the columns of $\mathbf{X}$). This is the best the student can do in the following sense: the gradient descent update direction $\frac{\partial \mathbf{w}(\tau)}{\partial \tau}$ always lies in the data span, so there is no way for the student to learn anything outside of it. The projection is the best subspace-constrained approximation of $\mathbf{w}_*$ with respect to the Euclidean norm. The extent to which Euclidean closeness actually implies closeness in predictions is the subject of Section 5.3.2.

Proof sketch of Theorem 5.1. First, notice that $\hat{\mathbf{w}}$ is a global minimiser of ℓ^1. Moreover, when $n \geq d$, it is (almost surely wrt. $\mathbf{X} \sim P_{\mathbf{x}}^n$) unique, and when $n < d$, it is almost surely the only one lying in the span of $\mathbf{X}$ and thus potentially reachable by gradient descent.

The proof consists of two parts. We prove that a) the gradient flow (5.3) drives the objective value towards the optimum, $\ell^1(\mathbf{w}(t)) \to 0$ as $t \to \infty$, and b) the distance between $\mathbf{w}(t)$ and the claimed asymptote $\hat{\mathbf{w}}$ is upper-bounded by the objective gap,

$$\|\mathbf{w}(t) - \hat{\mathbf{w}}\|^2 \leq c\, \ell^1(\mathbf{w}(t)) \tag{5.7}$$

for some constant $c > 0$ and all $t \in [0, \infty)$.

For part a), observe that ℓ^1 is convex. For any $\tau \in [0, \infty)$, the time-derivative of $\ell^1(\mathbf{w}(\tau))$ is negative unless we are at a global minimum,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau} \ell^1(\mathbf{w}(\tau)) &= \nabla \ell^1(\mathbf{w}(\tau))^\intercal \left(\frac{\partial \mathbf{w}(\tau)}{\partial \tau} \right) \\ &= -\left\| \nabla \ell^1(\mathbf{w}(\tau)) \right\|^2, \end{aligned} \tag{5.8}$$

implying that the objective value $\ell^1(\mathbf{w}(\tau))$ decreases monotonically in τ. Hence, if we denote by $\mathcal{W} = \{\mathbf{w} : \ell^1(\mathbf{w}) \leq \ell^1(\mathbf{0})\}$ the $\ell^1(\mathbf{0})$-sublevel set of the objective, we know that

$\mathbf{w}(\tau) \in \mathcal{W}$ for all $\tau \in [0, \infty)$. One can show that on this set, ℓ^1 satisfies strong convexity, but only along certain directions: for some $\mu > 0$ and all $\mathbf{w}, \mathbf{v} \in \mathcal{W}$ such that $\mathbf{v} - \mathbf{w} \in \operatorname{span}(\mathbf{X})$,

$$\ell^1(\mathbf{v}) \geq \ell^1(\mathbf{w}) + \nabla \ell^1(\mathbf{w})^\intercal (\mathbf{v} - \mathbf{w}) + \frac{\mu}{2} \|\mathbf{v} - \mathbf{w}\|^2. \tag{5.9}$$

This allows us (via a technical derivation that we omit here) to relate the objective gap to the gradient norm: it can be shown that there exists $c' > 0$, such that

$$c' \ell^1(\mathbf{w}) \leq \frac{1}{2} \left\| \nabla \ell^1(\mathbf{w}) \right\|^2. \tag{5.10}$$

Applying the above to $\mathbf{w}(\tau)$ in (5.8), we are able to bound the amount of reduction in the objective in terms of the objective itself, ultimately proving linear convergence.

For part b), invoke (5.9) with $\mathbf{v} = \mathbf{w}(\tau)$ and $\mathbf{w} = \hat{\mathbf{w}}$; this gives $\ell^1(\mathbf{w}(\tau)) \geq \frac{\mu}{2} \|\mathbf{w}(\tau) - \hat{\mathbf{w}}\|^2$. □

The full proof is given in Appendix D.2.

The following result is the analog of Theorem 5.1 for deep linear networks. Here, some technical conditions are needed because the parameters cannot all be initialised at 0.

Theorem 5.2. *Let $\hat{\mathbf{w}}$ be defined as in eq.* (5.6)*. Assume the student is a deep linear network, initialised such that for some $\epsilon > 0$,*

$$\|\mathbf{w}(0)\| < \min \left\{ \|\hat{\mathbf{w}}\|, \epsilon^L \left(\epsilon^2 \|\hat{\mathbf{w}}\|^{-\frac{2}{L}} + \|\hat{\mathbf{w}}\|^{2 - \frac{2}{L}} \right)^{-\frac{L}{2}} \right\}, \tag{5.11}$$

$$\ell^1(\mathbf{w}(0)) < \ell^1(\mathbf{0}), \tag{5.12}$$

$$\mathbf{W}_{l+1}(0)^\intercal \mathbf{W}_{l+1}(0) = \mathbf{W}_l(0) \mathbf{W}_l(0)^\intercal \tag{5.13}$$

for $l \in [L-1]$. Then, for $n \geq d$, the student's weight vector fulfills almost surely

$$\mathbf{w}(t) \to \hat{\mathbf{w}}, \tag{5.14}$$

and for $n < d$,

$$\|\mathbf{w}(t) - \hat{\mathbf{w}}\| \leq \epsilon, \tag{5.15}$$

for all t large enough.

The interpretation of the theorem is analogous to Theorem 5.1. Given enough data ($n \geq d$), the student learns to perfectly mimic the teacher. Otherwise, it learns an approximation at least ϵ-close to the projection of the teacher's weight vector onto the data span.

The conditions (5.11)–(5.13) appear for technical reasons and a closer look at them shows that they do not pose problems in practice. Condition (5.11) states that the network's weights should be initialised with sufficiently small values. Consequently, this assumption is easy to satisfy in practice. Condition (5.12) requires that the initial loss is smaller than the loss at $\mathbf{w} = \mathbf{0}$. This condition guarantees that the gradient flow does not hit the point $\mathbf{w} = \mathbf{0}$, where the gradient vanishes and the optimisation would stop prematurely. In practice, when the step size is finite, the condition is not needed. Nevertheless, it is also not hard to satisfy: for any near-zero initialisation, $\mathbf{w}(0) = \mathbf{w}_0$, either $\mathbf{w}_0$ or $-\mathbf{w}_0$ will satisfy (5.12), so at most one has to flip the sign of one of the $\mathbf{W}_l(0)$ matrices. Finally, condition (5.13) is called *balancedness* Arora et al. [2018] and discussed in-depth in Arora et al. [2019]). It simplifies

the analysis of matrix products and makes it possible to explicitly analyze the evolution of $\mathbf{w}$ induced by gradient flow wrt. $\{\mathbf{W}_l\}$. Assuming near-zero initialisation, the condition is automatically satisfied approximately and there is some evidence Arora et al. [2019] suggesting that *approximate balancedness* may suffice for convergence results of the kind we are interested in. Otherwise, the condition can also simply be enforced numerically.

Proof sketch of Theorem 5.2. First, we establish convergence in the objective, $\ell^1(\mathbf{w}(t)) \to 0$ as $t \to \infty$, similarly to the case $L = 1$. Unlike that case, however, the evolution of the end-to-end weight vector $\mathbf{w}(\tau)$ is governed by complex mechanics induced by gradient flow wrt. $\{\mathbf{W}_l\}$. A key tool for analysing this induced flow was recently established in Arora et al. [2018]: the authors show that the induced flow behaves similarly to gradient flow with momentum applied directly to $\mathbf{w}$. Making use of this result, one can proceed analogously as in the case of $L = 1$ to show convergence in the objective.

Second, to show convergence in parameter space, we decompose $\mathbf{w}(t)$ into its projection onto the span of $\mathbf{X}$, and an orthogonal component. The $\mathbf{X}$-component converges to $\hat{\mathbf{w}}$, by strong convexity arguments as in the case $L = 1$. It remains to show that the orthogonal component is small. Now, recall that in the case $L = 1$, we initialise at $\mathbf{w}(0) = \mathbf{0}$ and move within the span, so the orthogonal component is always zero. When $L \geq 2$, the situation is different: a) we initialise with a potentially non-zero orthogonal component (because we need to avoid the spurious stationary point $\mathbf{w} = \mathbf{0}$), and b) the momentum term causes the orthogonal component to grow during optimisation. Luckily, the rate of growth can be precisely characterised and controlled by the initialisation norm $\|\mathbf{w}(0)\|$, so depending on how close to zero we initialise, we can upper-bound the size of the orthogonal component. This yields a bound on the distance $\|\mathbf{w}(t) - \hat{\mathbf{w}}\|$. □

For the complete proof, we refer the reader to Appendix D.3.

5.3.2 How fast does the student learn?

In this section, we present the main quantitative result of this chapter, a bound for the expected transfer risk in linear distillation.

We first introduce some geometric concepts. For any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$, denote by $\bar{\alpha}(\mathbf{u}, \mathbf{v}) \in [0, \pi/2]$ the unsigned angle between the vectors $\mathbf{u}$ and $\mathbf{v}$

$$\bar{\alpha}(\mathbf{u}, \mathbf{v}) = \cos^{-1}\left(\frac{|\mathbf{u}^\intercal \mathbf{v}|}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}\right). \tag{5.16}$$

A key quantity for us is the angle between $\mathbf{w}_*$ and a randomly sampled $\mathbf{x}$ for $\mathbf{x} \sim P_{\mathbf{x}}$. For a given transfer task $(P_{\mathbf{x}}, \mathbf{w}_*)$, we denote by p the reverse cdf of $\bar{\alpha}(\mathbf{w}_*, \mathbf{x})$,

$$p(\theta) = \mathop{\mathbb{P}}_{\mathbf{x} \sim P_{\mathbf{x}}}[\bar{\alpha}(\mathbf{w}_*, \mathbf{x}) \geq \theta] \qquad \text{for} \quad \theta \in [0, \pi/2]. \tag{5.17}$$

By construction, $p(\theta)$ is monotonically decreasing, starting at $p(0) = 1$ and approaching 0 as $\theta \to \pi/2$. Figure 5.1 illustrates this behaviour for three illustrative data distributions denoted *Task A, Task B and Task C*. In *Task A*, the probability mass is well aligned with the direction of the teacher's weight vector. The probability that a randomly chosen data point $\mathbf{x} \sim P_{\mathbf{x}}$ has a large angle with $\mathbf{w}_*$ is small. Therefore, the value of $p(\theta)$ quickly drops with growing angle θ. In *Task B*, the data also aligns well with $\mathbf{w}_*$, but in addition, the data support remains

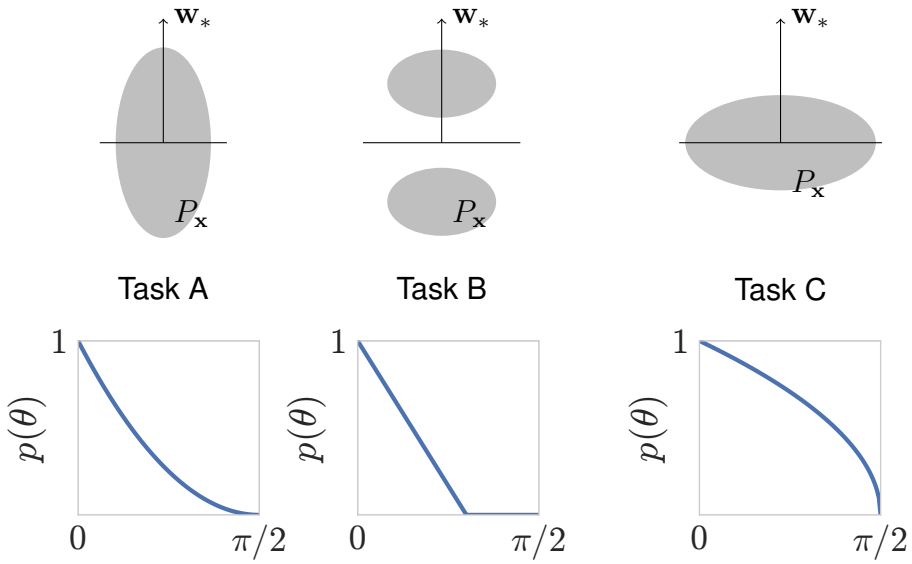


Figure 5.1: Schematic illustration of $p(\theta)$ for three different transfer tasks. In Task A, the angular alignment between the data and the teacher's weight vector is high, so $p(\theta)$ decreases quickly. In Task B, it is also high, and in addition the classes are separated by a margin, so $p(\theta)$ reaches 0 before reaching $\theta = \pi/2$. In Task C, the angular alignment is low, so $p(\theta)$ decreases rather slowly.

bounded away from the decision boundary. Therefore, certain large angles can never occur, i.e. there exists a value $\theta_0 < \pi/2$, such that $p(\theta) = 0$ for $\theta \geq \theta_0$. In *Task C*, the situation is different: the data distribution is concentrated along the decision boundary and the angle between $\mathbf{w}_*$ and a randomly chosen data point $\mathbf{x} \sim P_\mathbf{x}$ is large. As a consequence, $p(\theta)$ drops more slowly with the angle θ compared to the previous two settings.

We are now ready to state the main result. For improved readability, we phrase it for a student with infinitesimally small initialization, i.e. $\epsilon \to 0$. The general formulation can be found in Appendix D.4.

Theorem 5.3 (Transfer risk bound for linear distillation)**.** *For any training set $\mathbf{X} \in \mathbb{R}^{d \times n}$, let $\hat{h}_\mathbf{X}(\mathbf{x}) = \mathbb{1}\{\hat{\mathbf{w}}^\intercal \mathbf{x} \geq 0\}$ be the linear classifier learned by distillation from a teacher with weight vector $\mathbf{w}_*$. Then, when $n \geq d$, it holds that*

$$\mathop{\mathbb{E}}_{\mathbf{X} \sim P_\mathbf{x}^{\otimes n}} \left[R\big(\hat{h}_\mathbf{X}\big) \right] = 0. \tag{5.18}$$

For $n < d$, it holds for any $\beta \in [0, \pi/2]$ that

$$\mathop{\mathbb{E}}_{\mathbf{X} \sim P_\mathbf{x}^{\otimes n}} \left[R\big(\hat{h}_\mathbf{X}\big) \right] \leq p(\beta) + p(\pi/2 - \beta)^n \tag{5.19}$$

Equation (5.18) is unsurprising, since in Section 5.3.1 we already established that for $n \geq d$ the student is able to perfectly mimic the teacher. However, the inequality (5.19) is, to our knowledge, the first quantitative characterisation of how well a student can learn via distillation.

Before we provide its proof sketch, we present two instantiations of the bound for specific classes of tasks. Hopefully this will provide some insight into how fast the right hand side of (5.19) actually decreases.

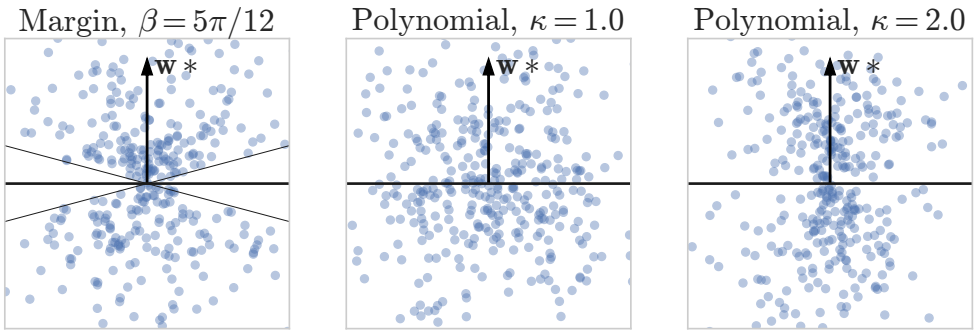


Figure 5.2: Examples of 2D distributions that fulfill the large-margin condition (left) and the polynomial condition with different parameters (center, right).

The large-margin case. The first class of tasks we consider are tasks in which the classes are separated by an *angular margin*, illustrated in Figure 5.2 (left). These tasks are characterized by a 'wedge' of zero probability mass near the boundary[1]. For these tasks, we obtain from Theorem 5.3 that the expected risk decays exponentially in n, up to $n = d - 1$.

Corollary 5.1 (Transfer risk of large-margin distributions)**.** *If there exists* $\beta \in [0, \pi/2]$ *such that* $p(\beta) = 0$ *and* $\gamma := p(\pi/2 - \beta) < 1$*, then*

$$\mathop{\mathbb{E}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} \left[R\big(\hat{h}_{\mathbf{X}}\big) \right] \leq \gamma^n. \tag{5.20}$$

The polynomial case. The second class are tasks for which we can upper-bound p by a κ-order polynomial; see Corollary 5.2 below. We can do so trivially for any task by setting $\kappa = 0.0$, but that choice would yield a vacuous bound. Higher values of κ correspond to stronger assumptions on the distribution but enable better rates. Figure 5.2 (center, right) shows examples of polynomial distributions for $\kappa \in \{1.0, 2.0\}$. The special case $\kappa = 1.0$ corresponds to a uniform angle distribution, while a distribution with $\kappa = 2.0$ has low probability mass near the decision boundary, while not necessarily exhibiting a margin.

Corollary 5.2 below establishes that for tasks with κ-polynomial behavior of $p(\theta)$, the expected risk decays essentially at a rate of $(\log n/n)^\kappa$ or faster.

Corollary 5.2 (Transfer risk of polynomial distributions)**.** *If there exists a* $\kappa \geq 0$ *such that* $p(\theta) \leq (1 - (2/\pi)\theta)^\kappa$ *for all* $\theta \in [0, \pi/2]$*, then*

$$\mathop{\mathbb{E}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} \left[R\big(\hat{h}_{\mathbf{X}}\big) \right] \leq \frac{1 + (\log n)^\kappa}{n^\kappa} \tag{5.21}$$

Proof. We apply Theorem 5.3 and insert the polynomial upper bound for p. For the case $n < d$, we obtain that for any $\beta \in [0, \pi/2]$,

$$\mathop{\mathbb{E}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} \left[R\big(\hat{h}_{\mathbf{X}}\big) \right] \leq (1 - (2/\pi)\beta)^\kappa + (1 - (2/\pi)(\pi/2 - \beta))^{n\kappa}. \tag{5.22}$$

Setting $\beta = (\pi/2) \cdot n^{-1/n}$ and simplifying the resulting expressions yields

$$\leq \left(1 - e^{-\frac{\log n}{n}}\right)^\kappa + n^{-\kappa}. \tag{5.23}$$

Finally, we use the inequality $e^u \geq 1 + u$ and the claim follows. □

[1] In bounded domains, this condition is for example fulfilled in the *classical margin* situation [Schölkopf and Smola, 2002], when the classes are separated by a positive distance from each other.

Note the contrast to classical learning-theoretic bounds: standard bounds are usually vacuous when $n < d$ and only start to be useful when $n \gg d$. In contrast, linear distillation provably achieves perfect transfer as soon as $n \geq d$ and non-trivial performance even in the low-data regime $n < d$. The latter can best be seen in Corollary 5.1, where $\gamma < 1$ and hence we obtain a meaningful risk guarantee of γ^n.

5.3.3 Proof of Theorem 5.3

The case $n \geq d$ follows directly from Theorems 5.1 and 5.2. For the remaining case, $n < d$, the following property turns out to be important for obtaining a transfer risk rate of the form that we do.

Lemma 5.1 (Strong monotonicity)**.** *Let $\hat{\mathbf{w}}(\mathbf{X})$ denote the distillation solution $\hat{\mathbf{w}}$ as a function of the training data $\mathbf{X}$. Then, for any full-rank datasets $\mathbf{X}_- \in \mathbb{R}^{d\times n_-}$ and $\mathbf{X}_+ \in \mathbb{R}^{d\times n_+}$ such that $\mathbf{X}_-$ is contained in $\mathbf{X}_+$,*

$$\bar{\alpha}(\mathbf{w}_*, \hat{\mathbf{w}}(\mathbf{X}_+)) \leq \bar{\alpha}(\mathbf{w}_*, \hat{\mathbf{w}}(\mathbf{X}_-)). \tag{5.24}$$

Proof. If $n_+ \geq d$, then the left-hand side of (5.24) is zero and the claim follows. Otherwise, assume wlog that the first n_- columns of $\mathbf{X}_-$ and $\mathbf{X}_+$ coincide. Let $\mathbf{Q}_+\mathbf{R}_+ = \mathbf{X}_+$ be the QR factorisation of $\mathbf{X}_+$ with $\mathbf{Q}_+ \in \mathbb{R}^{d\times n_+}$ and $\mathbf{R}_+ \in \mathbb{R}^{n_+\times n_+}$, and similarly for $\mathbf{X}_-$. Then $\hat{\mathbf{w}}(\mathbf{X}_+) = \mathbf{Q}_+\mathbf{Q}_+^\intercal\mathbf{w}_*$ and

$$\cos(\bar{\alpha}(\mathbf{w}_*, \hat{\mathbf{w}}(\mathbf{X}_+))) = \frac{\mathbf{w}_*^\intercal\mathbf{Q}_+\mathbf{Q}_+^\intercal\mathbf{w}_*}{\|\mathbf{w}_*\| \cdot \|\mathbf{Q}_+\mathbf{Q}_+^\intercal\mathbf{w}_*\|} \tag{5.25}$$

$$= \frac{\|\mathbf{Q}_+^\intercal\mathbf{w}_*\|}{\|\mathbf{w}_*\|}, \tag{5.26}$$

and an analogous statement holds for $\mathbf{X}_-$. Now, because the first n_- columns of $\mathbf{Q}_+$ coincide with $\mathbf{Q}_-$, we have $\|\mathbf{Q}_+^\intercal\mathbf{w}_*\| \geq \|\mathbf{Q}_-^\intercal\mathbf{w}_*\|$ and

$$\cos(\bar{\alpha}(\mathbf{w}_*, \hat{\mathbf{w}}(\mathbf{X}_+))) \geq \cos(\bar{\alpha}(\mathbf{w}_*, \hat{\mathbf{w}}(\mathbf{X}_-))). \tag{5.27}$$

Taking $\cos^{-1}$ on both sides (and remembering that $\cos^{-1}$ is decreasing) yields the claim. □

To interpret the theorem, think of $\bar{\alpha}(\mathbf{w}_*, \hat{\mathbf{w}})$ as a proxy for the transfer risk, i.e. the closer the trained student $\hat{\mathbf{w}}$ is to the teacher $\mathbf{w}_*$ in terms of angles, the lower the transfer risk. A direct consequence of Lemma 5.1, and the reason it is called 'strong mononoticity', is that including additional data in the transfer set can never harm the transfer risk, only improve it. This property is specific to distillation; it does not hold in hard-target learning.

Proof of Theorem 5.3 ($n < d$). For nonzero vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, we define $\alpha(\mathbf{u}, \mathbf{v}) \in [0, \pi]$ as a variant of $\bar{\alpha}$ (Equation 5.16) that takes the sign of $\mathbf{u}^\intercal\mathbf{v}$ into account,

$$\alpha(\mathbf{u}, \mathbf{v}) = \cos^{-1}\left(\frac{\mathbf{u}^\intercal\mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}\right). \tag{5.28}$$

We decompose the expected risk as follows:

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathbf{X}\sim P_{\mathbf{x}}^n}\Big[R\big(\hat{h}_{\mathbf{X}}\big)\Big] &= \mathop{\mathbb{P}}_{\substack{\mathbf{X}\sim P_{\mathbf{x}}^n\\ \mathbf{x}\sim P_{\mathbf{x}}}}[\mathbf{w}_*^\intercal\mathbf{x}\cdot\hat{\mathbf{w}}^\intercal\mathbf{x}<0]\\ &= \int_{\mathbf{x}:\bar{\alpha}(\mathbf{w}_*,\mathbf{x})\geq\beta}\mathop{\mathbb{P}}_{\mathbf{X}\sim P_{\mathbf{x}}^n}[\mathbf{w}_*^\intercal\mathbf{x}\cdot\hat{\mathbf{w}}^\intercal\mathbf{x}<0|\mathbf{x}]\,\mathrm{d}P_{\mathbf{x}}\\ &+ \int_{\mathbf{x}:\bar{\alpha}(\mathbf{w}_*,\mathbf{x})<\beta,\,\mathbf{w}_*^\intercal\mathbf{x}>0}\mathop{\mathbb{P}}_{\mathbf{X}\sim P_{\mathbf{x}}^n}[\hat{\mathbf{w}}^\intercal\mathbf{x}<0|\mathbf{x}]\,\mathrm{d}P_{\mathbf{x}}\\ &+ \int_{\mathbf{x}:\bar{\alpha}(\mathbf{w}_*,\mathbf{x})<\beta,\,\mathbf{w}_*^\intercal\mathbf{x}<0}\mathop{\mathbb{P}}_{\mathbf{X}\sim P_{\mathbf{x}}^n}[\hat{\mathbf{w}}^\intercal\mathbf{x}>0|\mathbf{x}]\,\mathrm{d}P_{\mathbf{x}}.\end{aligned} \tag{5.29}$$

Let us fix some $\mathbf{x}$ for which $\bar{\alpha}(\mathbf{w}_*,\mathbf{x})<\beta$ and $\mathbf{w}_*^\intercal\mathbf{x}>0$ (i.e. an 'easy' positive test example); for this $\mathbf{x}$ we have $\alpha(\mathbf{w}_*,\mathbf{x})=\bar{\alpha}(\mathbf{w}_*,\mathbf{x})$. Consider the situation where $\bar{\alpha}(\mathbf{w}_*,\mathbf{x}_i)<\pi/2-\beta$ for some i, i.e. there is at least one good teaching point. Then, Lemma 5.1 with $\mathbf{X}_+=\mathbf{X}$ and $\mathbf{X}_-=\mathbf{x}_i$ yields $\bar{\alpha}(\mathbf{w}_*,\hat{\mathbf{w}})\leq\bar{\alpha}(\mathbf{w}_*,\mathbf{x}_i)<\pi/2-\beta$. Combined with the triangle inequality, we obtain

$$\alpha(\hat{\mathbf{w}},\mathbf{x}) \leq \alpha(\mathbf{w}_*,\hat{\mathbf{w}})+\alpha(\mathbf{w}_*,\mathbf{x}) \tag{5.30}$$

$$\leq \bar{\alpha}(\mathbf{w}_*,\mathbf{x}_i)+\bar{\alpha}(\mathbf{w}_*,\mathbf{x})<\pi/2, \tag{5.31}$$

which implies $\hat{\mathbf{w}}^\intercal\mathbf{x}>0$, i.e. a correct prediction (same as the teacher's). Conversely, an error can occur only if $\bar{\alpha}(\mathbf{w}_*,\mathbf{x}_i)\geq\pi/2-\beta$ for all i. Because $\mathbf{x}_i$ are independent, we have

$$\begin{aligned}\mathop{\mathbb{P}}_{\mathbf{X}\sim P_{\mathbf{x}}^n}[\hat{\mathbf{w}}^\intercal\mathbf{x}<0|\,\mathbf{x}:\bar{\alpha}(\mathbf{w}_*,\mathbf{x})<\beta,\,\mathbf{w}_*^\intercal\mathbf{x}>0] &\leq \mathop{\mathbb{P}}_{\mathbf{X}\sim P_{\mathbf{x}}^n}[\forall_i:\bar{\alpha}(\mathbf{w}_*,\mathbf{x}_i)\geq\pi/2-\beta]\\ &= p(\pi/2-\beta)^n.\end{aligned} \tag{5.32}$$

An analogous statement can be shown for negative examples:

$$\mathop{\mathbb{P}}_{\mathbf{X}\sim P_{\mathbf{x}}^n}[\hat{\mathbf{w}}^\intercal\mathbf{x}>0|\,\mathbf{x}:\bar{\alpha}(\mathbf{w}_*,\mathbf{x})<\beta,\,\mathbf{w}_*^\intercal\mathbf{x}<0]\leq p(\pi/2-\beta)^n. \tag{5.33}$$

Combining (5.29), (5.32) and (5.33) yields the result:

$$\begin{aligned}\mathop{\mathbb{P}}_{\substack{\mathbf{X}\sim P_{\mathbf{x}}^n\\ \mathbf{x}\sim P_{\mathbf{x}}}}[\mathbf{w}_*^\intercal\mathbf{x}\cdot\hat{\mathbf{w}}^\intercal\mathbf{x}<0] &\leq \mathop{\mathbb{P}}_{\mathbf{x}}[\bar{\alpha}(\mathbf{w}_*,\mathbf{x})\geq\beta]+\mathop{\mathbb{P}}_{\mathbf{x}}[\bar{\alpha}(\mathbf{w}_*,\mathbf{x})<\beta]\cdot p(\pi/2-\beta)^n\\ &= p(\beta)+(1-p(\beta))\cdot p(\pi/2-\beta)^n.\end{aligned}$$

□

5.4 Discussion

In this section, we discuss some concepts that emerged in the formal analysis in previous sections: *data geometry, optimisation bias, and strong monotonicity*. We then provide empirical confirmation of their effect on the transfer risk.

5.4.1 Data geometry

From Theorem 5.3 we know that the data geometry, in particular the *angular alignment* between the data distribution and the teacher, impact how fast the student can learn. Formally, this is reflected in $p(\theta)$: the faster it decreases, the easier it should be for the student to learn the task.

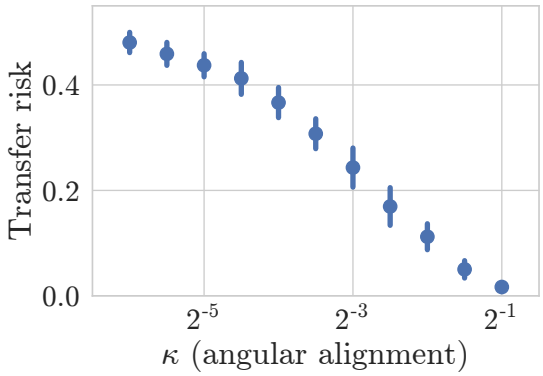


Figure 5.3: Transfer risk of linear distillation on tasks with varying angular alignment.

To experimentally test the effect of data geometry on the effectiveness of distillation, we adopt the setting of Corollary 5.2. We consider a series of tasks with varying angular alignment, as measured by the degree κ of the polynomial by which $p(\theta)$ is upper bounded.

Specifically, for any κ, the task $(P_{\mathbf{x}}^{\kappa}, \mathbf{w}_*^{\kappa})$ is defined by the following sampling procedure. First, an angle a is sampled from the κ-polynomial distribution, i.e. $\mathbb{P}[a \geq \theta] = (1-(2/\pi)\theta)^{\kappa}$ for $\theta \in [0, \pi/2]$. Then, a direction $\mathbf{x}$ is uniformly sampled from all unit-length vectors that are at angle a with the teacher's weight vector, $\bar{\alpha}(\mathbf{w}_*, \mathbf{x}) = a$. Finally, $\mathbf{x} = \nu\mathbf{x}$ is returned for a random ν, distributed as a one-dimensional standard Gaussian.

We use an input space dimension of $d = 1000$ and a transfer set size $n = 20$. Then, we train a linear student by distillation on each of the tasks and evaluate its transfer risk on held-out data. Figure 5.3 shows the results. The plot shows a clearly decreasing trend: on tasks with more favorable data geometry (higher κ), transfer via distillation is more effective and the student achieves lower risk.

5.4.2 Optimisation bias

Another important factor for the success of distillation is a specific inductive bias. For $n < d$, the distillation training objective (5.1) has many minima with identical function value but potentially different generalisation properties. Therefore, the optimisation method used could have a large impact on the transfer risk. As Theorems 5.1 and 5.2 show, gradient descent has a particularly favourable inductive bias for distillation.

To verify this observation experimentally, we consider learners that are guided by an optimisation bias to different degrees: at one end of the spectrum is the gradient-descent learner we have studied in previous sections, while at the other end is a learner that treats all minimizers of the distillation training loss equally, i.e. that has no bias toward any of the solutions. Specifically, consider learners with weights of the form $\mathbf{w}_\delta = \hat{\mathbf{w}} + \delta \frac{\|\hat{\mathbf{w}}\|}{\|\mathbf{q}\|}\mathbf{q}$, where $\hat{\mathbf{w}}$ is the gradient-descent distillation solution and $\mathbf{q}$ is a Gaussian random vector in the subspace orthogonal to the data span, i.e. if $\mathbf{X}$ is the data matrix, then $\mathbf{X}^\intercal\mathbf{q} = \mathbf{0}$. All learners of this form globally minimize the distillation training loss, and depending on δ, they are more or less guided by the gradient-descent bias: $\delta = 0$ and $|\delta| \to \infty$ represent the two extremes mentioned above.

We train the learners $\mathbf{w}_\delta$ for $\delta \in \{0, 10, \ldots, 90\}$ on the digits 0 and 1 of the MNIST dataset, where inputs are treated as vectors in $\mathbb{R}^{784}$ and the teacher $\mathbf{w}_*$ is a logistic regression

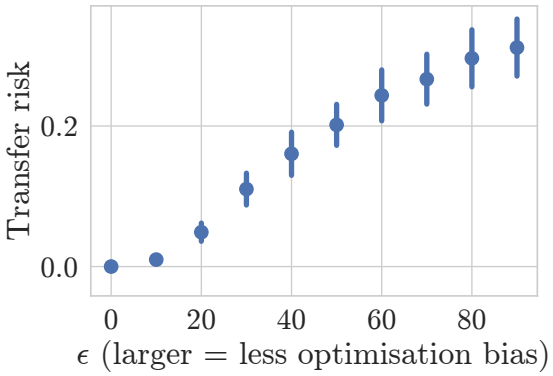


Figure 5.4: Transfer risk of linear distillation variants with different degrees of optimisation bias, on the digits 0 and 1 of MNIST.

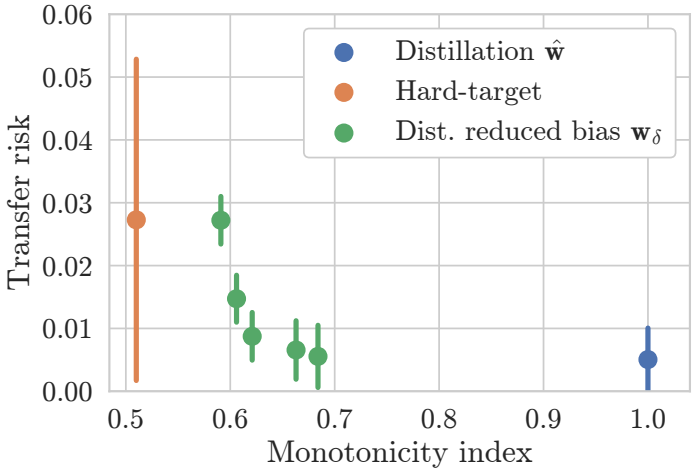


Figure 5.5: Expected transfer risk vs. monotonicity of different learners: gradient-descent based distillation (blue), hard-target learner (orange), and a series of distillation learners with reduced optimisation bias (green): $\mathbf{w}_\delta$ for $\delta \in \{1/16, 1/8, 1/4, 1/2, 1\}$, listed in order from left to right.

trained to classify 0s and 1s on an independent training set. We set the transfer set size to $n = 100$ and evaluate the risk on the test set.

Figure 5.4 shows the result. There is a clear trend in favour of learners that are more strongly guided by the gradient-descent bias (small δ); these learners generally achieve lower transfer risk. This result supports the idea of optimisation bias as a key component of distillation's success.

5.4.3 Strong monotonicity

We now investigate *strong monotonicity*, which was identified in Lemma 5.1: training the student on more data always leads to a better approximation of the teacher's weight vector.

Compared to data geometry and optimisation bias, strong monotonicity is less amenable to experimental study because it is a downstream property that cannot directly be manipulated. We therefore take an indirect approach. We consider a set of learners including the gradient-descent distillation learner, the hard-target learner, and several learners with reduced optimisation bias (like in Section 5.4.2), and train them on the same task. For each learner, we note its expected risk calculated on a held-out set, and its *monotonicity index*, defined as

the probability that an additional training example reduces the angle between the student's and the teacher's weight vectors rather than increasing it, i.e.

$$m(\mathbf{w}) = \mathop{\mathbb{P}}_{\substack{\mathbf{X}\sim P_{\mathbf{x}}^n \\ \mathbf{x}\sim P_{\mathbf{x}}}} [\bar{\alpha}(\mathbf{w}_*, \mathbf{w}([\mathbf{X}, \mathbf{x}])) < \bar{\alpha}(\mathbf{w}_*, \mathbf{w}(\mathbf{X}))], \tag{5.34}$$

where the student's weight vector $\mathbf{w}$ is now treated as a function of the training set. Thus, we can relate a learner's risk and its monotonicity.

We train the learners on the polynomial-angle task $(P_{\mathbf{x}}^{\kappa}, \mathbf{w}_*^{\kappa})$ from Section 5.4.1, with $\kappa = 1, d = 100$ and $n = 5$. The expected risk as well as the monotonicity index are estimated as averages over 1000 transfer sets.

The results are shown in Figure 5.5. There is a negative correlation between monotonicity and transfer risk, which supports the intuition of monotonicity as a desirable property.

However, a few reservations are in order. First, as mentioned above, monotonicity cannot easily be manipulated, so its *effect* on transfer risk remains unknown. We can only measure correlation. Second, monotonicity is of binary nature; it only captures whether an extra data point helps or not. Yet for a quantitative characterisation of risk, one would have to capture by how much an extra data point helps. We leave more refined definitions of monotonicity for future work.

5.5 Conclusion

In this chapter, we have formulated and studied a linear model of knowledge distillation. Within this model, we have derived a characterisation of the solution learned by the student, and a bound on the transfer risk, meaningful even in the low-data regime. In doing so, we hope to have enriched the current intuitive and theoretical understanding of distillation and of soft-label supervision.

Our work paints a picture of distillation as an extremely effective method for knowledge transfer that derives its power from an optimisation bias of gradient-based methods initialised near the origin, which in particular has the effect that any additionally included training point can only improve the student's approximation of the teacher. Distillation further benefits strongly from a favorable data geometry, in particular a margin between classes.

While we have supported this picture by theory and empirical work only in the linear case, we hypothesise that similar properties also govern the behaviour of distillation in the nonlinear setting. We consider the extension to nonlinear models the main direction for future work.

CHAPTER 6

Distillation-based training for multi-exit architectures

In this chapter, we explore an application of knowledge distillation to training *multi-exit architectures*, a class of neural network architectures suitable for time-efficient inference.

Efficient inference is important in many practical problems. For example, on mobile devices, execution speed directly influences battery life and heat release. For robotics applications, such as self-driving cars, low latency is crucial for operating under real-time constraints. Some of these settings in addition require *adaptivity at test time*: the required inference speed may be uncertain at the time of training or may vary over time, e.g. due to concurrently running jobs or a dynamic change in processor speed. Using a fixed model architecture in such a setting would be suboptimal: a model that is fast enough to run under all conditions yields suboptimal accuracy in situations where the available computational budget is higher than worst-case; on the other hand, a more accurate but slower model might fail to provide decisions at prediction time when the available budget falls below what the network needs to finish its computation.

Multi-exit models are *anytime predictors*: they can trade off accuracy and computation at test time with a single model and on a per-example basis. A typical multi-exit model quickly produces a crude initial prediction and then gradually improves it. At any time, a valid prediction for the given input is available to be used in case the time budget for the classification process runs out. Hence, multi-exit models are more robust and flexible under uncertain or changing conditions and thereby overall more resource-efficient.

They typically consist of a stack of processing (e.g. convolutional) layers interleaved with early output layers (*exits*) at different depths. These are standard classification layers acting on the feature representation that the network had computed up to this stage. The exits form a sequence of increasingly complex classifiers (see Figure 6.1, “Prediction” box), in which later layers reuse the representations, and thereby computations, of the earlier layers. To make a prediction, an input image is propagated through the stack of layers (left to right in Figure 6.1). When the process is interrupted, the model outputs either one of the already evaluated outputs, or an average of all of them.

Multi-exit architectures are typically trained with a ‘multi-task’ objective: one attaches a loss function to each exit, for example cross-entropy, and minimizes the sum of exit-wise losses, as if each exit formed a separate classification task. On the one hand, this is a canonical

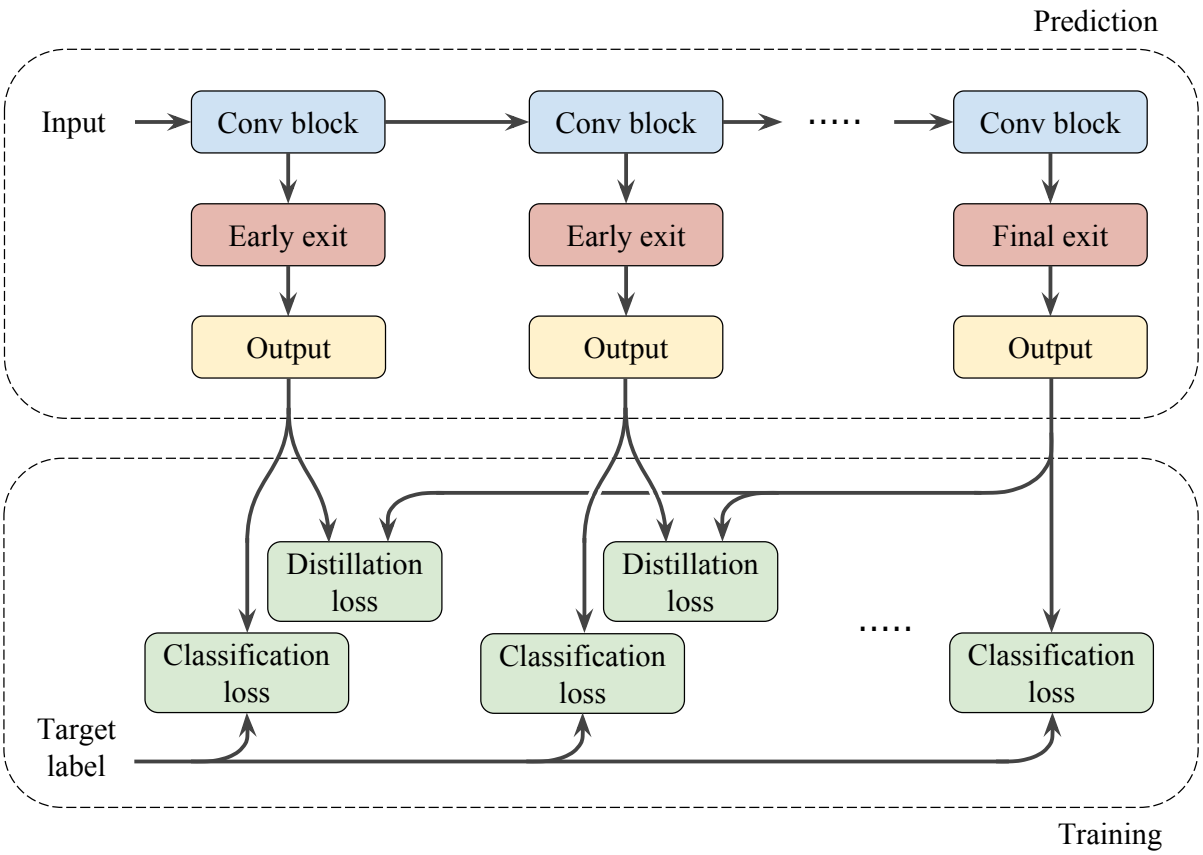


Figure 6.1: Illustration of the proposed method: distillation-based training (bottom) for a multi-exit architecture (top).

choice: not knowing which exits will be used at prediction time, we want all of them to perform well, so we should train all of them for best classification quality. On the other hand, this choice ignores a lot of the prior knowledge we have about anytime learning problems in general, and about the structure of the multi-exit architecture in particular. For example, multi-task learning is known to work best if all classifiers are of comparable complexity and quality, such that none of the loss term dominates the others. In contrast, in the multi-exit case we know a priori that the classifiers from later exits have more capacity and should be more accurate than the ones from early exits.

In this chapter, we propose a new objective for training multi-exit architectures based on knowledge distillation. The resulting *distillation-based training* method

- leads to substantially improved classification accuracy, especially for early exits,
- requires no change to the underlying architecture,
- opens up a natural way to make use also of unlabeled data when training multi-exit architectures in a semi-supervised scenario.

The main idea behind the method is sharing of information between exits. Specifically, we transfer information from late to early exits by encouraging early exits to mimic the probabilistic outputs of late exits. In practice, this is achieved by minimising the cross-

entropy between the outputs, with an additional temperature-scaling step that we detail in Section 6.2.

Our experiments on CIFAR100 and ImageNet show that distillation-based training significantly improves the accuracy of early exits while maintaining state-of-the-art accuracy for late ones. The method is particularly beneficial when training data is limited and it allows a straightforward extension to semi-supervised learning. Moreover, it takes only a few lines to implement and incurs almost no computational overhead at training time, and none at all at test time.

The rest of this chapter is devoted to describing the method in detail and to its experimental validation.

6.1 Related work

Anytime prediction. The roots of anytime computation go back to the work of Dean and Boddy [1988], Horvitz [1987]. In [Dean and Boddy, 1988], *anytime algorithms* are defined for the first time, and they become widely popular in planning and control [Drummond and Bresina, 1990, Likhachev et al., 2004, Zilberstein, 1996].

In the context of statistical learning, anytime classifiers were preceded by *cascades* [Šochman and Matas, 2005, Viola and Jones, 2001, 2004]. These are models with variable, instance-dependent runtime; however, they cannot be stopped exogenously. Early examples of truly anytime classifiers were based on streaming nearest neighbors [Ueno et al., 2006] or on classifier ensembles such as decision trees [Esmeir and Markovitch, 2010], random forests [Fröhlich et al., 2012], or boosting [Grubb and Bagnell, 2012]. A parallel line of work aimed at developing techniques for adapting an arbitrary ensemble to the anytime setting in a learner-agnostic way [Benbouzid et al., 2012, Gao and Koller, 2011, Trapeznikov and Saligrama, 2013]. These methods usually execute individual classifiers in a dynamically determined, input-dependent order.

More recently, in the context of convolutional networks, two broad approaches to anytime prediction have gained prominence: a) networks whose parts are selectively executed or skipped at test time [Larsson et al., 2017, Lin et al., 2017, Wang et al., 2018, Wu et al., 2018], or b) networks with additional exits [Huang et al., 2018, Kim et al., 2018, Teerapittayanon et al., 2016], from which an appropriate one is chosen at test time. We only discuss multi-exit architectures in detail, as this is the class of models to which our proposed training technique applies.

Multi-exit architectures. The first work to propose attaching early exits to a deep network was [Teerapittayanon et al., 2016], where standard image classification architectures such as LeNet, AlexNet and ResNet, were augmented by early exits. Huang et al. [2018] were the first to propose a custom multi-exit architecture, the Multi-Scale DenseNet (MSDNet), motivated by the observation that early exits interfere with the role of early layers as feature extractors for later use. The MSDNet was the state-of-the-art multi-exit architecture until very recently and it is the one we use in our experiments. Kim et al. [2018] propose a doubly nested architecture suitable for memory-constrained settings. Finally, there is recent work on discovering multi-exit architectures by neural architecture search (NAS) [Zhang et al., 2019].

While the main contributions of these works are architectural, our focus is on training. In all of these works except [Zhang et al., 2019] (which employs NAS-specific training), multi-exit networks are trained by minimising the sum of exit-wise losses. We propose a novel training procedure that is applicable to any of these multi-exit architectures.

Orthogonally to the subject of this work, networks with early exits have also been proposed for other purposes, such as providing stronger gradient signals for training Lee et al. [2015], or multi-resolution image processing Xie and Tu [2015].

Distillation. We review the foundational literature as well as applications of distillation in Section 5.1. In the vast majority of applications, including the work cited above, the teacher network is fixed.

Notable exceptions are [Anil et al., 2018], where distillation is used in a distributed optimisation setting to ensure consistency between different copies of a model, [Radosavovic et al., 2018], where a model is self-trained by distillation from its previous version's predictions, and [Li and Hoiem, 2016, Rebuffi et al., 2017], where distillation between stored predictions from the past and current predictions is used as a regulariser during incremental training. In contrast, we propose distillation from one part of a model to another part. To our knowledge, no previous work has addressed this setting.

6.2 Distillation-based training for multi-exit architectures

In this section, we introduce the mathematical notation and the necessary background for the discussion of the proposed method. Throughout this chapter, we consider the setting of multi-class classification with an input set $\mathcal{X}$, e.g. images, and an output set $\mathcal{Y} = \{1, \dots, K\}$, where K is the number of classes.

6.2.1 Multi-exit architectures

Multi-exit architectures are layered classification architectures with exits at different depths, see Figure 6.1 for an illustration. For a system with M exits, this results in a sequence $(\mathbf{p}_1, \dots, \mathbf{p}_M)$ of probabilistic classifiers $\mathbf{p}_m : \mathcal{X} \to \Delta^K$, each of which maps the input to the probability simplex Δ^K, i.e. the set of probability distributions over the K classes. We think of $\mathbf{p}_1, \dots, \mathbf{p}_M$ as being ordered from the least to the most expressive as well as computationally expensive. In principle, the classifiers may or may not share weights and computation, but in the most interesting and practically useful case, they do share both.

6.2.2 Prediction

At prediction time, the multi-exit system can operate in two different modes, depending on whether the computational budget to classify an example is known or not.

Budget-mode. If the computational budget for a given example is known, the system can directly identify a suitable exit, $\mathbf{p}_{M'}(\mathbf{x})$, to evaluate. This way, it only has to evaluate the shared parts of the architecture while skipping over the earlier exits ($m < M'$). How exactly the specific exit is chosen is model-dependent. In this paper, we first determine the runtime and quality of any single exit on a validation set. Then, for any target runtime, we output the decision of the exit with highest validation accuracy that runs within the available budget.

Anytime-mode. If the computational budget is unknown, i.e. for anytime prediction, after receiving a test input $\mathbf{x}$, the system starts evaluating the classifiers $\mathbf{p}_1, \mathbf{p}_2, \ldots$ in turn, reusing computation where possible. It continues doing this until it receives a signal to stop – say this happens after the M'-th exit – at which point it returns the predictions of the ensemble created from the evaluated exits, $\frac{1}{M'}\sum_{m=1}^{M'} \mathbf{p}_m(\mathbf{x})$.

6.2.3 Distillation training objective

Our main contribution is a new training objective for multi-exit architectures. Given a training set, $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, we propose to train the multi-exit architecture by a combination of a *classification loss*, ℓ_{cls} and a *distillation loss*, ℓ_{dist},

$$\frac{1}{N}\sum_{n=1}^{N}\Big[\ell_{\text{cls}}(\mathbf{x}_n, y_n) + \ell_{\text{dist}}(\mathbf{x}_n)\Big]. \tag{6.1}$$

Distillation loss. To introduce the distillation loss ℓ_{dist}, we first recall the multi-class knowledge distillation framework as introduced by Hinton et al. [2014]: assume we want a probabilistic multi-class classifier $\mathbf{s}$ (the student) to learn from another such classifier $\mathbf{t}$ (the teacher). This can be achieved by making the student output class distributions similar to those of the teacher. Specifically, for a given training example $\mathbf{x}$, we minimise the cross-entropy between the (temperature-adjusted) output distributions,

$$\ell^\tau(\mathbf{t}(\mathbf{x}), \mathbf{s}(\mathbf{x})) = -\tau^2 \sum_{k=1}^{K} [\mathbf{t}^{1/\tau}(\mathbf{x})]_k \log[\mathbf{s}^{1/\tau}(\mathbf{x})]_k, \tag{6.2}$$

with respect to the parameters of $\mathbf{s}$, where $\tau > 0$ is a scalar hyper-parameter called the *temperature*, and

$$[\mathbf{s}^{1/\tau}(\mathbf{x})]_k = \frac{s_k(\mathbf{x})^{1/\tau}}{\sum_{l=1}^{K} s_l(\mathbf{x})^{1/\tau}} \tag{6.3}$$

is the distribution obtained from the distribution $\mathbf{s}(\mathbf{x})$ by *temperature scaling* (similarly for $[\mathbf{t}^{1/\tau}(\mathbf{x})]_k$). Note that for typical network architectures, for which the outputs are the result of a softmax operation over logits, temperature scaling can be done efficiently by simply dividing all logits by τ.

The temperature parameter controls the 'softness' of the teacher's predictions: the higher the temperature, the more suppressed is the difference between the largest and the smallest value of the probability vector. This allows compensating for the fact that network outputs are often overconfident, i.e. they put too much probability mass on the top predicted class, and too little on the others. The factor τ^2 in eq. (6.2) ensures that the temperature scaling does not negatively affect the gradient magnitude.

Returning to multi-exit architectures, we follow the general strategy of classical distillation but use different exits of the multi-exit classifier both as the student and the teacher. For any exit m, let $\mathcal{T}(m) \subset \{1, \ldots, M\}$ (which could be empty) be the set of teacher exits it is meant to learn from. Then we define the *distillation loss* for the n-th example as

$$\ell_{\text{dist}}(\mathbf{x}_n) \triangleq \frac{1}{M}\sum_{m=1}^{M} \frac{1}{|\mathcal{T}(m)|}\sum_{t\in\mathcal{T}(m)} \ell^\tau(\mathbf{p}_t(\mathbf{x}_n), \mathbf{p}_m(\mathbf{x}_n)), \tag{6.4}$$

In practice, there are different ways how to choose the set of teachers. The simplest choice, where all exists learn only from the last one, i.e. $\mathcal{T}(m) = \{M\}$ for $m < M$ and $\mathcal{T}(M) = \emptyset$, has worked well for us, so we propose it as the default option. However, we also study other setups, for example each exit distilling from all later exits; see Section 6.3.

Classification loss. The first term in eq. (6.1) is a standard multi-class classification loss that acts separately on each exit,

$$\ell_{\text{cls}}(\mathbf{x}_n, y_n) = \frac{1}{M} \sum_{m=1}^{M} \ell_{\text{ce}}(y_n, \mathbf{p}_m(\mathbf{x}_n)) \tag{6.5}$$

where $\ell_{\text{ce}}(y, \mathbf{p}) = -\log p_y(\mathbf{x})$ is the cross-entropy loss.

Algorithm 6.1 Distillation-based training

1: **procedure** TRAINMULTIEXIT($\boldsymbol{\theta}, \mathcal{T}, \mu, \tau_*$) // Inputs: model parameters, teacher sets, ...
2: $\mathcal{T}(:) \leftarrow \mathcal{T}(1) \cup \cdots \cup \mathcal{T}(M)$ // ... confidence threshold, temperature multiplier.
3: $\tau \leftarrow 1$
4: **for** $n = 1, \ldots, N$ **do**
5: $\mathbf{z}_n \leftarrow$ ExtractRepresentation($\mathbf{x}_n; \boldsymbol{\theta}$) // Using the shared part of the model.
6: **for** $m = 1, \ldots, M$ **do**
7: $\mathbf{p}_m \leftarrow$ softmax(exit$_m(\mathbf{z}_n; \boldsymbol{\theta})$)
8: $\bar{\mathbf{p}}_m \leftarrow$ detach(softmax(exit$_m(\mathbf{z}_n; \boldsymbol{\theta})/\tau$))
9: **end for**
10: $\ell \leftarrow \frac{1}{M} \sum_{m=1}^{M} \ell_{\text{ce}}(y_n, \mathbf{p}_m) + \frac{1}{M} \sum_{m=1}^{M} \frac{1}{|\mathcal{T}(m)|} \sum_{t \in \mathcal{T}(m)} \ell^{\tau}(\bar{\mathbf{p}}_t, \mathbf{p}_m)$
11: $\boldsymbol{\theta} \leftarrow$ Update($\boldsymbol{\theta}, \nabla \ell$)
12: **if** $\max\left(\frac{1}{|\mathcal{T}(:)|} \sum_{t \in \mathcal{T}(:)} \bar{\mathbf{p}}_t\right) > \mu$ **then** $\tau \leftarrow \tau \tau_*$
13: **end if**
14: **end for**
15: **end procedure**

6.2.4 Optimisation

We minimise the training objective (6.1) using standard gradient-based methods with mini-batches. In particular, all exits are trained at the same time and on the same data. We provide the pseudo-code (for single sample batches) in Algorithm 6.1. It consists largely of standard gradient-based optimisation. However, two aspects are specific to distillation-based training: *partial detaching* (detach; line 8), and *temperature annealing* (line 12).

Partial detaching. When minimising the loss, we have to make sure that indeed only the student learns from the teacher and not vice versa. We achieve this by treating the teachers' predictions ($\mathbf{p}_t(\mathbf{x}_n)$ in eq. (6.4), or $\bar{\mathbf{p}}_t$ in Algorithm 6.1) as constant for gradient calculation of the distillation term.

Temperature annealing. Over the course of training, networks tend to grow more confident. In the multi-exit setting, this also applies to exits that serve as teachers (see Figure 6.6). Therefore, increasingly higher temperatures are needed to 'soften' their outputs, and we achieve this by increasing the temperature during training. To this end, we introduce an adaptive annealing scheme that aims at keeping the teachers' confidence roughly constant. Specifically, we define the confidence of a classifier to be the maximum of the output vector of class probabilities, averaged over a set of examples. Let μ be an upper bound for the

desired teacher confidence. We initialise the temperature at $\tau_0 = 1$ and multiply it by a constant $\tau_* > 1$ whenever the teachers' average temperature-adjusted confidence for a training batch exceeds μ.

6.2.5 Semi-supervised training

One characteristic property of the distilliaton loss (6.4) is that it does not depend on the labels of the training data. This means it can also be computed from unlabelled training data, opening up the possibility of training multi-exit architectures in a *semi-supervised* way.

Assume that, in addition to the labelled training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, we have an additional set of unlabelled training examples, $\{\mathbf{x}_n\}_{n=N+1}^{N'}$, potentially with $N' \gg N$. We then define the semi-supervised training objective as

$$\frac{1}{N}\sum_{n=1}^{N} \ell_{\text{cls}}(\mathbf{x}_n, y_n) + \frac{1}{N'}\sum_{n=1}^{N'} \ell_{\text{dist}}(\mathbf{x}_n). \tag{6.6}$$

We can minimise this objective using the same techniques as in the fully-supervised case and with only minor modifications to the source code.

6.3 Experiments

In this section, we present our experimental results which show that distillation-based training consistently outperforms the standard training procedure for multi-exit architectures on image classification benchmarks: ImageNet (subsets) and CIFAR100 (subsets, as well as the full dataset). We further present experiments showing the tentative benefit of semi-supervised distillation when unlabelled training data is available.

We also present in-depth experiments providing insight into the working mechanism of the proposed distillation-based training procedure, in particular the temperature annealing scheme, and we discuss the choice of teachers.

6.3.1 Experimental setup

Datasets. We present experiments on two standard datasets. For *CIFAR100* [Krizhevsky and Hinton, 2009], we follow the default split, using 50,000 images (500 for each of 100 classes) for training and model selection, and we report the accuracy on the remaining 10,000 test examples. For *ImageNet (ILSVRC 2012)* [Deng et al., 2009], we use the 1.2 million `train` images of 1000 classes for training and model selection. We report the accuracy on the 50,000 images of the ILSVRC `val` set. During training, we apply data augmentation as in [He et al., 2016, Huang et al., 2018]. For testing, we resize the images to 256×256 pixels and center-crop them to 224×224. For both datasets, we pre-process all images by subtracting the channel mean and dividing by the channel standard deviation.

Because we are particularly interested in the low-data regime, we perform experiments using only subsets of the available data: by ImageNet(X) we denote a dataset with X randomly selected examples from each ImageNet class (which are then split 90%/10% into a training and a model selection part). By CIFAR(X) we denote subsets of CIFAR100 that are constructed analogously to the above, but always with 50 images used for model selection (using 10% would be too few for this dataset), and the remaining $X - 50$ for

training. As additional unlabelled data for the experiments on semi-supervised learning we use $500 - X$ (in the case of CIFAR100) or $700 - X$ (in the case of ImageNet) images per class, sampled randomly from all remaining images, i.e. the ones that were not selected as training or validation data (nor as test data, of course).

Model architecture. We use the Multi-Scale DenseNet [Huang et al., 2018], a state-of-the-art multi-exit architecture with convolutional blocks arranged in a grid of multiple scales (rows) and multiple layers (columns). We train the MSDNets ourselves, following the original architectures and hyper-parameters.[1] The CIFAR MSDNet has 3×24 blocks and 11 exits, one attached to every second layer, starting from layer 4. The ImageNet MSDNet has 4×38 blocks and 5 exits, one on every seventh layer, starting from layer 10.

Baseline. We compare distillation training to the traditional way of training multi-exit architectures, namely by minimising the *exit-wise loss* (used e.g. by Huang et al. [2018], Kim et al. [2018], Teerapittayanon et al. [2016]),

$$\frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \ell_{\mathsf{ce}}(y_n, \mathbf{p}_m(\mathbf{x}_n)). \tag{6.7}$$

This coincides with using just the *classification loss* (6.5) of our training objective. Because labels for all training examples are needed to compute the loss, (6.7) does not have an obvious extension to semi-supervised learning.

Optimisation and hyper-parameters. We train all models from a random initialisation by SGD with Nesterov momentum, an initial learning rate of 0.5, a momentum weight of 0.9, and a weight decay of 10^{-4}. For CIFAR100, we set the batch size to 64 and train for 300 epochs. The learning rate is divided by 10 after epochs 150 and 225. For ImageNet, we set the batch size to 256 and train for 90 epochs. The learning rate is divided by 10 after epochs 30 and 60.

For the temperature annealing we use a confidence limit of $\mu = 0.5$ for CIFAR100 and $\mu = 0.1$ for ImageNet, and $\tau_* = 1.05$ as the temperature multiplier.

Repeated runs. We repeat each experiment 5 or 10 times, each time with a different random subset of the training data and different weight initialisation. We report the average performance across the repeated runs as well as its 95% confidence interval (i.e. 1.96 times the standard error).

6.3.2 Main results: budget-mode accuracy

We first present results from experiments when operating the model in *budget-mode*, i.e. with a known time budget at test time. As described in Section 6.2.2, for any value of the budget, we identify the best exit (according to the validation set) that can be computed within the budget, and evaluate its decision. This is often the latest exit among the available ones, but not always: for example, in the low-data regime, the additional capacity of a late exit may make it more likely to overfit, and an intermediate exit might perform better.

Figures 6.2 and 6.3 show the results for ImageNet and CIFAR100, respectively. Numeric results can be found in Tables 6.1 and 6.2. For each training procedure, we report the

[1] Our implementation achieves very similar performance to the original MSDNet, e.g. $\approx 75\%$ accuracy on CIFAR100.

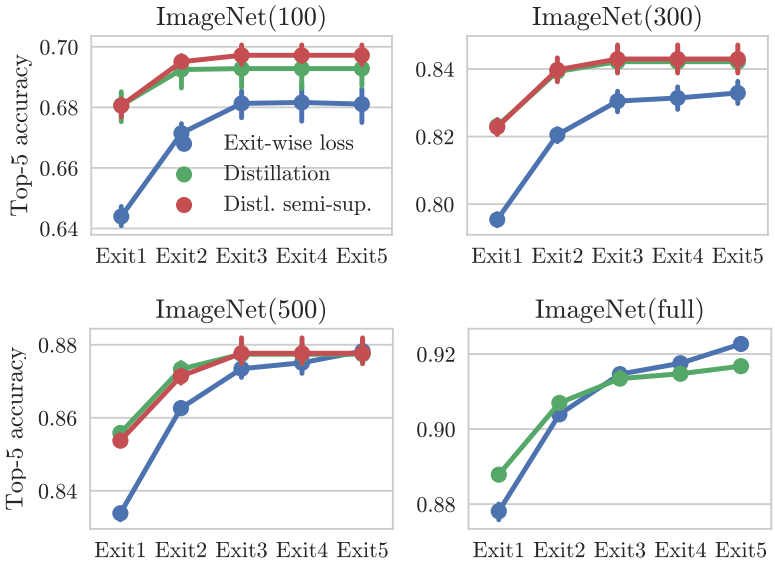


Figure 6.2: Top-5 accuracy as a function of computational budget (denominated in available exits). MSDNet trained by the exit-wise loss (blue) vs. trained by distillation (green) vs. trained by semi-supervised distillation (red) on ImageNet ILSVRC2012 with 100, 300, 500, and all available (≥ 700) images per class.

	ImageNet(100)			ImageNet(300)		
	Exit-wise loss	Distillation	Distl. semi-sup.	Exit-wise loss	Distillation	Distl. semi-sup.
Exit 1	64.4 ± 0.4	$\mathbf{68.1 \pm 0.5}$	$\mathbf{68.1 \pm 0.4}$	79.5 ± 0.2	$\mathbf{82.3 \pm 0.2}$	$\mathbf{82.3 \pm 0.3}$
Exit 2	67.1 ± 0.3	$\mathbf{69.2 \pm 0.6}$	$\mathbf{69.5 \pm 0.1}$	82.1 ± 0.2	$\mathbf{83.9 \pm 0.3}$	$\mathbf{84.0 \pm 0.5}$
Exit 3	68.1 ± 0.5	$\mathbf{69.3 \pm 0.6}$	$\mathbf{69.7 \pm 0.4}$	83.0 ± 0.3	$\mathbf{84.2 \pm 0.3}$	$\mathbf{84.3 \pm 0.6}$
Exit 4	68.2 ± 0.6	69.3 ± 0.6	$\mathbf{69.7 \pm 0.4}$	83.1 ± 0.3	$\mathbf{84.2 \pm 0.3}$	$\mathbf{84.3 \pm 0.6}$
Exit 5	68.1 ± 0.6	69.3 ± 0.6	$\mathbf{69.7 \pm 0.4}$	83.3 ± 0.3	$\mathbf{84.2 \pm 0.3}$	$\mathbf{84.3 \pm 0.6}$

	ImageNet(500)			ImageNet(full)	
	Exit-wise loss	Distillation	Distl. semi-sup.	Exit-wise loss	Distillation
Exit 1	83.4 ± 0.2	$\mathbf{85.6 \pm 0.1}$	$\mathbf{85.4 \pm 0.1}$	87.8 ± 0.2	$\mathbf{88.8 \pm 0.1}$
Exit 2	86.3 ± 0.2	$\mathbf{87.3 \pm 0.2}$	$\mathbf{87.1 \pm 0.3}$	90.4 ± 0.1	$\mathbf{90.7 \pm 0.1}$
Exit 3	87.3 ± 0.3	87.7 ± 0.3	87.8 ± 0.4	91.5 ± 0.1	91.3 ± 0.2
Exit 4	87.5 ± 0.3	87.7 ± 0.3	87.8 ± 0.4	$\mathbf{91.8 \pm 0.1}$	91.5 ± 0.1
Exit 5	87.8 ± 0.3	87.7 ± 0.3	87.8 ± 0.4	$\mathbf{92.3 \pm 0.1}$	91.7 ± 0.2

Table 6.1: Top-5 accuracy in % (mean $\pm$ 1.96 stderr) for different computational budgets (denominated in available exits). MSDNet trained by the exit-wise loss vs. trained by distillation vs. trained by semi-supervised distillation on ImageNet with 100, 300, 500 and all available (≥ 700) training images per class. Bold values indicate statistically significant improvements.

resulting model's accuracy for different values of the budget: when only *Exit1* is available, when *Exit1* and *Exit2* are available, and so on.

ImageNet results. Figure 6.2 and Table 6.1 compare distillation-based training and exit-wise training. Distillation-based training consistently outperforms exit-wise training, in many settings substantially. For most computational budgets, the distillation-trained model has a higher or comparable accuracy, with accuracy gains of up to 3.8%. Conversely, to achieve

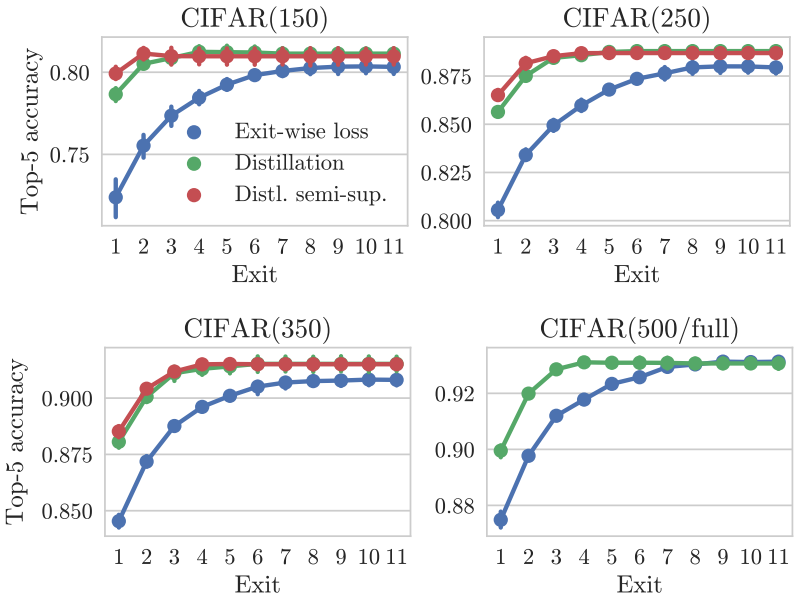


Figure 6.3: Top-5 accuracy as a function of computational budget (denominated in available exits). MSDNet trained by the exit-wise loss (blue) vs. trained by distillation (green) vs. trained by semi-supervised distillation (red) on CIFAR100 with 150, 250, 350, and 500 images per class.

	CIFAR(150)			CIFAR(250)		
	Exit-wise loss	Distillation	Distl. semi-sup.	Exit-wise loss	Distillation	Distl. semi-sup.
Exit 1	72.4 ± 1.3	$\mathbf{78.7 \pm 0.4}$	$\mathbf{79.9 \pm 0.4}$	80.6 ± 0.4	$\mathbf{85.6 \pm 0.2}$	$\mathbf{86.5 \pm 0.3}$
Exit 2	75.5 ± 0.7	$\mathbf{80.5 \pm 0.3}$	$\mathbf{81.1 \pm 0.4}$	83.4 ± 0.3	$\mathbf{87.5 \pm 0.2}$	$\mathbf{88.2 \pm 0.3}$
Exit 3	77.4 ± 0.6	$\mathbf{80.9 \pm 0.4}$	$\mathbf{81.0 \pm 0.5}$	84.9 ± 0.3	$\mathbf{88.4 \pm 0.2}$	$\mathbf{88.5 \pm 0.2}$
Exit 4	78.4 ± 0.4	$\mathbf{81.2 \pm 0.4}$	$\mathbf{81.0 \pm 0.5}$	86.0 ± 0.3	$\mathbf{88.6 \pm 0.2}$	$\mathbf{88.7 \pm 0.2}$
Exit 5	79.2 ± 0.3	$\mathbf{81.2 \pm 0.4}$	$\mathbf{81.0 \pm 0.5}$	86.8 ± 0.2	$\mathbf{88.8 \pm 0.3}$	$\mathbf{88.7 \pm 0.2}$
Exit 6	79.8 ± 0.2	$\mathbf{81.2 \pm 0.4}$	$\mathbf{81.0 \pm 0.5}$	87.4 ± 0.2	$\mathbf{88.8 \pm 0.3}$	$\mathbf{88.7 \pm 0.2}$
Exit 7	80.1 ± 0.4	$\mathbf{81.1 \pm 0.3}$	$\mathbf{81.0 \pm 0.5}$	87.6 ± 0.3	$\mathbf{88.8 \pm 0.3}$	$\mathbf{88.7 \pm 0.2}$
Exit 8	80.3 ± 0.4	$\mathbf{81.1 \pm 0.3}$	81.0 ± 0.5	87.9 ± 0.3	$\mathbf{88.8 \pm 0.3}$	$\mathbf{88.7 \pm 0.2}$
Exit 9	80.3 ± 0.5	81.1 ± 0.3	81.0 ± 0.5	88.0 ± 0.3	$\mathbf{88.8 \pm 0.3}$	$\mathbf{88.7 \pm 0.2}$
Exit 10	80.4 ± 0.5	81.1 ± 0.3	81.0 ± 0.5	88.0 ± 0.3	$\mathbf{88.8 \pm 0.3}$	$\mathbf{88.7 \pm 0.2}$
Exit 11	80.3 ± 0.5	81.1 ± 0.3	81.0 ± 0.5	87.9 ± 0.3	$\mathbf{88.8 \pm 0.3}$	$\mathbf{88.7 \pm 0.2}$

	CIFAR(350)			CIFAR(500)	
	Exit-wise loss	Distillation	Distl. semi-sup.	Exit-wise loss	Distillation
Exit 1	84.5 ± 0.3	$\mathbf{88.1 \pm 0.3}$	$\mathbf{88.5 \pm 0.3}$	87.5 ± 0.3	$\mathbf{90.0 \pm 0.2}$
Exit 2	87.2 ± 0.3	$\mathbf{90.1 \pm 0.2}$	$\mathbf{90.4 \pm 0.2}$	89.8 ± 0.1	$\mathbf{92.0 \pm 0.2}$
Exit 3	88.8 ± 0.2	$\mathbf{91.1 \pm 0.3}$	$\mathbf{91.2 \pm 0.2}$	91.2 ± 0.1	$\mathbf{92.9 \pm 0.2}$
Exit 4	89.6 ± 0.2	$\mathbf{91.3 \pm 0.3}$	$\mathbf{91.5 \pm 0.2}$	91.8 ± 0.2	$\mathbf{93.1 \pm 0.1}$
Exit 5	90.1 ± 0.2	$\mathbf{91.4 \pm 0.3}$	$\mathbf{91.5 \pm 0.1}$	92.3 ± 0.1	$\mathbf{93.1 \pm 0.2}$
Exit 6	90.5 ± 0.3	$\mathbf{91.5 \pm 0.4}$	$\mathbf{91.5 \pm 0.2}$	92.6 ± 0.1	$\mathbf{93.1 \pm 0.2}$
Exit 7	90.7 ± 0.2	$\mathbf{91.5 \pm 0.4}$	$\mathbf{91.5 \pm 0.2}$	92.9 ± 0.1	93.1 ± 0.2
Exit 8	90.8 ± 0.2	$\mathbf{91.5 \pm 0.4}$	$\mathbf{91.5 \pm 0.2}$	93.0 ± 0.1	93.1 ± 0.2
Exit 9	90.8 ± 0.2	$\mathbf{91.5 \pm 0.4}$	$\mathbf{91.5 \pm 0.2}$	93.1 ± 0.1	93.1 ± 0.2
Exit 10	90.8 ± 0.3	$\mathbf{91.5 \pm 0.4}$	$\mathbf{91.5 \pm 0.2}$	93.1 ± 0.2	93.1 ± 0.2
Exit 11	90.8 ± 0.2	$\mathbf{91.5 \pm 0.4}$	$\mathbf{91.5 \pm 0.2}$	93.1 ± 0.2	93.1 ± 0.2

Table 6.2: Top-5 accuracy in % (mean $\pm$ 1.96 stderr) for different computational budgets (denominated in available exits). MSDNet trained by the exit-wise loss vs. trained by distillation vs. trained by semi-supervised distillation on CIFAR100 with 150, 250, 350, and 500 images per class. Bold values indicate statistically significant improvements.

any given accuracy, the distillation-trained model typically requires far less computation, especially in the data-constrained regime. For example, in the case of ImageNet(100), already *Exit1* suffices to match the accuracy of the exit-wise trained model at *any* budget. Similarly, in the case of ImageNet(300), already at the time *Exit2* becomes available, the distillation-trained model dominates the exit-wise trained model at any budget.

Overall, two main factors seem to affect the performance gap: a) The amount of training data: comparing the results for ImageNet(100) to those for ImageNet(300), ImageNet(500) and ImageNet(full), we see that the smaller the training set, the bigger the benefit from distillation. In the regime of very large data (full ImageNet), distillation seems to trade off the accuracy of early and late exits, instead of providing a uniform improvement. This agrees with earlier studies, e.g. [Hinton et al., 2014], that found distillation to have a regularising effect, i.e. it helps prevent overfitting in the low-data regime. b) The inference budget: within each subplot, the largest gains are realised for the smallest inference budgets. Intuitively, this makes sense, as the earliest exits can benefit the most from a teacher during learning. In combination, the results suggest that distillation training can provide a large accuracy boost, especially when the amount of training data and/or the computational resources at test time are limited.

Figure 6.2 and Table 6.1 also show results for the semi-supervised variant of distillation-based training, as described in Section 6.2.5. We observe an additional small improvement over the fully-supervised variant, especially when labelled data is limited and unlabelled data plentiful.

CIFAR100 results. We present analogous results for CIFAR100 in Figure 6.3 and Table 6.2. We observe similar trends as for ImageNet, though in this case distillation training uniformly outperforms exit-wise training and yields an up to 6.3% improvement in accuracy for a fixed budget. Conversely, distillation training enables the resulting model to stop already after *Exit2* or *Exit3* with comparable accuracy as the conventionally trained model when executed in full. As previously for ImageNet, here, too, we observe that the gains from distillation are largest when training data is limited, and when the inference budget is low.

The semi-supervised variant provides an additional small but consistent improvement. For example, for *Exit1*, the additional unlabelled data translates into $1.2\%, 0.9\%,$ and 0.4% increase in accuracy for CIFAR(150), CIFAR(250), and CIFAR(350) respectively.

6.3.3 Main results: anytime-mode accuracy

In a second set of experiments, we operate the multi-exit model in anytime-mode, i.e. the model evaluates all its exits in turn until the (unknown) computational budget is spent, at which point it returns the ensembled prediction of all completed exits. As before, we report multiclass accuracy for different computational budgets, this time denominated in the number of completed exits, or the size of the ensemble.

The results for ImageNet and CIFAR100 are shown in Figures 6.4 and 6.5 respectively. The corresponding numeric results are given in Tables 6.3 and 6.4. The results are similar to those for budget-mode evaluation. Across datasets, dataset sizes (except for the very large-scale regime), and computation budgets, the models trained with distillation clearly outperform the model trained without it. The results for semi-supervised learning are less clear: for early exits, the unlabelled data often helps, but we observe a small drop of

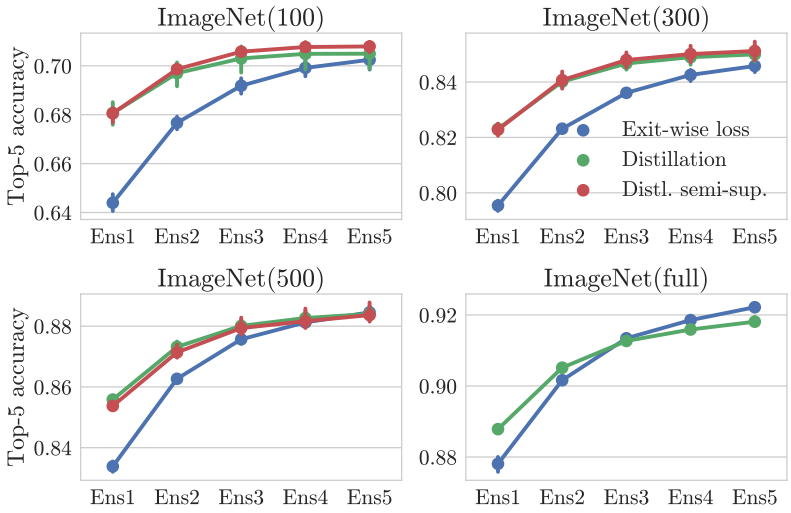


Figure 6.4: Top-5 accuracy of first-m-exits ensembles ($m = 1, \ldots, 5$) trained by the exit-wise loss (blue) vs. trained by distillation (green) vs. trained by semi-supervised distillation (red) on ImageNet ILSVRC2012 with 100, 300, 500 or all available (≥ 700) training images per class.

	ImageNet(100)			ImageNet(300)		
	Exit-wise loss	Distillation	Distl. semi-sup.	Exit-wise loss	Distillation	Distl. semi-sup.
Ensemble 1	64.4 ± 0.4	$\mathbf{68.1 \pm 0.5}$	$\mathbf{68.1 \pm 0.4}$	79.5 ± 0.2	$\mathbf{82.3 \pm 0.2}$	$\mathbf{82.3 \pm 0.3}$
Ensemble 2	67.7 ± 0.3	$\mathbf{69.7 \pm 0.6}$	$\mathbf{69.9 \pm 0.2}$	82.3 ± 0.1	$\mathbf{84.0 \pm 0.2}$	$\mathbf{84.1 \pm 0.5}$
Ensemble 3	69.2 ± 0.3	$\mathbf{70.3 \pm 0.6}$	$\mathbf{70.6 \pm 0.1}$	83.6 ± 0.2	$\mathbf{84.7 \pm 0.3}$	$\mathbf{84.8 \pm 0.4}$
Ensemble 4	69.9 ± 0.3	70.5 ± 0.6	$\mathbf{70.8 \pm 0.2}$	84.3 ± 0.2	$\mathbf{84.9 \pm 0.3}$	$\mathbf{85.0 \pm 0.5}$
Ensemble 5	70.2 ± 0.4	70.5 ± 0.7	70.8 ± 0.2	84.6 ± 0.2	85.0 ± 0.3	85.1 ± 0.5

	ImageNet(500)			ImageNet(full)	
	Exit-wise loss	Distillation	Distl. semi-sup.	Exit-wise loss	Distillation
Ensemble 1	83.4 ± 0.2	$\mathbf{85.6 \pm 0.1}$	$\mathbf{85.4 \pm 0.1}$	87.8 ± 0.2	$\mathbf{88.8 \pm 0.1}$
Ensemble 2	86.3 ± 0.1	$\mathbf{87.3 \pm 0.2}$	$\mathbf{87.1 \pm 0.3}$	90.2 ± 0.1	$\mathbf{90.5 \pm 0.1}$
Ensemble 3	87.6 ± 0.2	$\mathbf{88.0 \pm 0.2}$	87.9 ± 0.3	91.3 ± 0.1	91.3 ± 0.1
Ensemble 4	88.1 ± 0.2	88.3 ± 0.3	88.2 ± 0.4	$\mathbf{91.9 \pm 0.1}$	91.6 ± 0.1
Ensemble 5	88.5 ± 0.2	88.4 ± 0.3	88.4 ± 0.4	$\mathbf{92.2 \pm 0.1}$	91.8 ± 0.1

Table 6.3: Top-5 accuracy in % (mean $\pm$ 1.96 stderr) of first-m-exits ensembles ($m = 1, \ldots, 5$) trained by the exit-wise loss vs. trained by distillation vs. trained by semi-supervised distillation on ImageNet ILSVRC2012 with 100, 300, 500 or all available (≥ 700) training images per class.

accuracy of the late exits for CIFAR(150). Still, the proposed method outperforms the exit-wise trained model.

6.3.4 Additional experiments

Temperature annealing. In this section, provide further insight and justification for the proposed temperature annealing scheme. Figure 6.6 shows the teacher's confidence (blue) during training. One can see that it changes markedly and generally increases. The proposed temperature-scaling procedure reacts to this by raising the temperature over time

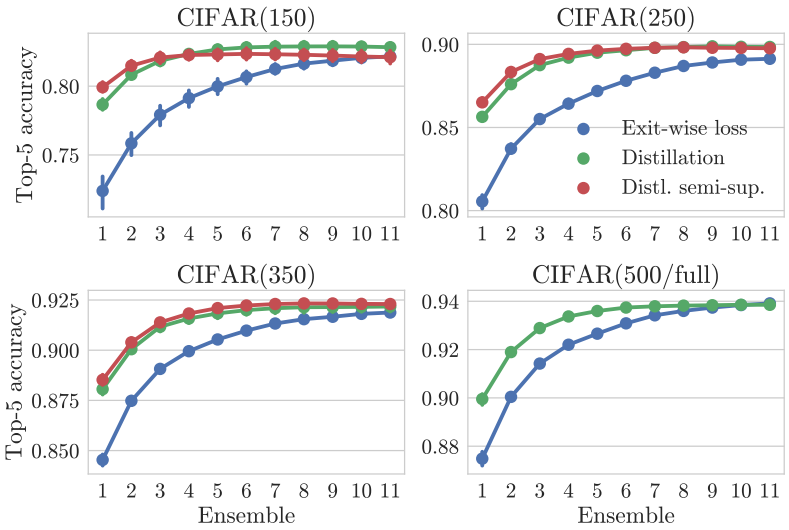


Figure 6.5: Top-5 accuracy of first-m-exits ensembles ($m = 1, \ldots, 11$) trained by the exit-wise loss (blue) vs. trained by distillation (green) vs. trained by semi-supervised distillation (red) on CIFAR100 with 150, 250, 350 or 500 images per class.

	CIFAR(150)			CIFAR(250)		
	Exit-wise loss	Distillation	Distl. semi-sup.	Exit-wise loss	Distillation	Distl. semi-sup.
Ensemble 1	72.4 ± 1.3	$\mathbf{78.7 \pm 0.4}$	$\mathbf{79.9 \pm 0.4}$	80.6 ± 0.4	$\mathbf{85.6 \pm 0.2}$	$\mathbf{86.5 \pm 0.3}$
Ensemble 2	75.8 ± 0.9	$\mathbf{80.8 \pm 0.3}$	$\mathbf{81.5 \pm 0.4}$	83.7 ± 0.3	$\mathbf{87.6 \pm 0.2}$	$\mathbf{88.3 \pm 0.3}$
Ensemble 3	77.9 ± 0.8	$\mathbf{81.8 \pm 0.3}$	$\mathbf{82.1 \pm 0.4}$	85.5 ± 0.2	$\mathbf{88.7 \pm 0.2}$	$\mathbf{89.1 \pm 0.2}$
Ensemble 4	79.1 ± 0.7	$\mathbf{82.3 \pm 0.3}$	$\mathbf{82.3 \pm 0.4}$	86.4 ± 0.2	$\mathbf{89.2 \pm 0.2}$	$\mathbf{89.4 \pm 0.2}$
Ensemble 5	80.0 ± 0.6	$\mathbf{82.7 \pm 0.3}$	$\mathbf{82.3 \pm 0.4}$	87.2 ± 0.2	$\mathbf{89.5 \pm 0.2}$	$\mathbf{89.6 \pm 0.2}$
Ensemble 6	80.7 ± 0.5	$\mathbf{82.8 \pm 0.4}$	$\mathbf{82.3 \pm 0.4}$	87.8 ± 0.2	$\mathbf{89.6 \pm 0.2}$	$\mathbf{89.7 \pm 0.2}$
Ensemble 7	81.2 ± 0.4	$\mathbf{82.9 \pm 0.4}$	$\mathbf{82.3 \pm 0.4}$	88.3 ± 0.2	$\mathbf{89.8 \pm 0.2}$	$\mathbf{89.8 \pm 0.2}$
Ensemble 8	81.6 ± 0.4	$\mathbf{82.9 \pm 0.4}$	82.3 ± 0.4	88.7 ± 0.2	$\mathbf{89.8 \pm 0.2}$	$\mathbf{89.8 \pm 0.2}$
Ensemble 9	81.9 ± 0.3	$\mathbf{82.9 \pm 0.3}$	82.2 ± 0.4	88.9 ± 0.2	$\mathbf{89.9 \pm 0.2}$	$\mathbf{89.8 \pm 0.2}$
Ensemble 10	82.1 ± 0.3	$\mathbf{82.9 \pm 0.4}$	82.2 ± 0.5	89.1 ± 0.2	$\mathbf{89.9 \pm 0.2}$	$\mathbf{89.8 \pm 0.2}$
Ensemble 11	82.2 ± 0.3	82.8 ± 0.3	82.1 ± 0.5	89.1 ± 0.2	$\mathbf{89.8 \pm 0.2}$	$\mathbf{89.8 \pm 0.2}$

	CIFAR(350)			CIFAR(500)	
	Exit-wise loss	Distillation	Distl. semi-sup.	Exit-wise loss	Distillation
Ensemble 1	84.5 ± 0.3	$\mathbf{88.1 \pm 0.3}$	$\mathbf{88.5 \pm 0.3}$	87.5 ± 0.3	$\mathbf{90.0 \pm 0.2}$
Ensemble 2	87.5 ± 0.2	$\mathbf{90.0 \pm 0.2}$	$\mathbf{90.4 \pm 0.2}$	90.0 ± 0.1	$\mathbf{91.9 \pm 0.2}$
Ensemble 3	89.1 ± 0.2	$\mathbf{91.2 \pm 0.2}$	$\mathbf{91.4 \pm 0.2}$	91.4 ± 0.1	$\mathbf{92.9 \pm 0.1}$
Ensemble 4	90.0 ± 0.2	$\mathbf{91.6 \pm 0.2}$	$\mathbf{91.8 \pm 0.2}$	92.2 ± 0.1	$\mathbf{93.4 \pm 0.1}$
Ensemble 5	90.5 ± 0.2	$\mathbf{91.8 \pm 0.3}$	$\mathbf{92.1 \pm 0.2}$	92.7 ± 0.1	$\mathbf{93.6 \pm 0.1}$
Ensemble 6	91.0 ± 0.1	$\mathbf{92.0 \pm 0.3}$	$\mathbf{92.2 \pm 0.1}$	93.1 ± 0.1	$\mathbf{93.7 \pm 0.1}$
Ensemble 7	91.3 ± 0.1	$\mathbf{92.1 \pm 0.2}$	$\mathbf{92.3 \pm 0.2}$	93.4 ± 0.1	$\mathbf{93.8 \pm 0.1}$
Ensemble 8	91.5 ± 0.1	$\mathbf{92.1 \pm 0.3}$	$\mathbf{92.3 \pm 0.2}$	93.6 ± 0.1	$\mathbf{93.8 \pm 0.1}$
Ensemble 9	91.7 ± 0.1	$\mathbf{92.1 \pm 0.3}$	$\mathbf{92.3 \pm 0.2}$	93.7 ± 0.1	93.8 ± 0.1
Ensemble 10	91.8 ± 0.1	92.2 ± 0.3	$\mathbf{92.3 \pm 0.2}$	93.8 ± 0.1	93.9 ± 0.1
Ensemble 11	91.9 ± 0.1	92.2 ± 0.2	$\mathbf{92.3 \pm 0.2}$	93.9 ± 0.1	93.9 ± 0.1

Table 6.4: Top-5 accuracy in % (mean $\pm$ 1.96 stderr) of first-m-exits ensembles trained by the exit-wise loss vs. trained by distillation vs. trained by semi-supervised distillation on CIFAR100 with 150, 250, 350 or 500 images per class.

(purple). The result is that the temperature-adjusted confidence (green) remains roughly constant, and slightly below the confidence limit μ (red).

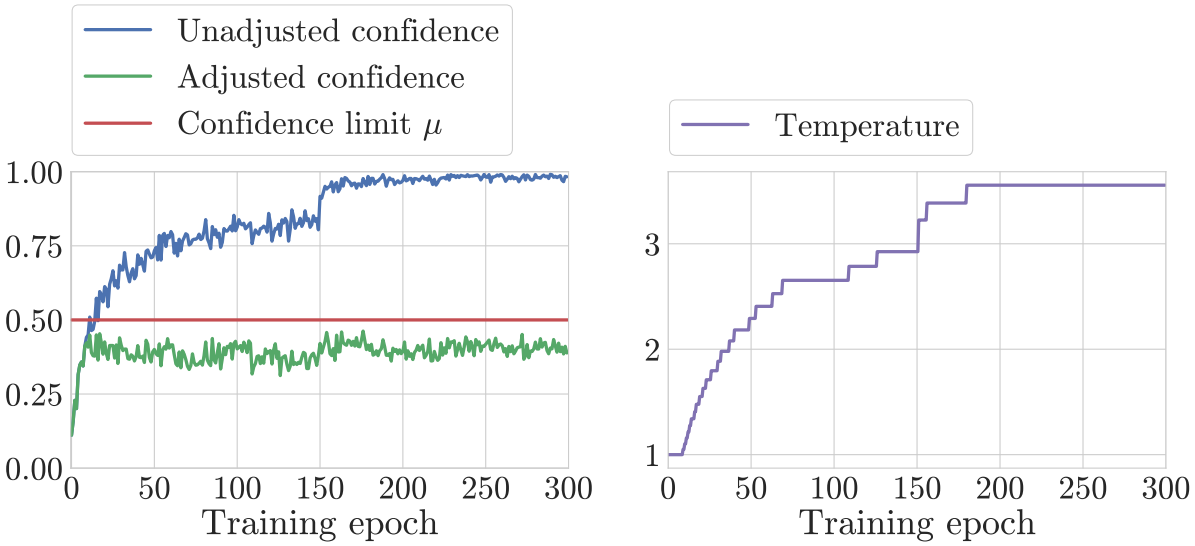


Figure 6.6: Confidence of MSDNet's last exit with and without temperature annealing throughout training on CIFAR(150). At epochs 150 and 225, the learning rate drops.

Figure 6.7 shows the accuracy curves of five MSDNets trained by distillation, each with a different temperature setting. We compare the proposed annealing scheme (green) to training with constant temperature, $T \in \{1.0, 2.0, 3.0, 4.0\}$. The figure shows that while using no temperature at all ($T = 1.0$) leads to significant accuracy drops, as long as the temperature is 'high enough', its exact value seems to matter little for the final model's accuracy. Still, the proposed annealing scheme performs as well as, or better than any choice of a constant temperature, and has the advantage of being easier to tune.

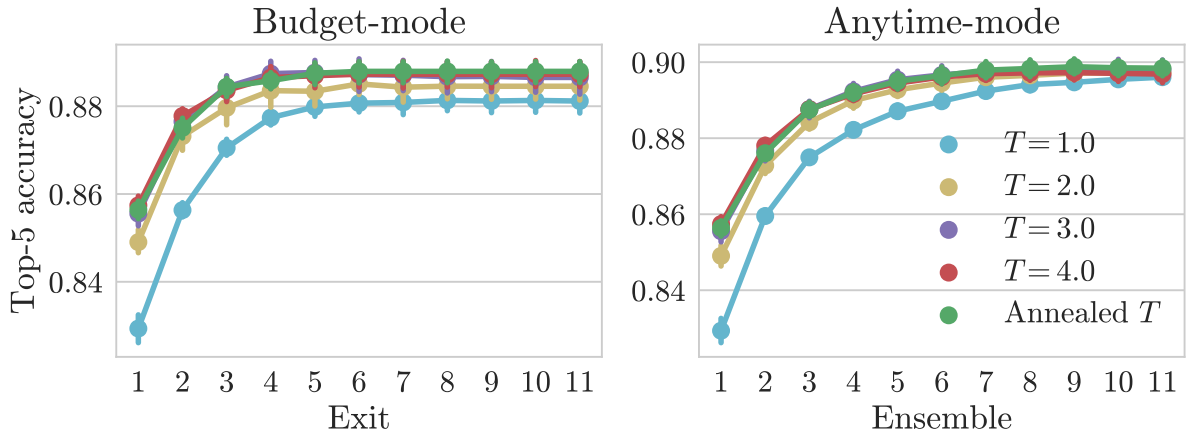


Figure 6.7: Top-5 accuracy of five models trained by distillation, each with a different temperature setting, on CIFAR(250). Results for different computational budgets in both the budget-mode (left) and the anytime-mode (right).

Choice of teachers. For all experiments reported so far, we used the last exit as the teacher for all other exits. We also performed exploratory studies on how the choice of teacher affects the overall performance, but found the effect to be minor (see Appendix E.1).

6.4 Conclusion

In this chapter, we explored *distillation-based training* for multi-exit image classification architectures. The method is conceptually simple, architecture-agnostic, and as our experiments show, it provides large and robust improvements over the state-of-the-art training procedure, especially in data- or computation-constrained settings. It also naturally supports learning from additional unlabelled training data.

CHAPTER 7

Conclusion

In this work, we studied different aspects of *underspecification in deep learning*, the learning regime in which the number of degrees of freedom afforded by deep architectures vastly exceeds the number of constraints posed by the training data. While classical statistics and machine learning theory assert a clear negative effect of underspecification on generalisation, the situation is more complex in deep learning: underspecification may refer to one of two distinct phenomena which differ markedly both in terms of their extent and relationship to generalisation.

- In Chapter 2, we first formulate how deep learning models are underspecified on two levels: first, any given training dataset can be fit in many different ways (by different hypotheses), and second, any given hypothesis can itself be expressed in many different ways (by different parameter vectors). We refer to the second kind of underspecification as *parameterisation redundancy* and we precisely characterise its extent. Specifically, we show there are only two types of universally applicable redundancies, neuron permutations and rescalings. The corresponding redundant subspace is much smaller than we expected a priori.
- In Chapter 3, we tackle the central mystery of deep learning: How come deep networks generalise when the data is provably insufficient to pin down a good solution? This pertains to *hypothesis class redundancy*, the other kind of underspecification identified above. The most plausible narrative is that deep learning implicitly adheres to a favourable criterion for choosing between hypotheses with equal empirical risk. This implicit criterion, called the *inductive bias*, is still poorly understood. Here we focus on the analytically tractable setting of orthogonally separable data and show that given the choice, neural networks learn classifiers with a large margin.
- Although the inductive bias in deep learning does tend to be favourable, this is not true in all settings. In Chapter 4, we consider representation learning with variational autoencoders and argue that the default implicit bias is insufficient to guarantee good performance – in fact, it is harmful. We then propose a method for steering the underspecified learning problem: a hand-crafted regulariser based on mutual information.
- In Chapter 5, we consider a different way of tightening the specification: strong supervision, or more informative data. We consider the specific case of *soft-label*

supervision in binary classification, where instead of binary labels, the learning algorithm observes real-valued labels. We then derive a generalisation bound for linear networks supervised in this way and verify that soft labels facilitate fast learning.

- And finally, in Chapter 6 we explore an application of soft-label supervision, namely to self-learning of multi-exit models. We demonstrate that soft labels are beneficial not only when learning from world-observations (as shown above), but also when learning from a more expressive, but not necessarily fully trained, model.

We think of the work on parameterisation redundancy and the work on hypothesis class redundancy as related, but warranting distinction. As we have shown, the redundancy in parameterisation is not as severe, and its connections to generalisation are perhaps less obvious. The most promising pathway to impact in our view is via a better understanding of the loss landscape. That said, parameterisation redundancy is a very recent concept, and may yet inspire unexpected, hard-to-anticipate insights.

The literature on inductive biases is more established and constantly growing – compared to the state of the art five years ago, we find that the understanding of inductive biases has progressed substantially. This is especially true of the literature on nonlinear models [Chizat and Bach, 2020, Ji and Telgarsky, 2020]. On the one hand, we are hopeful that this great undertaking continues into the future and that it leads to a new class of generalisation guarantees for deep learning. On the other hand, much of the inductive bias work focuses on binary classification, and neglects learning scenarios in which the default inductive bias may not be as favourable and where some additional steering is necessary (such as in representation learning, as discussed above). These settings are currently a domain of heuristic approaches. We hope that future work yields more principled approaches to dealing with underspecification, and a better understanding of when they should be preferred to the default inductive bias.

In the next section, we highlight a few particularly promising directions for further research.

7.1 Future work

The effect of parameterisation on the loss landscape. How we parameterise our neural networks, i.e. how the parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ determine the input-output mapping $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, affects the shape of the loss landscape, $\boldsymbol{\theta} \mapsto \ell(\boldsymbol{\theta})$. This effect could potentially be profound, e.g. it could make training more effective or easier to analyse. While it has been observed that such benefits can be attained by architectural changes [Li et al., 2018] or normalisation schemes [Santurkar et al., 2018], to our knowledge very little is known about how the loss landscape is shaped by different parameterisations.

In the context of the current work, one important question is whether introducing additional parameterisation redundancies (beyond the already present permutation and rescaling) helps optimisation, and what kinds of reparameterisations are the most helpful.

Inductive bias via descent-reachability. Throughout this work, we used the term inductive bias to mean the preferential selection of certain *hypotheses* by the learning algorithm when multiple hypotheses are tied in empirical risk. However, for learning algorithms that operate in the parameter space, it may make more sense to explicitly consider the preferential selection of certain parts of the parameter space, and to define the inductive bias in

terms of properties of the loss landscape. One example of such an approach is the 'flat minima' hypothesis [Keskar et al., 2017], according to which stochastic gradient descent (with a small batch size) prefers parts of the parameter space where the loss is flat.

We put forward an alternative hypothesis that is less contingent on the presence of gradient noise: gradient descent and variants are more likely to converge to parameter vectors that are 'descent-reachable' from a large part of the parameter space, i.e. the algorithm is more likely to return $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ if there is a descent path to it starting from many different $\boldsymbol{\eta} \in \boldsymbol{\Theta}$. The degree of preference for a hypothesis h would then be determined by the volume of parameter vectors $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ that implement it, $h_{\boldsymbol{\theta}} \equiv h$, and the descent-reachability of those parameter vectors.

Inductive biases of general architectures. The quest for understanding the inductive bias in deep learning is one of the most important endeavours in theoretical ML – unless there is solid understanding of what it is that deep nets actually learn, there is little hope of rigorous guarantees on performance or robustness. Though this understanding is slowly emerging for (shallow) fully-connected networks, it is questionable how applicable the results are to other, more commonly used architectures such as deep residual architectures or transformers. We see the understanding of inductive biases of such modern architectures as the next important frontier.

One could imagine two general strategies for getting there. First, one might hope to solve the inductive bias problem for each new architecture separately. Second, one might develop an architecture-agnostic way of characterising the inductive bias. We are much more optimistic about the second approach, despite its more ambitious scope. That said, we do expect to hit the limits of current analytical tools (such as classical optimisation tools or the mean-field approach). We are more optimistic about somewhat more exotic paths forward, for example a hybrid experimental-analytical approach, or looser, less detailed abstractions of the learning model and algorithm.

Bibliography

Francesca Albertini and Eduardo D. Sontag. For neural networks, function determines form. *Neural networks*, 6(7):975–990, 1993a.

Francesca Albertini and Eduardo D. Sontag. Identifiability of discrete-time neural networks. In *European Control Conference*, 1993b.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E. Dahl, and Geoffrey E. Hinton. Large scale distributed neural network training through online distillation. In *International Conference on Learning Representations (ICLR)*, 2018.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learing (ICML)*, 2018.

Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

Vijay Badrinarayanan, Bamdev Mishra, and Roberto Cipolla. Symmetry-invariant optimization in deep networks. In *International Conference on Learning Representations (ICLR)*, 2016.

David Barber and Felix V Agakov. The IM Algorithm: A variational approach to information maximization. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2003.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. In *Proceedings of the National Academy of Sciences (PNAS)*, 2019.

Djalel Benbouzid, Róbert Busa-Fekete, and Balázs Kégl. Fast classification using sparse decision DAGs. In *International Conference on Machine Learing (ICML)*, 2012.

Julius Berner, Dennis Elbrächter, and Philipp Grohs. How degenerate is the parametrization of neural networks with the ReLU activation function? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

D. P. Bertsekas. *Nonlinear Optimization*. Athena Scientific, Belmont, MA, 2nd edition, 1999.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *AAAI Conference on Artificial Intelligence*, 2018.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

Z. Berkay Celik, David Lopez-Paz, and Patrick McDaniel. Patient-driven privacy control through generalized distillation. In *IEEE Symposium on Privacy-Aware Computing (PAC)*, 2017.

An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *International Conference on Learning Representations (ICLR)*, 2017.

Lénaïc Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Computational Learning Theory (COLT)*, 2020.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in supervised differentiable programming. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Frank H Clarke. Generalized gradients and applications. *Transactions of the American Mathematical Society*, 205:247–262, 1975.

Frank H Clarke, Yuri S Ledyaev, Ronald J Stern, and Peter R Wolenski. *Nonsmooth analysis and control theory*, volume 178. Springer Science & Business Media, 2008.

Mark Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 1996.

Chris Cremer, Quaid Morris, and David Duvenaud. Reinterpreting importance-weighted autoencoders. In *International Conference on Learning Representations (ICLR) Workshop*, 2017.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1): 119–154, 2020.

Thomas L. Dean and Mark S. Boddy. An analysis of time-dependent planning. In *AAAI Conference on Artificial Intelligence*, 1988.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Conference on Knowledge Discovery and Data Mining (KDD)*, 2014.

Mark Drummond and John Bresina. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *AAAI Conference on Artificial Intelligence*, 1990.

Saher Esmeir and Shaul Markovitch. Anytime learning of anycost classifiers. *Machine Learning*, 82:445–473, 2010.

Charles Fefferman and Scott Markel. Recovering a feed-forward net from its output. In *Conference on Neural Information Processing Systems (NeurIPS)*, 1994.

Björn Fröhlich, Erik Rodner, and Joachim Denzler. As time goes by – anytime semantic segmentation with iterative context forests. In *German Conference on Pattern Recognition (GCPR)*, 2012.

Tianshi Gao and Daphne Koller. Active classification based on value of classifier. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2011.

Krzysztof J. Geras, Abdel-Rahman Mohamed, Rich Caruana, Gregor Urban, Shengjie Wang, Ozlem Aslan, Matthai Philipose, Matthew Richardson, and Charles Sutton. Blending LSTMs into CNNs. In *International Conference on Learning Representations (ICLR) Workshop*, 2016.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

Prasoon Goyal, Zhiting Hu, Xiaodan Liang, Chenyu Wang, and Eric Xing. Nonparametric variational auto-encoders for hierarchical representation learning. In *International Conference on Computer Vision (ICCV)*, 2017.

Alex Grubb and Drew Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2012.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learing (ICML)*, 2018a.

Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018b.

S-P Han and Olvi L Mangasarian. Exact penalty functions in nonlinear programming. *Mathematical programming*, 17(1):251–269, 1979.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learing (ICML)*, 2019.

M. Hardt and T. Ma. Identity matters in deep learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer, 2008.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier, 1990.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Deep Learning Workshop at NeurIPS*, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Uncertainty in Artificial Intelligence (UAI)*, 1987.

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

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In *International Conference on Machine Learing (ICML)*, 2017.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations (ICLR)*, 2018.

F. Huszár. Is maximum likelihood useful for representation learning?, 2017. http://www.inference.vc/maximum-likelihood-for-representation-learning-2/, visited 2017-10-27.

Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations (ICLR)*, 2019a.

Ziwei Ji and Matus Telgarsky. The implicit bias of gradient descent on nonseparable data. In *Conference on Computational Learning Theory (COLT)*, 2019b.

Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

K. Kawaguchi. Deep learning without poor local minima. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017.

Jaehong Kim, Sungeun Hong, Yongseok Choi, and Jiwon Kim. Doubly nested network for resource-efficient inference. In *arXiv:1806.07568*, 2018.

Akisato Kimura, Zoubin Ghahramani, Koh Takeuchi, Tomoharu Iwata, and Naonori Ueda. Few-shot learning of neural networks from scratch by pseudo example optimization. In *British Machine Vision Conference (BMVC)*, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Věra Kůrková and Paul C. Kainen. Functionally equivalent feedforward neural networks. *Neural Computation*, 6(3):543–558, 1994.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. FractalNet: Ultra-deep neural networks without residuals. In *International Conference on Learning Representations (ICLR)*, 2017.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2015.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. Learning small-size DNN with output-distribution-based criteria. In *Conference of the International Speech Communication Association (Interspeech)*, 2014.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision (ECCV)*, 2016.

Percy Liang, Hal Daumé III, and Dan Klein. Structure compilation: trading structure for features. In *International Conference on Machine Learing (ICML)*, 2008.

Maxim Likhachev, Geoffrey J. Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2004.

Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Ji Liu and Xiaojin Zhu. The teaching dimension of linear learners. *Journal of Machine Learning Research (JMLR)*, 17(1):5631–5655, 2016.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations (ICLR)*, 2016.

Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Hartmut Maennel, Olivier Bousquet, and Sylvain Gelly. Gradient descent quantizes ReLU network features. In *arXiv:1803.08367*, 2018.

Mor Shpigel Nacson, Jason Lee, Suriya Gunasekar, Pedro Henrique Pamplona Savarese, Nathan Srebro, and Daniel Soudry. Convergence of gradient descent on separable data. In *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2019.

Behnam Neyshabur, Ruslan R. Salakhutdinov, and Nati Srebro. Path-SGD: Path-normalized optimization in deep neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations (ICLR)*, 2018.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2016.

Philipp Petersen, Mones Raslan, and Felix Voigtlaender. Topological properties of the set of functions generated by neural networks of fixed size, 2020.

Mary Phuong and Christoph H. Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learing (ICML)*, 2019a.

Mary Phuong and Christoph H. Lampert. Distillation-based training for multi-exit architectures. In *International Conference on Computer Vision (ICCV)*, 2019b.

Mary Phuong and Christoph H. Lampert. Functional vs. parametric equivalence of ReLU networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Mary Phuong and Christoph H. Lampert. The inductive bias of ReLU networks on orthogonally separable data. In *International Conference on Learning Representations (ICLR)*, 2021.

Mary Phuong, Max Welling, Nate Kushman, Ryota Tomioka, and Sebastian Nowozin. The mutual autoencoder: Controlling information in latent code representations. 2018.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *International Conference on Learning Representations (ICLR)*, 2018.

Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learing (ICML)*, 2017.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learing (ICML)*, pages 1278–1286, 2014.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations (ICLR)*, 2016.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Bernhard Schölkopf and Alexander J. Smola. *Learning With Kernels*. MIT Press, 2002.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11), 1997.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learing (ICML)*, 2018.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Jan Šochman and Jiří Matas. Waldboost – learning for time constrained sequential detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research (JMLR)*, 19(1):2822–2878, 2018.

Pierre Stock, Benjamin Graham, Rémi Gribonval, and Hervé Jégou. Equi-normalization of neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Zhiyuan Tang, Dong Wang, and Zhiyong Zhang. Recurrent neural network training with dark knowledge transfer. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *International Conference on Pattern Recognition (ICPR)*, 2016.

Kirill Trapeznikov and Venkatesh Saligrama. Supervised sequential classification under budget constraints. In *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2013.

Ken Ueno, Xiaopeng Xi, Eamonn Keogh, and Dah-Jye Lee. Anytime classification using the nearest neighbor algorithm with applications to stream mining. In *IEEE International Conference on Data Mining (ICDM)*, 2006.

Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *Journal of Machine Learning Research (JMLR)*, 16(2):2023–2049, 2015.

Vladimir Naumovich Vapnik. *Statistical learning theory*. John Wiley & Sons, 1998.

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision (IJCV)*, 57(2):137–154, 2004.

Xin Wang, Fisher Yu, Zi-Yi Dou, and Joseph E. Gonzalez. SkipNet: Learning dynamic routing in convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. BlockDrop: Dynamic inference paths in residual networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *International Conference on Machine Learing (ICML)*, 2017.

Serena Yeung, Anitha Kannan, Yann Dauphin, and Li Fei-Fei. Tackling over-pruning in variational autoencoders. In *International Conference on Machine Learing (ICML) Workshop*, 2017.

Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation. In *International Conference on Computer Vision (ICCV)*, 2017.

Willard I. Zangwill. Non-linear programming via penalty functions. *Management Science*, 13(5):344–358, 1967.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph HyperNetworks for neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Information maximizing variational autoencoders. In *AAAI Conference on Artificial Intelligence*, 2019.

Jerry Zhu. Machine teaching for Bayesian learners in the exponential family. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2013.

Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI Conference on Artificial Intelligence*, 2015.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73, 1996.

APPENDIX A

Proofs for Chapter 2

Use the following look-up table to find a particular lemma or theorem and its proof.

Lemma 2.1	→	Lemma A.15
Lemma 2.2	→	Corollary A.1
Lemma 2.3	→	Lemma A.18
Lemma 2.4	→	Lemma A.17
Theorem 2.1	→	Theorem A.1

A.1 Piece-wise linear surfaces

Definition A.1 (Partition)**.** *Let $\mathcal{S} \subseteq \mathcal{X}$. We define the* partition of $\mathcal{X}$ induced by $\mathcal{S}$*, denoted $\mathcal{P}_{\mathcal{X}}(\mathcal{S})$, as the set of connected components of $\mathcal{X} \setminus \mathcal{S}$.*

Definition A.2 (Piece-wise hyperplane)**.** *Let $\mathcal{P}$ be a partition of $\mathcal{X}$. We say $\mathcal{H} \subseteq \mathcal{X}$ is a* piece-wise hyperplane with respect to partition $\mathcal{P}$*, if $\mathcal{H} \neq \emptyset$ and there exist $(\mathbf{w}, b) \neq (\mathbf{0}, 0)$ and $P \in \mathcal{P}$ such that $\mathcal{H} = \{\mathbf{x} \in P \,|\, \mathbf{w}^\intercal \mathbf{x} + b = 0\}$.*

Definition A.3 (Piece-wise linear surface / pwl. surface)**.** *A set $\mathcal{S} \subseteq \mathcal{X}$ is called a* piece-wise linear surface on $\mathcal{X}$ of order κ *if it can be written as $\mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l$, where each $\mathcal{H}_i^l$ is a piece-wise hyperplane with respect to $\mathcal{P}_{\mathcal{X}}(\bigcup_{k \in [l-1], j \in [n_k]} \mathcal{H}_j^k)$, and no number smaller than κ admits such a representation.*

Lemma A.1. *If $\mathcal{S}_1, \mathcal{S}_2$ are piece-wise linear surfaces on $\mathcal{X}$ of order k_1 and k_2, then $\mathcal{S}_1 \cup \mathcal{S}_2$ is a piece-wise linear surface on $\mathcal{X}$ of order at most $\max\{k_1, k_2\}$.*

Proof. Let $\mathcal{S}_1 = \bigcup_{l \in [k_1], i \in [n_l]} \mathcal{H}_i^l$ and $\mathcal{S}_2 = \bigcup_{l \in [k_2], i \in [m_l]} \mathcal{G}_i^l$ be the pwl. surface representations of $\mathcal{S}_1, \mathcal{S}_2$. Given $\mathcal{H}_i^l$, consider the partition

$$\mathcal{P} := \mathcal{P}_{\mathcal{X}}\left(\bigcup_{k \in [l-1], j \in [n_k]} \mathcal{H}_j^k \cup \bigcup_{k \in [\max\{l-1, k_2\}], j \in [m_k]} \mathcal{G}_j^k \right). \tag{A.1}$$

We can write $\mathcal{H}_i^l = \bigcup_{P \in \mathcal{P}} \mathcal{H}_i^l \cap P$ and denote the nonempty intersections $\mathcal{H}_i^l \cap P$ as $\left\{\bar{\mathcal{H}}_{i,j}^l\right\}_j$. Similarly, we decompose $\mathcal{G}_i^l = \bigcup_j \bar{\mathcal{G}}_{i,j}^l$. Then $\mathcal{S}_1 \cup \mathcal{S}_2 = \bigcup_{l \in [\max\{k_1, k_2\}]} (\bigcup_{i,j} \bar{\mathcal{H}}_{i,j}^l \cup \bigcup_{i,j} \bar{\mathcal{G}}_{i,j}^l)$, where each $\bar{\mathcal{H}}_{i,j}^l$ and $\bar{\mathcal{G}}_{i,j}^l$ is a piece-wise hyperplane wrt. $\mathcal{P} = \mathcal{P}_{\mathcal{X}}(\bigcup_{k \in [l-1]} (\bigcup_{i',j'} \bar{\mathcal{H}}_{i',j'}^k \cup \bigcup_{i',j'} \bar{\mathcal{G}}_{i',j'}^k))$. □

Given sets $\mathcal{X}$ and $\mathcal{S} \subseteq \mathcal{X}$, we introduce the notation

$$\square_i \mathcal{S} := \bigcup \{\mathcal{S}' \subseteq \mathcal{S} \mid \mathcal{S}' \text{ is a pwl. surface on } \mathcal{X} \text{ of order at most } i\}. \tag{A.2}$$

(The dependence on $\mathcal{X}$ is suppressed.) By Lemma A.1, $\square_i \mathcal{S}$ is itself a pwl. surface on $\mathcal{X}$ of order at most i.

Lemma A.2. *For $i \leq j$ and any set $\mathcal{S}$, we have $\square_i \square_j \mathcal{S} = \square_j \square_i \mathcal{S} = \square_i \mathcal{S}$.*

Proof. We will need these definitions:

$$\square_i \mathcal{S} = \bigcup \{\mathcal{S}'' \subseteq \mathcal{S} \mid \mathcal{S}'' \text{ is a pwl. surface of order at most } i\}, \tag{A.3}$$

$$\square_j \mathcal{S} = \bigcup \{\mathcal{S}'' \subseteq \mathcal{S} \mid \mathcal{S}'' \text{ is a pwl. surface of order at most } j\}, \tag{A.4}$$

$$\square_i \square_j \mathcal{S} = \bigcup \{\mathcal{S}'' \subseteq \square_j \mathcal{S} \mid \mathcal{S}'' \text{ is a pwl. surface of order at most } i\}, \tag{A.5}$$

$$\square_j \square_i \mathcal{S} = \bigcup \{\mathcal{S}'' \subseteq \square_i \mathcal{S} \mid \mathcal{S}'' \text{ is a pwl. surface of order at most } j\}. \tag{A.6}$$

Consider first the equality $\square_j \square_i \mathcal{S} = \square_i \mathcal{S}$. We know that $\square_j \square_i \mathcal{S} \subseteq \square_i \mathcal{S}$ because the square operator always yields a subset. At the same time, $\square_i \mathcal{S} \subseteq \square_j \square_i \mathcal{S}$, because $\square_i \mathcal{S}$ satisfies the condition for membership in (A.6).

To prove the equality $\square_i \square_j \mathcal{S} = \square_i \mathcal{S}$, we use the inclusion $\square_j \mathcal{S} \subseteq \mathcal{S}$ to deduce $\square_i \square_j \mathcal{S} \subseteq \square_i \mathcal{S}$. Now let $\mathcal{S}'' \subseteq \mathcal{S}$ be one of the sets under the union in (A.3), i.e. it is a pwl. surface of order at most i. Then it is also a pwl. surface of order at most j, implying $\mathcal{S}'' \subseteq \square_j \mathcal{S}$. This means $\mathcal{S}''$ is also one of the sets under the union in (A.5), proving that $\square_i \mathcal{S} \subseteq \square_i \square_j \mathcal{S}$. □

Lemma A.3. *Let $\mathcal{X}$ and $\mathcal{S} \subseteq \mathcal{X}$ be sets. Then one can write $\square_{k+1} \mathcal{S} = \square_k \mathcal{S} \cup \bigcup_i \mathcal{H}_i$ where $\mathcal{H}_i$ are piece-wise hyperplanes wrt. $\mathcal{P}_{\mathcal{X}}(\square_k \mathcal{S})$.*

Proof. Let $\square_{k+1} \mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l$ be the pwl. surface representation of $\square_{k+1} \mathcal{S}$. If $\kappa \leq k$, then $\square_{k+1} \mathcal{S} = \square_k \mathcal{S}$ and we are done. Otherwise, $\bigcup_{l \in [k], i \in [n_k]} \mathcal{H}_i^l \subseteq \square_k \mathcal{S}$, implying

$$\square_{k+1} \mathcal{S} \subseteq \square_k \mathcal{S} \cup \bigcup_{i \in [n_{k+1}]} \mathcal{H}_i^{k+1}. \tag{A.7}$$

At the same time, $\square_k \mathcal{S} \cup \bigcup_{i \in [n_{k+1}]} \mathcal{H}_i^{k+1}$ is a pwl. surface of order at most $k+1$ because $\square_k \mathcal{S}$ is a pwl. surface of order at most k and $\mathcal{H}_i^{k+1}$ can be decomposed into piece-wise hyperplanes wrt. $\mathcal{P}_{\mathcal{X}}(\square_k \mathcal{S})$. Therefore, $\square_k \mathcal{S} \cup \bigcup_{i \in [n_{k+1}]} \mathcal{H}_i^{k+1} \subseteq \square_{k+1} \mathcal{S}$, implying in fact equality. □

Definition A.4 (Canonical representation of a pwl. surface)**.** *Let $\mathcal{S}$ be a pwl. surface on $\mathcal{X}$. The pwl. surface representation $\mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l$ is called* canonical *if $\bigcup_{l \in [k], i \in [n_l]} \mathcal{H}_i^l = \square_k \mathcal{S}$ for all $k \in [\kappa]$, and each $\mathcal{H}_i^l$ is distinct.*

Lemma A.4. *If $\mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l$ is a pwl. surface in canonical form, then κ is the order of $\mathcal{S}$.*

Proof. Denote the order of $\mathcal{S}$ by λ. By the definition of order, $\lambda \leq \kappa$, and $\mathcal{S} = \square_\lambda \mathcal{S}$. Then, since $\mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l$ is a canonical representation, we have

$$\bigcup_{l \in [\lambda], i \in [n_l]} \mathcal{H}_i^l = \square_\lambda \mathcal{S} = \mathcal{S} = \bigcup_{l \in [\kappa], i \in [n_l]} \mathcal{H}_i^l. \tag{A.8}$$

It follows that $\kappa = \lambda$. □

Lemma A.5. *Every pwl. surface has a canonical representation.*

Proof. The inclusion $\bigcup_{l\in[k],i\in[n_l]} \mathcal{H}_i^l \subseteq \square_k \mathcal{S}$ holds for any representation. We will show the other inclusion by induction in the order of $\mathcal{S}$. If $\mathcal{S}$ is order one, $\square_1 \mathcal{S} \subseteq \mathcal{S} = \bigcup_{i\in[n_1]} \mathcal{H}_i^1$ holds for any representation and we are done. Now assume the lemma holds up to order $\kappa - 1$, and let $\mathcal{S}$ be order κ. Then by Lemma A.3, $\mathcal{S} = \square_\kappa \mathcal{S} = \square_{\kappa-1}\mathcal{S} \cup \bigcup_i \mathcal{H}_i^\kappa$, where $\mathcal{H}_i^\kappa$ are piece-wise hyperplanes wrt. $\mathcal{P}_\mathcal{X}(\square_{\kappa-1}\mathcal{S})$. By the inductive assumption, $\square_{\kappa-1}\mathcal{S}$ has a canonical representation, say $\square_{\kappa-1}\mathcal{S} = \bigcup_{l\in[\kappa-1],i\in[n_l]} \mathcal{H}_i^l$. We claim that $\mathcal{S} = \bigcup_{l\in[\kappa],i\in[n_l]} \mathcal{H}_i^l$ is a canonical representation of $\mathcal{S}$. If $k = \kappa$, then clearly $\square_k \mathcal{S} \subseteq \mathcal{S} = \bigcup_{l\in[\kappa],i\in[n_l]} \mathcal{H}_i^l$. If $k \in [\kappa - 1]$, then by Lemma A.2, $\square_k \mathcal{S} = \square_k \square_{\kappa-1}\mathcal{S} = \bigcup_{l\in[k],i\in[n_l]} \mathcal{H}_i^l$, where we have used the canonical representation of $\square_{\kappa-1}\mathcal{S}$.

Finally, distinctness of $\mathcal{H}_i^l$ can be ensured by throwing away duplicates. □

Lemma A.6. *Let $\mathcal{X}$ be an open set. If $\mathcal{S}$ is a piece-wise linear surface on $\mathcal{X}$, and if $\mathcal{S} = \bigcup_{l\in[\kappa],i\in[n_l]} \mathcal{H}_i^l$ and $\mathcal{S} = \bigcup_{k\in[\kappa],j\in[m_k]} \mathcal{G}_j^k$ are two canonical representations of $\mathcal{S}$, then for all $l \in [\kappa]$, $n_l = m_l$ and there exists a permutation $\pi : [n_l] \to [n_l]$ such that $\mathcal{H}_i^l = \mathcal{G}_{\pi(i)}^l$. In other words, the canonical representation is unique up to within-order indexing.*

Proof. Let $k \in [\kappa]$. Because both representations are canonical, we have

$$\square_{k-1}\mathcal{S} \cup \bigcup_{i\in[n_k]} \mathcal{H}_i^k = \square_k \mathcal{S} = \square_{k-1}\mathcal{S} \cup \bigcup_{j\in[m_k]} \mathcal{G}_j^k, \tag{A.9}$$

where $\mathcal{H}_i^k$ and $\mathcal{G}_j^k$ are piece-wise hyperplanes wrt. $\mathcal{P}_\mathcal{X}(\square_{k-1}\mathcal{S})$. Then for each $P \in \mathcal{P}_\mathcal{X}(\square_{k-1}\mathcal{S})$,

$$P \cap \bigcup_{i\in[n_k]} \mathcal{H}_i^k = P \cap \bigcup_{j\in[m_k]} \mathcal{G}_j^k, \tag{A.10}$$

where on both sides above we have a union of hyperplanes on an open set. The claim follows. □

Definition A.5 (Dependency graph of a pwl. surface)**.** *Let $\mathcal{S}$ be a piece-wise linear surface on $\mathcal{X}$, and let $\mathcal{S} = \bigcup_{l\in[\kappa],i\in[n_l]} \mathcal{H}_i^l$ be its canonical representation. We define the* dependency graph of $\mathcal{S}$ *as the directed graph that has the piece-wise hyperplanes $\left\{\mathcal{H}_i^l\right\}_{l,i}$ as vertices, and has an edge $\mathcal{H}_i^l \to \mathcal{H}_j^k$ iff $l < k$ and $\mathcal{H}_i^l \cap \operatorname{cl} \mathcal{H}_j^k \neq \emptyset$.*

A.2 ReLU networks and folds

We denote by ρ the ReLU function: $\rho(\mathbf{u})_i = \max\{0, u_i\}$ for $i \in [\dim(\mathbf{u})]$.

Definition A.6 (ReLU network)**.** *Let $\mathcal{X} \subseteq \mathbb{R}^{d_0}$ with $d_0 \geq 2$ be a nonempty open set, and let $\boldsymbol{\theta} \triangleq (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, b_L)$ be the network's parameters, with $\mathbf{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$, $\mathbf{b}_l \in \mathbb{R}^{d_l}$, and $d_L = 1$. A* ReLU network parameterised by $\boldsymbol{\theta}$ *is the function $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}$, defined by*

$$h_{\boldsymbol{\theta}} \triangleq h_{\boldsymbol{\theta}}^L \circ \rho \circ h_{\boldsymbol{\theta}}^{L-1} \circ \cdots \circ \rho \circ h_{\boldsymbol{\theta}}^1, \tag{A.11}$$

where $h_{\boldsymbol{\theta}}^l(\mathbf{x}) = \mathbf{W}_l \cdot \mathbf{x} + \mathbf{b}_l$. For $1 \leq l \leq k \leq L$, we also denote

$$h_{\boldsymbol{\theta}}^{l:k} \triangleq h_{\boldsymbol{\theta}}^k \circ \rho \circ h_{\boldsymbol{\theta}}^{k-1} \cdots \circ \rho \circ h_{\boldsymbol{\theta}}^l, \tag{A.12}$$

$$\check{h}_{\boldsymbol{\theta}}^{l:k} \triangleq \rho \circ h_{\boldsymbol{\theta}}^{l:k}. \tag{A.13}$$

For a ReLU network $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}$ and $l \in [L-1]$, denote $\mathcal{X}_{\boldsymbol{\theta}}^l \triangleq \left\{\check{h}_{\boldsymbol{\theta}}^{1:l}(\mathbf{x}) \,|\, \mathbf{x} \in \mathcal{X}\right\}$. Also, for convenience, define $\mathcal{X}_{\boldsymbol{\theta}}^0 \triangleq \mathcal{X}$. (We will omit the subscript $\boldsymbol{\theta}$ when it is clear from the context.) We write $f|_{\mathcal{A}}$ to denote the restriction of the function f to the domain $\mathcal{A}$.

Definition A.7 (Activation indicator)**.** *A tuple* $\mathbf{I} \triangleq (\mathbf{I}_1, \ldots, \mathbf{I}_{L-1})$ *is called an* activation indicator *if* $\mathbf{I}_l = \operatorname{diag}(\mathbf{i}_l) \in \mathbb{R}^{d_l \times d_l}$ *and* $\mathbf{i}_l \in \{0,1\}^{d_l}$ *for* $l \in [L-1]$*. It is called* non-trivial *if* $\mathbf{i}_l \neq \mathbf{0}$ *for all* $l \in [L-1]$ *and* non-trivial up to k *if* $\mathbf{i}_l \neq \mathbf{0}$ *for all* $l \in [k]$*.*

Given a parameter vector $\boldsymbol{\theta} \triangleq (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, \mathbf{b}_L)$ and an activation indicator $\mathbf{I}$, we introduce the notation

$$\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}) \triangleq \mathbf{e}_i^\intercal \mathbf{W}_l \mathbf{I}_{l-1} \mathbf{W}_{l-1} \cdots \mathbf{I}_1 \mathbf{W}_1, \tag{A.14}$$

$$b_i^l(\boldsymbol{\theta}, \mathbf{I}) \triangleq \mathbf{e}_i^\intercal \sum_{k=1}^{l} \mathbf{W}_l \mathbf{I}_{l-1} \cdots \mathbf{W}_{k+1} \mathbf{I}_k \mathbf{b}_k. \tag{A.15}$$

(We will omit the argument $\boldsymbol{\theta}$ when it is clear from the context.) These quantities characterise the different linear pieces implemented by the network's units. Also define $\mathbf{I}^{\boldsymbol{\theta}}(\mathbf{x}) \triangleq (\mathbf{I}_1^{\boldsymbol{\theta}}(\mathbf{x}), \ldots, \mathbf{I}_{L-1}^{\boldsymbol{\theta}}(\mathbf{x}))$ as the activation indicator for a specific input: $I_l^{\boldsymbol{\theta}}(\mathbf{x})[i,i] \triangleq \mathbb{1}\left\{h_i^{1:l}(\mathbf{x}) \geq 0\right\}$ for all (l,i).

Lemma A.7. *In a ReLU network with parameters* $\boldsymbol{\theta} = (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, b_L)$*, the pre-activations satisfy* $h_i^{1:l}(\mathbf{x}) \in \left\{\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}) \cdot \mathbf{x} + b_i^l(\boldsymbol{\theta}, \mathbf{I})\right\}_{\mathbf{I}}$*, where the indexing runs over all possible activation indicators* $\mathbf{I}$*. More precisely,* $h_i^{1:l}(\mathbf{x}) = \mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}^{\boldsymbol{\theta}}(\mathbf{x})) \cdot \mathbf{x} + b_i^l(\boldsymbol{\theta}, \mathbf{I}^{\boldsymbol{\theta}}(\mathbf{x}))$*.*

Proof. Left as exercise. □

Definition A.8 (Fold-set)**.** *Let* $\mathcal{X}$ *be an open set, and* $f : \mathcal{X} \to \mathbb{R}$ *a continuous, piece-wise linear function. We define the* fold-set of f*, denoted by* $\mathcal{F}(f)$*, as the set of all points at which* f *is non-differentiable.*

Definition A.9 (Positive / negative in a neighbourhood)**.** *Let* $\mathcal{X}$ *be an open set. The function* $f : \mathcal{X} \to \mathbb{R}$ *is* positive (negative) in the neighbourhood of $\mathbf{x} \in \mathcal{X}$ *if for any* $\epsilon > 0$ *there exists* $\mathbf{x}' \in \mathcal{B}_\epsilon(\mathbf{x})$ *such that* $f(\mathbf{x}') > 0$ *(*$f(\mathbf{x}') < 0$*).*

Definition A.10 (Unit fold-set)**.** *Let* $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}$ *be a ReLU network. We define the* unit (l,i) fold-set of $h_{\boldsymbol{\theta}}$*, denoted* $\mathcal{F}_i^l(h_{\boldsymbol{\theta}})$*, as the set of all* $\mathbf{x} \in \mathcal{X}$ *where* $h_{\boldsymbol{\theta}}^{1:l}[i](\mathbf{x}) = 0$ *and* $h_{\boldsymbol{\theta}}^{1:l}[i]$ *is positive in the neighbourhood of* $\mathbf{x}$*.*

Lemma A.8. *Let* $\mathcal{X}$ *be an open set, and* $f : \mathcal{X} \to \mathbb{R}$ *a continuous piece-wise linear function. Then* $\mathcal{F}(\rho \circ f)$ *consists of those* $\mathbf{x} \in \mathcal{X}$ *that satisfy*

- $f(\mathbf{x}) > 0$ *and* $\mathbf{x} \in \mathcal{F}(f)$*, or*
- $f(\mathbf{x}) = 0$ *and* f *is positive in the neighbourhood of* $\mathbf{x}$*.*

Proof. We will prove that if $\mathbf{x}$ satisfies any of the two conditions, then $\mathbf{x} \in \mathcal{F}(\rho \circ f)$, and if it violates both, then $\mathbf{x} \in \mathcal{F}(\rho \circ f)^{\mathsf{c}}$. We begin with the latter implication.

Let $\mathbf{x}$ be such that $f(\mathbf{x}) > 0$ and $\mathbf{x} \notin \mathcal{F}(f)$, i.e. f is differentiable at $\mathbf{x}$. Since f is piece-wise linear, there exists $\epsilon > 0$ such that all of $\mathcal{B}_\epsilon(\mathbf{x})$ lies inside a single linear region of f and

$f(\mathcal{B}_\epsilon(\mathbf{x})) \subseteq (0, \infty]$. Then, on $\mathcal{B}_\epsilon(\mathbf{x})$, the ReLU behaves like an identity, implying $\rho \circ f$ is differentiable at $\mathbf{x}$, proving that $\mathbf{x} \in \mathcal{F}(\rho \circ f)^{\mathsf{c}}$. Next, consider $\mathbf{x}$ such that $f(\mathbf{x}) = 0$. For it to violate the second condition, there must exist a ball $\mathcal{B}_\epsilon(\mathbf{x})$ around $\mathbf{x}$ such that $f(\mathcal{B}_\epsilon(\mathbf{x})) \subseteq (-\infty, 0]$. (This is also true if $f(\mathbf{x}) < 0$.) Then, on $\mathcal{B}_\epsilon(\mathbf{x})$, the ReLU behaves like a constant zero, implying that $\rho \circ f$ is differentiable at $\mathbf{x}$.

We now prove the other implication. If $f(\mathbf{x}) > 0$ and $\mathbf{x} \in \mathcal{F}(f)$, then there exists $\epsilon > 0$ such that $f(\mathcal{B}_\epsilon(\mathbf{x})) \subseteq (0, \infty]$, which guarantees that the ReLU behaves like an identity on $\mathcal{B}_\epsilon(\mathbf{x})$. In this ball, we have $\rho \circ f = f$, so $\mathbf{x} \in \mathcal{F}(\rho \circ f)$.

If $f(\mathbf{x}) = 0$ and f is positive in the neighbourhood of $\mathbf{x}$, we distinguish several cases. If $\mathbf{x} \notin \mathcal{F}(f)$, then there exists a ball $\mathcal{B}_\delta(\mathbf{x})$ on which f behaves linearly, i.e. $\rho(f(\mathbf{x})) = \rho(\mathbf{w}^\intercal \mathbf{x} + b)$, implying $\mathbf{x} \in \mathcal{F}(\rho \circ f)$. If $\mathbf{x} \in \mathcal{F}(f)$ and, in addition, there exists a ball $\mathcal{B}_\delta(\mathbf{x})$ such that $f(\mathcal{B}_\delta(\mathbf{x})) \subseteq [0, \infty)$, then the ReLU behaves like an identity on $\mathcal{B}_\delta(\mathbf{x})$ and $\mathbf{x} \in \mathcal{F}(\rho \circ f)$. The final case is $\mathbf{x} \in \mathcal{F}(f)$ such that f attains both positive and negative values in its neighbourhood. Since f is piece-wise linear, there exist $\mathbf{p}, \mathbf{n}$ such that $f(\mathbf{x} + \epsilon\mathbf{n}) < 0 < f(\mathbf{x} + \epsilon\mathbf{p})$, and $\mathbf{x} + \epsilon\mathbf{p}, \mathbf{x} + \epsilon\mathbf{n} \notin \mathcal{F}(f)$ for all $\epsilon \in (0, 1]$. Then $\nabla(\rho \circ f)(\mathbf{x} + \epsilon\mathbf{p}) \neq \mathbf{0}$ and $\nabla(\rho \circ f)(\mathbf{x} + \epsilon\mathbf{n}) = \mathbf{0}$, yielding $\mathbf{x} \in \mathcal{F}(\rho \circ f)$. □

Lemma A.9. *Let $\mathcal{X}$ be an open set, and let $f_1, \ldots, f_n : \mathcal{X} \to \mathbb{R}$ be continuous, piece-wise linear functions. For any $w_1, \ldots, w_n \in \mathbb{R}$, define $f = \sum_{i=1}^n w_i f_i$. Then $\mathcal{F}(f) \subseteq \bigcup_{i=1}^n \mathcal{F}(f_i)$.*

Proof. Left as exercise. □

A.3 General and transparent ReLU networks

Lemma A.10. *For all θ except a closed zero-measure set,*

$$\operatorname{rank}\left(\mathbf{W}_l \mathbf{I}_{l-1} \cdots \mathbf{I}_k \mathbf{W}_k\right) = \min\left\{d_{k-1}, \operatorname{rank}(\mathbf{I}_k), \ldots, \operatorname{rank}(\mathbf{I}_{l-1}), d_l\right\}, \tag{A.16}$$

$$\operatorname{rank}\left(\mathbf{I}_l \mathbf{W}_l \mathbf{I}_{l-1} \cdots \mathbf{I}_k \mathbf{W}_k\right) = \min\left\{d_{k-1}, \operatorname{rank}(\mathbf{I}_k), \ldots, \operatorname{rank}(\mathbf{I}_{l-1}), \operatorname{rank}(\mathbf{I}_l)\right\}, \tag{A.17}$$

for all activation indicators $\mathbf{I}$ and all $k \leq l$.

Proof. First, notice that (A.16) is just a special case of (A.17) with $\mathbf{I}_l$ equal to the identity matrix. It therefore suffices to prove (A.17).

To further simplify, we will prove the statement for a single fixed activation indicator $\mathbf{I}$. Then if $\boldsymbol{\Theta}(\mathbf{I})$ is the set of networks for which (A.17) holds given $\mathbf{I}$, and $\boldsymbol{\Theta}(\mathbf{I})$ contains all networks except a closed zero-measure set, then also $\bigcap_{\mathbf{I}} \boldsymbol{\Theta}(\mathbf{I})$ contains all networks except a closed zero-measure set, proving the lemma.

Let us hence fix $\mathbf{I}$, and let $k \in [L]$. We proceed by induction. For the initial step, notice that the matrix $\mathbf{I}_k \mathbf{W}_k$ is just $\mathbf{W}_k$ with some rows replaced by zeroes. The rank of such a matrix is the same as the matrix obtained by removing the zero rows, which has size $(\operatorname{rank}(\mathbf{I}_k), d_{k-1})$. For all $\mathbf{W}_k$ except a closed zero-measure set, this matrix has rank $\min\{d_{k-1}, \operatorname{rank}(\mathbf{I}_k)\}$.

For the inductive step, denote $\bar{\mathbf{W}}_i := \mathbf{I}_i \mathbf{W}_i \cdots \mathbf{I}_k \mathbf{W}_k$ and

$$r_i := \min\left\{d_{k-1}, \operatorname{rank}(\mathbf{I}_k), \ldots, \operatorname{rank}(\mathbf{I}_i)\right\}. \tag{A.18}$$

We assume that $\operatorname{rank}(\bar{\mathbf{W}}_{i-1}) = r_{i-1}$ and want to prove the same for i. Notice that for all $\mathbf{W}_i$ except a closed zero-measure set, any r_i rows of $\mathbf{W}_i$ are linearly independent and

their span intersects with $\ker(\bar{\mathbf{W}}_{i-1}^\intercal)$ only at $\mathbf{0}$. To see this, recall that by the inductive assumption, $\operatorname{rank}(\bar{\mathbf{W}}_{i-1}^\intercal) = r_{i-1}$, so $\ker(\bar{\mathbf{W}}_{i-1}^\intercal)$ has dimension $d_{i-1} - r_{i-1}$. We can concatenate any r_i-subset of rows of $\mathbf{W}_i$ to the basis of $\ker(\bar{\mathbf{W}}_{i-1}^\intercal)$ to obtain a matrix of size $(r_i + d_{i-1} - r_{i-1}, d_{i-1})$, which is a wide matrix, because $r_i \leq r_{i-1}$. Hence, its rows are linearly independent for all $\mathbf{W}_i$ except a closed zero-measure set.

We now prove that $\operatorname{rank}(\mathbf{I}_i\mathbf{W}_i\bar{\mathbf{W}}_{i-1}) = \min\left\{\operatorname{rank}(\bar{\mathbf{W}}_{i-1}), \operatorname{rank}(\mathbf{I}_i)\right\} \triangleq r_i$. The "$\leq$" direction is immediate. For the "$\geq$" direction, we distinguish between two cases. If $\operatorname{rank}(\mathbf{I}_i) \leq \operatorname{rank}(\bar{\mathbf{W}}_{i-1})$, let $\mathbf{v}_1, \dots \mathbf{v}_{r_i}$ be the (linearly independent) nonzero rows of $\mathbf{I}_i\mathbf{W}_i$. We want to show that $\left\{\mathbf{v}_j^\intercal\bar{\mathbf{W}}_{i-1}\right\}_j$ are linearly independent, i.e. that $\mathbf{I}_i\mathbf{W}_i\bar{\mathbf{W}}_{i-1}$ has at least r_i linearly independent rows. If $\sum_{j=1}^{r_i}\lambda_j\mathbf{v}_j^\intercal\bar{\mathbf{W}}_{i-1} = \mathbf{0}$, then $\sum_{j=1}^{r_i}\lambda_j\mathbf{v}_j \in \ker(\bar{\mathbf{W}}_{i-1}^\intercal)$, which by assumption implies $\sum\lambda_j\mathbf{v}_j = \mathbf{0}$. By the independence of $\{\mathbf{v}_j\}$, we obtain $\lambda_j = 0$, i.e. $\left\{\mathbf{v}_j^\intercal\bar{\mathbf{W}}_{i-1}\right\}_j$ are linearly independent, and $\operatorname{rank}(\mathbf{I}_i\mathbf{W}_i\bar{\mathbf{W}}_{i-1}) = r_i$.

If $\operatorname{rank}(\mathbf{I}_i) > \operatorname{rank}(\bar{\mathbf{W}}_{i-1})$, we can reduce the problem to the case $\operatorname{rank}(\mathbf{I}_i) \leq \operatorname{rank}(\bar{\mathbf{W}}_{i-1})$ by observing that $\operatorname{rank}(\mathbf{I}_i\mathbf{W}_i\bar{\mathbf{W}}_{i-1}) \geq \operatorname{rank}(\mathbf{J}_i\mathbf{W}_i\bar{\mathbf{W}}_{i-1})$ if $\mathbf{J}_i$ equals $\mathbf{I}_i$ only with some 1's replaced by 0's. We can thus take any such $\mathbf{J}_i$ and apply the argument from the previous paragraph to obtain $\operatorname{rank}(\mathbf{I}_i\mathbf{W}_i\bar{\mathbf{W}}_{i-1}) \geq \operatorname{rank}(\mathbf{J}_i\mathbf{W}_i\bar{\mathbf{W}}_{i-1}) \geq r_i$. □

Lemma A.11. *For all $\boldsymbol{\theta}$ except a closed zero-measure set, the following holds. Let (l, i), (k, j) be any units, let $\mathbf{I}$ be an activation indicator non-trivial up to $l-1$, and let $\mathbf{J}$ be an activation indicator non-trivial up to $k-1$, such that $(l, i, \mathbf{I}_{1:l-1}) \neq (k, j, \mathbf{J}_{1:k-1})$. Then, for all scalars $c \in \mathbb{R}$, it holds that $[\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I})] \neq c[\mathbf{w}_j^k(\boldsymbol{\theta}, \mathbf{J}), b_j^k(\boldsymbol{\theta}, \mathbf{J})]$.*

Proof. First, we exclude from consideration all configurations $\boldsymbol{\theta} = (\mathbf{W}_1, \mathbf{b}_1, \dots, \mathbf{W}_L, \mathbf{b}_L)$ such that $\mathbf{e}_i^\intercal\mathbf{W}_l\mathbf{I}_{l-1}\mathbf{W}_{l-1}\cdots\mathbf{I}_k\mathbf{W}_k\mathbf{e}_j = 0$ for some l, k, i, j, and some $\mathbf{I}$ non-trivial up to $l-1$. Since for any fixed $(l, k, i, j, \mathbf{I})$, the set of $\boldsymbol{\theta}$ satisfying the above is the set of roots of a non-trivial polynomial in $\boldsymbol{\theta}$, it is zero-measure and closed. Because there are only finitely many configurations of $(l, k, i, j, \mathbf{I})$, we have thus excluded a closed zero-measure set of parameters. We will denote its complement $\boldsymbol{\Theta}^*$.

From now on, we assume $\boldsymbol{\theta} \in \boldsymbol{\Theta}^*$. Notice that the case $c = 0$ of the lemma is thus automatically satisfied, since $\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}) \triangleq \mathbf{e}_i^\intercal\mathbf{W}_l\mathbf{I}_{l-1}\mathbf{W}_{l-1}\cdots\mathbf{I}_1\mathbf{W}_1 \neq \mathbf{0}$ by the definition of $\boldsymbol{\Theta}^*$. In the following, we can therefore assume $c \neq 0$ and treat $(l, i, \mathbf{I})$ and $(k, j, \mathbf{J})$ symmetrically.

Denote by $\boldsymbol{\Theta}^\neg \subseteq \boldsymbol{\Theta}^*$ the set of parameters $\boldsymbol{\theta}$ for which the lemma does *not* hold; we need to show that $\boldsymbol{\Theta}^\neg$ is closed and zero-measure. We start by showing the latter property by contradiction.

Suppose $\boldsymbol{\Theta}^\neg$ is positive-measure. We know that for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}^\neg$, there exist triples $(l, i, \mathbf{I}), (k, j, \mathbf{J})$ as stated in the lemma, and a scalar $c \in \mathbb{R}$ such that $[\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I})] = c[\mathbf{w}_j^k(\boldsymbol{\theta}, \mathbf{J}), b_j^k(\boldsymbol{\theta}, \mathbf{J})]$. Let $\mathcal{C}$ denote the set of all triplet-pairs $((l, i, \mathbf{I}), (k, j, \mathbf{J}))$ satisfying the conditions of the lemma; then the previous statement can be written as

$$\boldsymbol{\Theta}^\neg \subseteq \bigcup_{((l,i,\mathbf{I}),(k,j,\mathbf{J}))\in\mathcal{C}} \left\{\boldsymbol{\theta} \in \boldsymbol{\Theta}^* \mid \exists c \in \mathbb{R} : [\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I})] = c[\mathbf{w}_j^k(\boldsymbol{\theta}, \mathbf{J}), b_j^k(\boldsymbol{\theta}, \mathbf{J})]\right\}. \quad \text{(A.19)}$$

Since $\mathcal{C}$ is finite, there exist $((l, i, \mathbf{I}), (k, j, \mathbf{J})) \in \mathcal{C}$ for which the set under the union (call it $\boldsymbol{\Theta}'$) is positive-measure.

We now consider two cases. If $(l,i) \neq (k,j)$, then observe that $\boldsymbol{\Theta}'$ must contain some $\boldsymbol{\theta}, \boldsymbol{\theta}'$ such that $\boldsymbol{\theta} = (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, \mathbf{b}_L)$ and $\boldsymbol{\theta}' = (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_l, \mathbf{b}_l + \delta \mathbf{e}_i, \ldots, \mathbf{W}_L, \mathbf{b}_L)$, where $\delta \neq 0$ and $l \geq k$. By membership in $\boldsymbol{\Theta}'$, there exist $c, c' \in \mathbb{R}$ such that

$$[\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I})] = c[\mathbf{w}_j^k(\boldsymbol{\theta}, \mathbf{J}), b_j^k(\boldsymbol{\theta}, \mathbf{J})], \tag{A.20}$$

$$[\mathbf{w}_i^l(\boldsymbol{\theta}', \mathbf{I}), b_i^l(\boldsymbol{\theta}', \mathbf{I})] = c'[\mathbf{w}_j^k(\boldsymbol{\theta}', \mathbf{J}), b_j^k(\boldsymbol{\theta}', \mathbf{J})]. \tag{A.21}$$

Notice that $\mathbf{w}_i^l, \mathbf{w}_j^k$ do not depend on the $b_l[i]$-component of $\boldsymbol{\theta}$, and neither does b_j^k because $k \leq l$ and $(k,j) \neq (l,i)$. It follows that $[\mathbf{w}_j^k(\boldsymbol{\theta}, \mathbf{J}), b_j^k(\boldsymbol{\theta}, \mathbf{J})] = [\mathbf{w}_j^k(\boldsymbol{\theta}', \mathbf{J}), b_j^k(\boldsymbol{\theta}', \mathbf{J})] =: \mathbf{v}$. Notice also that $[\mathbf{w}_i^l(\boldsymbol{\theta}', \mathbf{I}), b_i^l(\boldsymbol{\theta}', \mathbf{I})] = [\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I}) + \delta]$. Putting everything together, we have that

$$c\mathbf{v} = [\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I})], \tag{A.22}$$

$$c'\mathbf{v} = [\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I}) + \delta], \tag{A.23}$$

which implies $(c' - c)\mathbf{v} = [\mathbf{0}, \delta]$, and in particular $\mathbf{w}_j^k(\boldsymbol{\theta}, \mathbf{J}) = \mathbf{0}$. This contradicts the assumption that $\boldsymbol{\theta} \in \boldsymbol{\Theta}^*$ and completes the proof for the case $(l,i) \neq (k,j)$.

If $(l,i) = (k,j)$, then it must be that $\mathbf{I}_{1:l-1} \neq \mathbf{J}_{1:l-1}$. Wlog, let $(\lambda, \iota) \in [l-1] \times [d_\lambda]$ be such that $I_\lambda[\iota, \iota] = 1$ and $J_\lambda[\iota, \iota] = 0$. Then there exist $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \boldsymbol{\Theta}'$ such that $\boldsymbol{\theta} = (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_L, \mathbf{b}_L)$ and $\boldsymbol{\theta}' = (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_\lambda, \mathbf{b}_\lambda + \delta \mathbf{e}_\iota, \ldots, \mathbf{W}_L, \mathbf{b}_L)$, where $\delta \neq 0$. Then there exist $c, c' \in \mathbb{R}$ such that

$$[\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I})] = c[\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{J}), b_i^l(\boldsymbol{\theta}, \mathbf{J})], \tag{A.24}$$

$$[\mathbf{w}_i^l(\boldsymbol{\theta}', \mathbf{I}), b_i^l(\boldsymbol{\theta}', \mathbf{I})] = c'[\mathbf{w}_i^l(\boldsymbol{\theta}', \mathbf{J}), b_i^l(\boldsymbol{\theta}', \mathbf{J})], \tag{A.25}$$

where as before, $\mathbf{w}_i^l$ does not depend on the $b_\lambda[\iota]$-component. For b_i^l we now have $b_i^l(\boldsymbol{\theta}', \mathbf{J}) = b_i^l(\boldsymbol{\theta}, \mathbf{J})$ and $b_i^l(\boldsymbol{\theta}', \mathbf{I}) = b_i^l(\boldsymbol{\theta}, \mathbf{I}) + d$, where $d = \delta \mathbf{e}_i^\intercal \mathbf{W}_l \mathbf{I}_{l-1} \cdots \mathbf{W}_{\lambda+1} \mathbf{e}_\iota$ and by membership in $\boldsymbol{\Theta}^*, d \neq 0$. From here, we can proceed as in the case $(l,i) \neq (k,j)$, completing the proof for $\boldsymbol{\Theta}^\neg$ being zero-measure.

Finally, we show that $\boldsymbol{\Theta}^\neg$ is closed. Let $\boldsymbol{\theta} \in \boldsymbol{\Theta}^* \setminus \boldsymbol{\Theta}^\neg$, i.e. for all $((l,i,\mathbf{I}),(k,j,\mathbf{J})) \in \mathcal{C}$, the vectors $[\mathbf{w}_i^l(\boldsymbol{\theta}, \mathbf{I}), b_i^l(\boldsymbol{\theta}, \mathbf{I})]$ and $[\mathbf{w}_j^k(\boldsymbol{\theta}, \mathbf{J}), b_j^k(\boldsymbol{\theta}, \mathbf{J})]$ are non-colinear. Since $\mathbf{w}_i^l, b_i^l, \mathbf{w}_j^k, b_j^k$ are continuous functions in $\boldsymbol{\theta}$, there exists a small enough $\epsilon > 0$ such that $[\mathbf{w}_i^l(\boldsymbol{\theta}', \mathbf{I}), b_i^l(\boldsymbol{\theta}', \mathbf{I})]$ and $[\mathbf{w}_j^k(\boldsymbol{\theta}', \mathbf{J}), b_j^k(\boldsymbol{\theta}', \mathbf{J})]$ are non-colinear for all $\boldsymbol{\theta}' \in \mathcal{B}_\epsilon(\boldsymbol{\theta})$ and all $((l,i,\mathbf{I}),(k,j,\mathbf{J})) \in \mathcal{C}$. Hence, $\boldsymbol{\Theta}^* \setminus \boldsymbol{\Theta}^\neg$ is open, and $\boldsymbol{\Theta}^\neg$ is closed. □

Lemma A.12. *For all ReLU nets $h : \mathcal{X} \to \mathbb{R}$ except a closed zero-measure set,*

$$\mathcal{F}_i^l(h) = \left\{\mathbf{x} \in \mathcal{X} \mid h_i^{1:l}(\mathbf{x}) = 0\right\} \tag{A.26}$$

$$= \left\{\mathbf{x} \in \mathcal{X} \mid h_i^{1:l} \textit{ is positive and negative in the neighbourhood of } \mathbf{x}\right\} \tag{A.27}$$

for all units (l,i).

Proof. We provide a proof for a single unit (l,i); the extension to all units follows from the finite number of units.

Let $\mathcal{G}, \mathcal{H}$, denote the sets defined on the right-hand sides of (A.26) and (A.27) respectively. Clearly, $\mathcal{H} \subseteq \mathcal{F}_i^l(h) \subseteq \mathcal{G}$. We will show that $\mathcal{G} \subseteq \mathcal{H}$. Let $\mathcal{Y} \subseteq \mathbb{R}$ denote the set of all local optima of the function $\mathbf{x} \mapsto \mathbf{W}_l[i,:] \cdot \check{h}^{1:l-1}(\mathbf{x})$. Due to piece-wise linearity of the function, and the finite number of pieces, $\mathcal{Y}$ is finite. It follows that for all ReLU networks except a closed zero-measure set, $-b_l[i] \notin \mathcal{Y}$. It is thus guaranteed that $h_i^{1:l}$ never attains a local maximum or minimum at zero. No $\mathbf{x} \in \mathcal{G}$ can therefore be a local maximum or minimum, implying that $h_i^{1:l}$ is both positive and negative in the neighbourhood of $\mathbf{x}$. Hence, $\mathbf{x} \in \mathcal{H}$. □

Definition A.11 (General ReLU network)**.** *A ReLU network is* general *if it satisfies Lemmas A.10, A.11 and A.12.*

All ReLU networks except a closed zero-measure set are general.

Lemma A.13. *If h is a general ReLU network, then $\mathcal{F}(h_i^{1:l}) = \bigcup_{j=1}^{d_{l-1}} \mathcal{F}(\check{h}_j^{1:l-1})$ for all (l, i).*

Proof. The inclusion $\mathcal{F}(h_i^{1:l}) \subseteq \bigcup_{j=1}^{d_{l-1}} \mathcal{F}(\check{h}_j^{1:l-1})$ follows from Lemma A.9. For the other inclusion, let $\mathbf{x} \in \mathcal{F}(\check{h}_k^{1:l-1})$ for some $k \in [d_{l-1}]$. Then there exist sequences of points $\mathbf{x}_1(\epsilon), \mathbf{x}_2(\epsilon) \in \mathcal{B}_\epsilon(\mathbf{x}) \setminus \bigcup_{j=1}^{d_{l-1}} \mathcal{F}(\check{h}_j^{1:l-1})$ such that $\mathbf{I}(\mathbf{x}_1(\epsilon)) =: \mathbf{I}$ and $\mathbf{I}(\mathbf{x}_2(\epsilon)) =: \mathbf{J}$ are independent of ϵ, and $\nabla \check{h}_k^{1:l-1}(\mathbf{x}_1(\epsilon)) \neq \nabla \check{h}_k^{1:l-1}(\mathbf{x}_2(\epsilon))$. We consider three cases based on the (non-)triviality of $\mathbf{I}$ and $\mathbf{J}$.

First, suppose both $\mathbf{I}$ and $\mathbf{J}$ are trivial up to $l-1$. Then by Lemma A.7,

$$\nabla \check{h}_k^{1:l-1}(\mathbf{x}_1(\epsilon)) = I_{l-1}[k,k]\, \mathbf{w}_k^{l-1}(\mathbf{I}) = \mathbf{0}, \tag{A.28}$$

and similarly $\nabla \check{h}_k^{1:l-1}(\mathbf{x}_2(\epsilon)) = \mathbf{0}$, which contradicts $\nabla \check{h}_k^{1:l-1}(\mathbf{x}_1(\epsilon) \neq \nabla \check{h}_k^{1:l-1}(\mathbf{x}_2(\epsilon)$. Hence, at least one of $\mathbf{I}, \mathbf{J}$, must be non-trivial up to $l-1$.

Second, say both $\mathbf{I}$ and $\mathbf{J}$ are non-trivial up to $l-1$. From $\nabla \check{h}_k^{1:l-1}(\mathbf{x}_1(\epsilon)) \neq \nabla \check{h}_k^{1:l-1}(\mathbf{x}_2(\epsilon))$ it follows that $\mathbf{I}_{1:l-1} \neq \mathbf{J}_{1:l-1}$, we can therefore apply Lemma A.11 to $(l, i, \mathbf{I})$ and $(l, i, \mathbf{J})$. We obtain $\mathbf{w}_i^l(\mathbf{I}) \neq \mathbf{w}_i^l(\mathbf{J})$, implying $\nabla h_i^{1:l}(\mathbf{x}_1(\epsilon)) \neq \nabla h_i^{1:l}(\mathbf{x}_2(\epsilon))$. Thus, $\mathbf{x}$ must be a fold-point of $h_i^{1:l}$.

Finally, say $\mathbf{I}$ is trivial up to $l-1$ and $\mathbf{J}$ is non-trivial up to $l-1$. Then $\nabla h_i^{1:l}(\mathbf{x}_1(\epsilon)) = \mathbf{w}_i^l(\mathbf{I}) = \mathbf{0}$, whereas Lemma A.11 applied to $(l, i, \mathbf{J})$ with $c = 0$ yields $\nabla h_i^{1:l}(\mathbf{x}_2(\epsilon)) = \mathbf{w}_i^l(\mathbf{J}) \neq \mathbf{0}$. Hence, $\nabla h_i^{1:l}(\mathbf{x}_1(\epsilon)) \neq \nabla h_i^{1:l}(\mathbf{x}_2(\epsilon))$ and $\mathbf{x}$ must be a fold-point of $h_i^{1:l}$. □

Definition A.12 (Transparent ReLU network)**.** *A ReLU network $h : \mathcal{X} \to \mathbb{R}$ is called* transparent up to layer m*, if for all $\mathbf{x} \in \mathcal{X}$ and $l \in [m]$, there exists $i \in [d_l]$ such that $h_i^{1:l}(\mathbf{x}) \geq 0$, or in other words, $\operatorname{rank}(\mathbf{I}_l(\mathbf{x})) \geq 1$. If h is transparent up to layer $L-1$, we say it is* transparent*.*

Lemma A.14. *Let $h : \mathcal{X} \to \mathbb{R}$ be a ReLU network, and let $\lambda \in [L]$. If h is general, then $h^{\lambda:L}|_{\operatorname{int} \mathcal{X}^{\lambda-1}}$ is general. If h is transparent, then $h^{\lambda:L}|_{\operatorname{int} \mathcal{X}^{\lambda-1}}$ is transparent.*

Proof. We will abbreviate $h^{\lambda:L}|_{\operatorname{int} \mathcal{X}^{\lambda-1}}$ as $h^{\lambda:L}$. Assume h is general. Then $h^{\lambda:L}$ clearly satisfies Lemma A.10, and for all (l, i), $\mathbf{W}_l[i,:] \neq \mathbf{0}^\intercal$. Next, we prove that $h^{\lambda:L}$ satisfies Lemma A.11. Suppose this was *not* the case; then there exist units $(\lambda-1+l, i), (\lambda-1+k, j)$, and non-trivial activation indicators $\mathbf{I} = (\mathbf{I}_\lambda, \ldots, \mathbf{I}_{\lambda-1+l})$, $\mathbf{J} = (\mathbf{J}_\lambda, \ldots, \mathbf{I}_{\lambda-1+k})$, with $(l, i, \mathbf{I}) \neq (k, j, \mathbf{J})$, and a scalar $C \in \mathbb{R}$ such that

$$\mathbf{e}_i^\intercal \mathbf{W}_{\lambda-1+l} \mathbf{I}_{\lambda-2+l} \cdots \mathbf{I}_\lambda \mathbf{W}_\lambda = C \cdot \mathbf{e}_j^\intercal \mathbf{W}_{\lambda-1+k} \mathbf{J}_{\lambda-2+k} \cdots \mathbf{J}_\lambda \mathbf{W}_\lambda, \tag{A.29}$$

and

$$\mathbf{e}_i^\intercal \sum_{l'=\lambda}^{\lambda-1+l} \mathbf{W}_{\lambda-1+l} \mathbf{I}_{\lambda-2+l} \cdots \mathbf{W}_{l'+1} \mathbf{I}_{l'} \mathbf{b}_{l'} = \tag{A.30}$$

$$C \cdot \mathbf{e}_j^\intercal \sum_{k'=\lambda}^{\lambda-1+k} \mathbf{W}_{\lambda-1+k} \mathbf{J}_{\lambda-2+k} \cdots \mathbf{W}_{k'+1} \mathbf{I}_{k'} \mathbf{b}_{k'}. \tag{A.31}$$

Then for any non-trivial indicator $(\mathbf{I}_1, \dots, \mathbf{I}_{\lambda-1}) \triangleq (\mathbf{J}_1, \dots, \mathbf{J}_{\lambda-1})$, we obtain by post-multiplying (A.29),

$$\mathbf{e}_i^\intercal \mathbf{W}_{\lambda-1+l} \mathbf{I}_{\lambda-2+l} \cdots \mathbf{I}_1 \mathbf{W}_1 = C \cdot \mathbf{e}_j^\intercal \mathbf{W}_{\lambda-1+k} \mathbf{J}_{\lambda-2+k} \cdots \mathbf{J}_1 \mathbf{W}_1, \tag{A.32}$$

and for all $\iota \in [\lambda - 1]$,

$$\mathbf{e}_i^\intercal \mathbf{W}_{\lambda-1+l} \mathbf{I}_{\lambda-2+l} \cdots \mathbf{W}_{\iota+1} \mathbf{I}_\iota \mathbf{b}_\iota = C \cdot \mathbf{e}_j^\intercal \mathbf{W}_{\lambda-1+k} \mathbf{J}_{\lambda-2+k} \cdots \mathbf{W}_{\iota+1} \mathbf{J}_\iota \mathbf{b}_\iota. \tag{A.33}$$

The first equality means that $\mathbf{w}_i^l(\mathbf{I}) = C \cdot \mathbf{w}_j^k(\mathbf{J})$, and the second equality implies $b_i^l(\mathbf{I}) = C \cdot b_j^k(\mathbf{J})$. However, that contradicts the fact that h satisfies Lemma A.11.

The last condition of generality is Lemma A.12. Suppose $h^{\lambda:L}$ does *not* satisfy the lemma. Then there exists a unit (l, i) such that

$$\Big\{\mathbf{x} \in \operatorname{int} \mathcal{X}^{l-1} \,|\, h_i^{\lambda:l}(\mathbf{x}) = 0\Big\} \not\subseteq \Big\{\mathbf{x} \in \operatorname{int} \mathcal{X}^{l-1} \,|\, h_i^{\lambda:l} \text{ is positive and negative in the neighbourhood of } \mathbf{x}\Big\},$$

i.e. there exists $\mathbf{x} \in \operatorname{int} \mathcal{X}^{l-1}$ such that $h_i^{\lambda:l}(\mathbf{x}) = 0$, and for some $\epsilon > 0$ either $h_i^{\lambda:l}(\mathcal{B}_\epsilon(\mathbf{x})) \subseteq (-\infty, 0]$ or $h_i^{\lambda:l}(\mathcal{B}_\epsilon(\mathbf{x})) \subseteq [0, \infty)$. However, then there exists $\mathbf{x}' \in \mathcal{X}$ such that $\check{h}^{1:l-1}(\mathbf{x}') = \mathbf{x}$, and for $\mathbf{x}'$ we obtain $h_i^{1:l}(\mathbf{x}') = 0$, and by continuity, there is $\delta > 0$ such that either $h_i^{1:l}(\mathcal{B}_\delta(\mathbf{x}')) \subseteq (-\infty, 0]$ or $h_i^{1:l}(\mathcal{B}_\delta(\mathbf{x}')) \subseteq [0, \infty)$. This contradicts the fact that h satisfies Lemma A.12. We have thus shown that if h is general, then $h^{\lambda:L}|_{\operatorname{int} \mathcal{X}^{\lambda-1}}$ is general.

Finally, assume h is transparent, i.e. for all $\mathbf{x} \in \mathcal{X}$ and $l \in [L-1]$, there exists $i \in [d_l]$ such that $h_i^{1:l}(\mathbf{x}) \geq 0$. Then also for all $\mathbf{x} \in \operatorname{int} \mathcal{X}^{\lambda-1}$ and $l \in \{\lambda, \dots, L-1\}$, there exists $i \in [d_l]$ such that $h_i^{\lambda:l}(\mathbf{x}) \geq 0$. Hence, $h^{\lambda:L}$ is transparent. □

Lemma A.15. *a) For all ReLU networks $h : \mathcal{X} \to \mathbb{R}$ and all $l \in [L]$, $i \in [d_l]$, we have $\mathcal{F}(h_i^{1:l}) \subseteq \bigcup_{k \in [l-1], j \in [d_k]} \mathcal{F}_j^k(h)$. In particular, $\mathcal{F}(h) \subseteq \bigcup_{k \in [L-1], j \in [d_l]} \mathcal{F}_j^k(h)$.*

b) For all general ReLU networks $h : \mathcal{X} \to \mathbb{R}$ transparent up to layer $l-1$, we have $\mathcal{F}(h_i^{1:l}) = \bigcup_{k \in [l-1], j \in [d_k]} \mathcal{F}_j^k(h)$. In particular, for all general transparent ReLU networks, $\mathcal{F}(h) = \bigcup_{k \in [L-1], j \in [d_k]} \mathcal{F}_j^k(h)$.

Proof. We give a proof of b) only. A proof of a) can be obtained by replacing some equalities by inclusions. We will prove by induction that $\mathcal{F}(h_i^{1:l}) = \bigcup_{k \in [l-1], j \in [d_k]} \mathcal{F}_j^k(h)$ if h is general and transparent up to layer $l-1$. For $l = 1$, the function $h_i^{1:l}$ is linear, so $\mathcal{F}(h) = \emptyset$ and the claim holds trivially. Now assume that $\mathcal{F}(h_i^{1:l}) = \bigcup_{k \in [l-1], j \in [d_k]} \mathcal{F}_j^k(h)$ holds; we will prove the same statement for $l+1$. By Lemma A.8 and Lemma A.13, we have

$$\mathcal{F}(\check{h}_i^{1:l}) = \Big(\Big\{\mathbf{x} \in \mathcal{X} \,|\, h_i^{1:l}(\mathbf{x}) > 0\Big\} \cap \mathcal{F}(h_i^{1:l})\Big) \cup \mathcal{F}_i^l(h) \tag{A.34}$$

$$= \Big(\Big\{\mathbf{x} \in \mathcal{X} \,|\, h_i^{1:l}(\mathbf{x}) > 0\Big\} \cap \bigcup_{k \in [l-1], j \in [d_k]} \mathcal{F}_j^k(h)\Big) \cup \mathcal{F}_i^l(h), \tag{A.35}$$

$$\mathcal{F}(h_\iota^{1:l+1}) = \bigcup_{i=1}^{d_l} \Big(\Big\{\mathbf{x} \in \mathcal{X} \,|\, h_i^{1:l}(\mathbf{x}) > 0\Big\} \cap \bigcup_{k \in [l-1], j \in [d_k]} \mathcal{F}_j^k(h)\Big) \cup \mathcal{F}_i^l(h) \tag{A.36}$$

$$= \Big(\bigcup_{i=1}^{d_l} \Big\{\mathbf{x} \in \mathcal{X} \,|\, h_i^{1:l}(\mathbf{x}) > 0\Big\} \cap \bigcup_{k \in [l-1], j \in [d_k]} \mathcal{F}_j^k(h)\Big) \cup \bigcup_{i=1}^{d_l} \mathcal{F}_i^l(h). \tag{A.37}$$

Since $\bigcup_{i=1}^{d_l} \Big\{\mathbf{x} \in \mathcal{X} \,|\, h_i^{1:l}(\mathbf{x}) > 0\Big\} \subseteq \mathcal{X}$, we obtain

$$\mathcal{F}(h_\iota^{1:l+1}) \subseteq \bigcup_{k \in [l-1], j \in [d_k]} \mathcal{F}_j^k(h) \cup \bigcup_{j \in [d_l]} \mathcal{F}_j^l(h) = \bigcup_{k \in [l], j \in [d_k]} \mathcal{F}_j^k(h). \tag{A.38}$$

It remains to show the reverse inclusion; we do so by contradiction.

Suppose $\bar{\mathbf{x}} \in \bigcup_{k\in[l],j\in[d_k]} \mathcal{F}_j^k(h) \setminus \mathcal{F}(h_\iota^{1:l+1})$, or equivalently

$$\bar{\mathbf{x}} \in \bigcup_{k\in[l-1],j\in[d_k]} \mathcal{F}_j^k(h) \cap \left(\mathcal{X} \setminus \bigcup_{i\in[d_l]} \left\{\mathbf{x}\in\mathcal{X} \mid h_i^{1:l}(\mathbf{x}) > 0\right\}\right) \tag{A.39}$$

$$= \bigcup_{k\in[l-1],j\in[d_k]} \mathcal{F}_j^k(h) \cap \bigcap_{i\in[d_l]} \left\{\mathbf{x}\in\mathcal{X} \mid h_i^{1:l}(\mathbf{x}) \le 0\right\}. \tag{A.40}$$

Because h is transparent, there exists $i \in [d_l] : h_i^{1:l}(\bar{\mathbf{x}}) \ge 0$, so for this i we have $h_i^{1:l}(\bar{\mathbf{x}}) = 0$. However, by Lemma A.12, this implies $\bar{\mathbf{x}} \in \mathcal{F}_i^l(h) \subseteq \mathcal{F}(h_\iota^{1:l+1})$. □

Lemma A.16. *Let $h : \mathcal{X} \to \mathbb{R}$ be a ReLU network. Then $\mathcal{F}_i^{l+1}(h)$ is a union of piece-wise hyperplanes wrt. $\mathcal{P}_\mathcal{X}(\bigcup_{k\in[l],j\in[d_k]} \mathcal{F}_j^k(h))$.*

Proof. Since $\mathcal{P}_\mathcal{X}(\mathcal{F}(h_i^{1:l+1}))$ is the partition of the input space into the linear regions of $h_i^{1:l+1}$, and $\mathcal{F}(h_i^{1:l+1}) \subseteq \bigcup_{k\in[l],j\in[d_k]} \mathcal{F}_j^k(h)$ by Lemma A.15, the function $h_i^{1:l+1}$ is also linear on the regions of $\mathcal{P}_\mathcal{X}(\bigcup_{k\in[l],j\in[d_k]} \mathcal{F}_j^k(h))$. For any $P \in \mathcal{P}_\mathcal{X}(\bigcup_{k\in[l],j\in[d_k]} \mathcal{F}_j^k(h))$, denote the slope and bias of $h_i^{1:l+1}$ on P by $\mathbf{w}(P), b(P)$. Then

$$P \cap \mathcal{F}_i^{l+1}(h) = \{\mathbf{x} \in P \mid \mathbf{w}(P)^\intercal \mathbf{x} + b(P) = 0 \tag{A.41}$$

$$\text{and } h_i^{1:l+1} \text{ is positive in the neighbourhood of } \mathbf{x}\}. \tag{A.42}$$

The positivity condition guarantees that $(\mathbf{w}(P), b(P)) \neq (\mathbf{0}, 0)$, so $P \cap \mathcal{F}_i^{l+1}$ is either an empty set or a piece-wise hyperplane. □

Corollary A.1. *Let $h : \mathcal{X} \to \mathbb{R}$ be a ReLU network. Then the set $\bigcup_{l\in[\kappa],i\in[d_l]} \mathcal{F}_i^l(h)$ is a pwl. surface of order at most κ. In particular, if h is general and transparent, then $\mathcal{F}(h) = \bigcup_{l\in[L-1],i\in[d_l]} \mathcal{F}_i^l(h)$ is a pwl. surface of order at most $L-1$.*

A.4 Main result

Lemma A.17. *For any bounded domain $\mathcal{X}$ and any architecture $(d_1, \ldots, d_{L-1})$ with $d_0 \ge d_1 \ge \cdots \ge d_{L-1} \ge 2$, there exists a nonempty open set of transparent ReLU networks $h : \mathcal{X} \to \mathbb{R}$ such that for $l \in [L-1]$,*

- $\dim \mathcal{X}^l = d_l$, *and*
- *the set $\mathcal{F}(h^{l:L}|_{\operatorname{int}\mathcal{X}^{l-1}})$ is a pwl. surface whose dependency graph contains d_l directed paths of length $(L-1-l)$ with distinct starting vertices.*

Proof. We give an explicit construction; we first state it and then we prove its properties. For $l = 1$, we choose the parameters $(\mathbf{W}_l, \mathbf{b}_l)$ as follows. Let P_1 be some nonempty open convex subset of $\mathcal{X}$, and define

$$\mathcal{W}_1^l := \left\{(\mathbf{w}, b) \in \mathbb{R}^{d_{l-1}} \times \mathbb{R} \;\middle|\; \inf_{\mathbf{x}\in P_l} \mathbf{w}^\intercal \mathbf{x} + b < 0 < \sup_{\mathbf{x}\in P_l} \mathbf{w}^\intercal \mathbf{x} + b\right\}. \tag{A.43}$$

Since P_l has at least two elements, the strict separation theorem implies that $\mathcal{W}_1^l$ is nonempty. It is also open. We can therefore choose $(\mathbf{W}_l[1,:], b_l[1])$ from $\mathcal{W}_1^l$, and define

$$\begin{aligned}\mathcal{W}_2^l := \{(\mathbf{w}, b) \in \mathcal{W}_1^l \mid \mathbf{w}^\intercal \mathbf{x} + b > 0 \text{ for all } \mathbf{x} \in \mathcal{X}^l \\ \text{such that } \mathbf{W}_l[1,:]\mathbf{x} + b_l[1] \leq 0\}.\end{aligned} \tag{A.44}$$

The set $\mathcal{A} := \left\{\mathbf{x} \in \operatorname{conv} \operatorname{cl} \mathcal{X}^l \mid \mathbf{W}_l[1,:]\mathbf{x} + b_l[1] \leq 0\right\}$ is nonempty, convex and compact, and by construction there exists $\mathbf{x} \in P_l \setminus \mathcal{A}$. Again, the strict separation theorem implies $\mathcal{W}_2^l$ is nonempty. It is also open by the boundedness of $\mathcal{A}$. We then choose $(\mathbf{W}_l[2,:], b_l[2])$ from $\mathcal{W}_2^l$, and define

$$\bar{Q}_\iota \triangleq \{\mathbf{x} \in P_l \mid \mathbf{W}_l[i,:]\mathbf{x} + b_l[i] \geq 0 \text{ for all } i \in [\iota]\}. \tag{A.45}$$

Note that $\operatorname{int} \bar{Q}_2$ is nonempty, open and convex. Also, there exist $\mathbf{x}_1, \mathbf{x}_2 \in \bar{Q}_2$ such that

$$\mathbf{W}_l[i,:]\mathbf{x}_i + b_l[i] = 0, \tag{A.46}$$

$$\mathbf{W}_l[j,:]\mathbf{x}_i + b_l[j] > 0 \text{ for } j \neq i. \tag{A.47}$$

For $i \in \{3, \ldots, d_l\}$, we then choose $(\mathbf{W}_l[i,:], b_l[i])$ as follows. Let $\mathcal{A} := \operatorname{conv}\{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}\}$ and let $\mathbf{x} \in \operatorname{int} \bar{Q}_{i-1} \setminus \mathcal{A}$. Such $\mathbf{x}$ exists because $\dim \bar{Q}_{i-1} = \dim P_l = d_{l-1}$, and $\dim \mathcal{A} \leq i - 2 \leq d_l - 2 \leq d_{l-1} - 2$. By strict separation and boundedness, there exists a nonempty open set of hyperplanes that strictly separate $\mathcal{A}$ and $\mathbf{x}$. Let $(\mathbf{W}_l[i,:], b_l[i])$ be any of them, oriented such that $\mathbf{W}_l[i,:]\mathbf{x}_j + b_l[i] > 0$ for $j \in [i-1]$. Then denote by $\mathbf{x}_i \in \operatorname{int} \bar{Q}_{i-1}$ any point that satisfies $\mathbf{W}_l[i,:]\mathbf{x}_i + b_l[i] = 0$; it exists by convexity. Then $\operatorname{int} \bar{Q}_i$ is nonempty and convex, and the construction for $\{i+1, \ldots, d_l\}$ goes through.

For layers $l \in \{2, \ldots, L-1\}$, we use a similar construction as for $l = 1$. Denote

$$P_l := (\mathbf{W}_{l-1} P_{l-1} + \mathbf{b}_{l-1}) \cap \{\mathbf{x} > \mathbf{0}\}, \tag{A.48}$$

$$\bar{P}_l := (\mathbf{W}_{l-1} P_{l-1} + \mathbf{b}_{l-1}) \cap \{\mathbf{x} \geq \mathbf{0}\}. \tag{A.49}$$

P_l is nonempty (because $\operatorname{int} \bar{Q}_{d_{l-1}}$ is nonempty), open (because $\mathbf{W}_{l-1}$ is wide) and convex. We assume $\mathbf{W}_{l-1}$ is full-rank; this holds for all choices of $\mathbf{W}_{l-1}$ except a closed zero-measure set. By the construction of $\bar{Q}_{d_{l-1}}$, there exist $\mathbf{x}_1, \ldots, \mathbf{x}_{d_{l-1}} \in \bar{Q}_{d_{l-1}} \subseteq P_{l-1}$ such that or $i \in [d_{l-1}]$,

$$\mathbf{W}_{l-1}[i,:]\mathbf{x}_i + b_{l-1}[i] = 0, \tag{A.50}$$

$$\mathbf{W}_{l-1}[j,:]\mathbf{x}_i + b_{l-1}[j] > 0 \text{ for } j \neq i. \tag{A.51}$$

Denote their images $\mathbf{x}_i' := \mathbf{W}_{l-1}\mathbf{x}_i + \mathbf{b}_{l-1}$. Then by openness, there exists a ball around each $\mathbf{x}_i'$ such that

$$\mathcal{B}_\epsilon(\mathbf{x}_i') \subseteq (\mathbf{W}_{l-1} P_{l-1} + \mathbf{b}_{l-1}) \cap \{z'[j] > 0 \text{ for } j \neq i\}, \tag{A.52}$$

implying that each $\mathbf{x}_i'$ has a d_{l-1}-dimensional, relatively open neighbourhood $\mathcal{B}_\epsilon(\mathbf{x}_i') \cap \{z'[i] = 0\} \subseteq \bar{P}_l$ whose elements satisfy $z'[i] = 0$ and $z'[j] > 0$ for $j \neq i$. It follows that the set

$$\begin{aligned}\mathcal{W}_1^l := \Bigg\{(\mathbf{w}, b) \in \mathbb{R}^{d_{l-1}} \times \mathbb{R} \;\Bigg|\; \inf_{\mathbf{x} \in P_l} \mathbf{w}^\intercal \mathbf{x} + b < 0 < \sup_{\mathbf{x} \in P_l} \mathbf{w}^\intercal \mathbf{x} + b, \text{ and} \\ \forall i \in [d_{l-1}]\ \exists \mathbf{x} \in \bar{P}_l : z[i] = 0, z[j] > 0 \text{ for } j \neq i, \text{ and } \mathbf{w}^\intercal \mathbf{x} + b = 0\Bigg\}\end{aligned} \tag{A.53}$$

contains a nonempty open subset. We can therefore choose $(\mathbf{W}_l[1,:], b_l[1])$ from $\mathcal{W}_1^l$. For the choice of $(\mathbf{W}_l[i,:], b_l[i])$ for $i \in \{2, \dots, d_l\}$, we use the same procedure as in the first layer. Finally, choose $(\mathbf{W}_L, b_L)$ arbitrarily.

We will show that this construction satisfies the lemma. The networks are transparent because of how we define $\mathcal{W}_2^l$: for all $\mathbf{x} \in \mathcal{X}$ and $l \in [L-1]$, either $h_1^{1:l}(\mathbf{x}) > 0$ or $h_2^{1:l}(\mathbf{x}) > 0$. Also, $\dim \mathcal{X}^l = d_l$ because $\mathcal{X}^l$ contains $\operatorname{int} \bar{Q}_{d_l}$, which is nonempty and open.

Now let $l \in [L]$. We can think of the function $h^{l:L}|_{\operatorname{int} \mathcal{X}^{l-1}}$ as an $(L-l+1)$-layer ReLU network parameterised by $(\mathbf{W}_l, \mathbf{b}_l, \dots, \mathbf{W}_L, b_L)$. Because h is transparent, also $h^{l:L}|_{\operatorname{int} \mathcal{X}^{l-1}}$ is transparent. Corollary A.1 then implies that $\mathcal{S} := \mathcal{F}(h^{l:L}|_{\operatorname{int} \mathcal{X}^{l-1}}) = \bigcup_{k \in [L-l], j \in [d_k]} \mathcal{F}_j^k(h^{l:L}|_{\operatorname{int} \mathcal{X}^{l-1}})$ is a pwl. surface. Let $\mathcal{S} = \bigcup_{k \in [L-l], j \in [n_\lambda]} \mathcal{H}_j^k$ be its canonical representation, and let G denote its dependency graph.

To find the required paths in G, we first identify some important vertices. For $\lambda \in [L-l]$, denote

$$\mathcal{X}_\lambda^+ := \Big\{ \mathbf{x} \in P_l \,|\, h^{l:l+k-1}(\mathbf{x}) > \mathbf{0} \text{ for } k \in [\lambda - 1] \Big\}. \tag{A.54}$$

This set is nonempty and open because $P_{l+\lambda}$ is nonempty and open. Next, for any unit (λ, ι),

$$\begin{aligned} \mathcal{F}_\iota^\lambda(h^{l:L}|_{\operatorname{int} \mathcal{X}^{l-1}}) \cap \mathcal{X}_\lambda^+ = \{ \mathbf{x} \in P_l \,|\, & h_\iota^{l:l+\lambda-1}(\mathbf{x}) = 0, \\ & h^{l:l+k-1}(\mathbf{x}) > \mathbf{0} \text{ for } k \in [\lambda - 1] \}. \end{aligned} \tag{A.55}$$

By the definition of $\mathcal{W}_1^{l+\lambda-1}$ and because $h^{l:l+\lambda-1}(\mathcal{X}_\lambda^+) = P_{l+\lambda}$, the set $\mathcal{A} := \mathcal{F}_\iota^\lambda(h^{l:L}|_{\operatorname{int} \mathcal{X}^{l-1}}) \cap \mathcal{X}_\lambda^+$ is nonempty. Also, by the definition of $\{P_{l'}\}_{l'}$, $h_\iota^{l:l+\lambda-1}$ is in fact linear on $\mathcal{X}_\lambda^+$, so $\mathcal{A}$ is a hyperplane on $\mathcal{X}_\lambda^+$. Therefore there exists a piece-wise hyperplane $\mathcal{H}_{j(\lambda,\iota)}^{k(\lambda,\iota)}$ from the canonical representation of $\mathcal{S}$ that contains $\mathcal{A}$; by Lemma A.11, all $\{\mathcal{H}_{j(\lambda,\iota)}^{k(\lambda,\iota)}\}_{\lambda,\iota}$ are distinct from each other.

We now show that G contains the edge $\mathcal{H}_{j(\lambda,i)}^{k(\lambda,i)} \to \mathcal{H}_{j(\lambda+1,\iota)}^{k(\lambda+1,\iota)}$ for $\lambda \in [L-l-1]$ and all (i, ι). By the definition of $\mathcal{W}^{l+\lambda}$, there exists $\bar{\mathbf{x}} \in \bar{P}_{l+\lambda}$ such that $\bar{z}[i] = 0$, $\bar{z}[j] > 0$ for $j \neq i$, and $\mathbf{W}_{l+\lambda}[\iota,:]\bar{\mathbf{x}} + b_{l+\lambda}[\iota] = 0$. There also exists a ball $\mathcal{B}_\epsilon(\bar{\mathbf{x}}) \subseteq (\mathbf{W}_{l+\lambda-1} P_{l+\lambda-1} + \mathbf{b}_{l+\lambda-1}) \cap \{z[j] > 0 \text{ for } j \neq i\}$. Then because of how $\{P_{l'}\}_{l'}$ are defined, there exists $\bar{\mathbf{x}}' \in P_l$ such that $h^{l:l+\lambda-1}(\bar{\mathbf{x}}') = \bar{\mathbf{x}}$, so it satisfies

$$h_i^{l:l+\lambda-1}(\bar{\mathbf{x}}') = 0, \tag{A.56}$$

$$h_j^{l:l+k-1}(\bar{\mathbf{x}}') > 0, \text{ for } k \in [\lambda], (k,j) \neq (\lambda, i), \tag{A.57}$$

$$h_\iota^{l:l+\lambda}(\bar{\mathbf{x}}') = 0. \tag{A.58}$$

It follows that $\bar{\mathbf{x}}' \in \mathcal{F}_i^\lambda(h^{l:L}|_{\operatorname{int} \mathcal{X}^{l-1}}) \cap \mathcal{X}_\lambda^+ \subseteq \mathcal{H}_{j(\lambda,i)}^{k(\lambda,i)}$. At the same time, the preimage $(h^{l:l+\lambda-1})^{-1}(\mathcal{B}_\epsilon(\bar{\mathbf{x}}))$ is open by continuity, and contains $\bar{\mathbf{x}}'$. So there exists a ball $\mathcal{B}_\epsilon(\bar{\mathbf{x}}') \subseteq P_l$ such that all $\mathbf{x}' \in \mathcal{B}_\epsilon(\bar{\mathbf{x}}')$ satisfy

$$h_j^{l:l+k-1}(\mathbf{x}') > 0, \text{ for } k \in [\lambda], (k,j) \neq (\lambda, i). \tag{A.59}$$

On this ball, $h_i^{l:l+\lambda-1}$ is linear, so the set $\mathcal{A} := \mathcal{B}_\epsilon(\bar{\mathbf{x}}') \cap \Big\{ h_i^{l:l+\lambda-1}(\mathbf{x}') > 0 \Big\}$ is an open half-ball. On $\mathcal{A}$, $h_\iota^{l:l+\lambda}$ is linear as well, and the set $\Big\{ \mathbf{x}' \,|\, h_\iota^{l:l+\lambda}(\mathbf{x}') = 0 \Big\}$ intersects the center of the half-ball, $\bar{\mathbf{x}}'$. Therefore there exists a sequence of points $\{\mathbf{x}_n'\} \subseteq P_l$ such that $\mathbf{x}_n' \to \bar{\mathbf{x}}'$ and

$$h^{l:l+k-1}(\mathbf{x}') > \mathbf{0}, \text{ for } k \in [\lambda], \tag{A.60}$$

$$h_\iota^{l:l+\lambda}(\mathbf{x}') = 0. \tag{A.61}$$

We obtain that $\bar{\mathbf{x}}' \in \mathrm{cl}(\mathcal{F}_\iota^{\lambda+1}(h^{l:L}|_{\mathrm{int}\,\mathcal{X}^{l-1}}) \cap \mathcal{X}_{\lambda+1}^+) \subseteq \mathrm{cl}\ \mathcal{H}_{j(\lambda+1,\iota)}^{k(\lambda+1,\iota)}$, which implies

$$\mathrm{cl}\ \mathcal{H}_{j(\lambda,i)}^{k(\lambda,i)} \cap \mathrm{cl}\ \mathcal{H}_{j(\lambda+1,\iota)}^{k(\lambda+1,\iota)} \neq \emptyset. \tag{A.62}$$

It remains to show that $k(\lambda,i) < k(\lambda+1,\iota)$. Consider again the ball $\mathcal{B}_\epsilon(\bar{\mathbf{x}}')$. By Lemma A.11, $h_\iota^{l:l+\lambda}$ is a different linear function on the region $\mathcal{B}_\epsilon(\bar{\mathbf{x}}') \cap \left\{h_i^{l:l+\lambda-1}(\mathbf{x}') > 0\right\}$ and on the region $\mathcal{B}_\epsilon(\bar{\mathbf{x}}') \cap \left\{h_i^{l:l+\lambda-1}(\mathbf{x}') < 0\right\}$. Hence, $\mathcal{H}_{j(\lambda+1,\iota)}^{k(\lambda+1,\iota)}$ is *not* a piece-wise hyperplane wrt. any partition that does not include $\mathcal{H}_{j(\lambda,i)}^{k(\lambda,i)}$. We obtain that $k(\lambda,i) < k(\lambda+1,\iota)$, proving that G contains the edge $\mathcal{H}_{j(\lambda,i)}^{k(\lambda,i)} \to \mathcal{H}_{j(\lambda+1,\iota)}^{k(\lambda+1,\iota)}$.

Finally, observe that the d_l paths $\mathcal{H}_{j(1,i)}^{k(1,i)} \to \mathcal{H}_{j(2,1)}^{k(2,1)} \to \cdots \to \mathcal{H}_{j(L-l,1)}^{k(L-l,1)}$ have length $(L-l-1)$, and distinct starting vertices. This proves the theorem. □

Lemma A.18. *For all general ReLU networks $h : \mathcal{X} \to \mathbb{R}$, the following holds. Denote $\mathcal{S} = \bigcup_{l\in[\lambda],i\in[d_l]} \mathcal{F}_i^l(h)$ and let $\mathcal{S} = \bigcup_{k\in[\kappa],j\in[n_k]} \mathcal{H}_j^k$ be the canonical representation of $\mathcal{S}$. Then $\mathcal{H}_j^k \subseteq \mathcal{F}_i^l(h)$ for some (l,i) with $l \geq k$. Moreover, if the dependency graph of $\mathcal{S}$ contains a directed path of length m starting at $\mathcal{H}_j^k$, then $l \leq \lambda - m$.*

Proof. Because the representation is canonical, we have

$$\mathcal{H}_j^k \not\subseteq \square_{k-1}\mathcal{S} \supseteq \bigcup_{l\in[k-1],i\in[d_l]} \mathcal{F}_i^l(h), \tag{A.63}$$

which implies $\mathcal{H}_j^k \subseteq \bigcup_{l\geq k,i} \mathcal{F}_i^l(h)$. By piece-wise linearity, we can write

$$\bigcup_{l\geq k,i} \mathcal{F}_i^l(h) = \bigcup_{l\geq k,i,P} \mathcal{F}_i^l(h) \cap P = \bigcup_{l\geq k,i,P} \left\{\mathbf{x} \in P \,|\, \mathbf{w}_i^l(P)\cdot\mathbf{x} + b_i^l(P) = 0\right\}, \tag{A.64}$$

where P runs over the linear regions of $h_i^{1:l}$. Moreover, by the definition of $\mathcal{F}_i^l(h)$, all $\mathbf{w}_i^l(P) \neq \mathbf{0}$. Combined with Lemma A.11, we obtain that each nonempty set on the right-hand side of (A.64) is a different hyperplane on an open set. Therefore there exists one for which $\mathcal{H}_j^k \subseteq \mathcal{F}_i^l(h) \cap P \subseteq \mathcal{F}_i^l(h)$.

Now assume that the dependency graph of $\mathcal{S}$ contains a directed path of length m starting at $\mathcal{H}_j^k =: \mathcal{H}_{j_0}^{k_0}$; denote the path $\mathcal{H}_{j_0}^{k_0} \to \mathcal{H}_{j_1}^{k_1} \to \cdots \to \mathcal{H}_{j_m}^{k_m}$. By the first part of the lemma, we know that $\mathcal{H}_{j_\iota}^{k_\iota} \subseteq \mathcal{F}_{i_\iota}^{l_\iota}(h)$ for some l_ι. Let $\iota \in [m]$; we will show that $l_{\iota-1} < l_\iota$.

Because of the edge $\mathcal{H}_{j_{\iota-1}}^{k_{\iota-1}} \to \mathcal{H}_{j_\iota}^{k_\iota}$, we know that $\mathcal{H}_{j_{\iota-1}}^{k_{\iota-1}}$ and $\mathrm{cl}\ \mathcal{H}_{j_\iota}^{k_\iota}$ intersect, and that $\mathcal{H}_{j_\iota}^{k_\iota}$ is a piece-wise hyperplane wrt. some partition $\mathcal{P}$ for which $\mathcal{H}_{j_{\iota-1}}^{k_{\iota-1}}$ is a boundary. Let $\hat{\mathbf{x}}$ be any point of intersection. By openness, there exists a ball $\mathcal{B}_\epsilon(\hat{\mathbf{x}})$ such that $\mathcal{H}_{j_{\iota-1}}^{k_{\iota-1}}$ is a hyperplane on $\mathcal{B}_\epsilon(\hat{\mathbf{x}})$, and $\mathcal{H}_{j_\iota}^{k_\iota}$ is a hyperplane on one half-ball defined by $\mathcal{B}_\epsilon(\hat{\mathbf{x}})$ and $\mathcal{H}_{j_{\iota-1}}^{k_{\iota-1}}$. If it was the case that $l_{\iota-1} \geq l_\iota$, then $\mathcal{F}_{i_\iota}^{l_\iota}(h)$ would be a hyperplane on $\mathcal{B}_\epsilon(\hat{\mathbf{x}})$, i.e. there would have to exist some piece-wise hyperplane on the opposite half-ball as $\mathcal{H}_{j_{\iota-1}}^{k_{\iota-1}}$, but included in the same hyperplane. However, by Lemma A.11, no two piece-wise hyperplanes in $\mathcal{S}$ are included in a single hyperplane, so we get a contradiction.

Hence, we obtain $l_0 < l_1 < \cdots < l_m \leq \lambda$, which yields $l_0 \leq \lambda - m$. □

Lemma A.19. *Let $(\mathbf{w},b),(\mathbf{c},a) \in \mathbb{R}^d \times \mathbb{R}$ and let $F \subseteq \mathbb{R}^d$ with $\dim F = d-1$. If $\mathbf{w}^\intercal\mathbf{x}+b = 0$ and $\mathbf{c}^\intercal\mathbf{x} + a = 0$ for all $\mathbf{x} \in F$, then either $(\mathbf{w},b) = (\mathbf{0},0)$, $(\mathbf{c},a) = (\mathbf{0},0)$, or there exists $\beta \in \mathbb{R} : (\mathbf{c},a) = \beta(\mathbf{w},b)$.*

Proof. Since $\dim F = d-1$, there exist d affinely independent vectors $\mathbf{f}_0, \ldots, \mathbf{f}_{d-1}$ in F. Hence there are $d-1$ linearly independent vectors $\mathbf{v}_1 := \mathbf{f}_1 - \mathbf{f}_0$, ..., $\mathbf{v}_{d-1} := \mathbf{f}_{d-1} - \mathbf{f}_0$, such that $\mathbf{w}^\mathsf{T}\mathbf{v}_i = \mathbf{c}^\mathsf{T}\mathbf{v}_i = 0$. In other words, both $\mathbf{w}$ and $\mathbf{c}$ lie in the orthogonal complement of the span of $\mathbf{v}_1, \ldots, \mathbf{v}_{d-1}$. If $\mathbf{w} = \mathbf{0}$, then necessarily $b = 0$, and similarly for $(\mathbf{c}, a)$. If $\mathbf{w} \neq \mathbf{0} \neq \mathbf{c}$, then because the orthogonal complement is one-dimensional, there exists $\beta \in \mathbb{R}$ such that $\mathbf{c} = \beta\mathbf{w}$. Then $\mathbf{c}^\mathsf{T}\mathbf{x} + a - \beta(\mathbf{w}^\mathsf{T}\mathbf{x} + b) = a - \beta b = 0$ and the lemma follows. □

Theorem A.1. *Consider a bounded domain $\mathcal{X}$ and any architecture $(d_1, \ldots, d_{L-1})$ with $d_0 \geq d_1 \geq \cdots \geq d_{L-1} \geq 2$. Let $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}$ be a general ReLU network satisfying Lemma A.17, and let $h_{\boldsymbol{\eta}} : \mathcal{X} \to \mathbb{R}$ be any general ReLU network such that $h_{\boldsymbol{\theta}}(\mathbf{x}) = h_{\boldsymbol{\eta}}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. Denote $\boldsymbol{\eta} \triangleq (\mathbf{W}'_1, \mathbf{b}'_1, \ldots, \mathbf{W}'_L, b'_L)$. Then there exist permutation matrices $\mathbf{P}_1, \ldots \mathbf{P}_{L-1}$, and positive-entry diagonal matrices $\mathbf{M}_1, \ldots, \mathbf{M}_{L-1}$, such that*

$$\begin{aligned} \mathbf{W}_1 &= \mathbf{M}_1\mathbf{P}_1\mathbf{W}'_1, & \mathbf{b}_1 &= \mathbf{M}_1\mathbf{P}_1\mathbf{b}'_1, & & \\ \mathbf{W}_l &= \mathbf{M}_l\mathbf{P}_l\mathbf{W}'_l\mathbf{P}_{l-1}^{-1}\mathbf{M}_{l-1}^{-1}, & \mathbf{b}_l &= \mathbf{M}_l\mathbf{P}_l\mathbf{b}'_l, & l &\in \{2, \ldots, L-1\}, \\ \mathbf{W}_L &= \mathbf{W}'_L\mathbf{P}_{L-1}^{-1}\mathbf{M}_{L-1}^{-1}, & b_L &= b'_L. & & \end{aligned} \tag{A.65}$$

Proof. First, notice that $h_{\boldsymbol{\eta}}$ is transparent. To see this, observe that $h_{\boldsymbol{\theta}}$ is transparent, i.e. $\operatorname{rank}(\mathbf{I}_l^{\boldsymbol{\theta}}(\mathbf{x})) \geq 1$ for all $l \in [L-1]$ and $\mathbf{x} \in \mathcal{X}$. By Lemma A.10, $\nabla_{\mathbf{x}} h_{\boldsymbol{\eta}}(\mathbf{x}) = \nabla_{\mathbf{x}} h_{\boldsymbol{\theta}}(\mathbf{x}) \neq \mathbf{0}^\mathsf{T}$, implying that $h_{\boldsymbol{\eta}}$ is transparent.

We proceed by induction. Let $l = 1$. Then we have

$$h_{\boldsymbol{\theta}}^{l:L}|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}} \equiv h_{\boldsymbol{\theta}} \equiv h_{\boldsymbol{\eta}} \equiv h_{\boldsymbol{\eta}}^{l:L}|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}} \tag{A.66}$$

which implies $\mathcal{F}(h_{\boldsymbol{\theta}}^{l:L}|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}}) = \mathcal{F}(h_{\boldsymbol{\eta}}^{l:L}|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}})$. (For notational convenience, we will omit the domain restriction for now.) Because both networks are general and transparent, Corollary A.1 implies that the set

$$\bigcup_{k \in [L-l], j \in [d_k]} \mathcal{F}_j^k(h_{\boldsymbol{\theta}}^{l:L}) = \mathcal{F}(h_{\boldsymbol{\theta}}^{l:L}) = \mathcal{F}(h_{\boldsymbol{\eta}}^{l:L}) = \bigcup_{k \in [L-l], j \in [d_k]} \mathcal{F}_j^k(h_{\boldsymbol{\eta}}^{l:L}) \tag{A.67}$$

is a pwl. surface of order at most $L - l$. By Lemma A.17, its graph contains d_l directed paths of length $(L-1-l)$ with distinct starting vertices. Denote these vertices $\mathcal{H}_1, \ldots, \mathcal{H}_{d_l}$. By Lemma A.18, $\mathcal{H}_i \subseteq \mathcal{F}_\iota^\lambda(h_{\boldsymbol{\theta}}^{l:L})$ for some (λ, ι) with $\lambda \leq (L-l) - (L-1-l) = 1$. We thus obtain $\bigcup_{i \in [d_l]} \mathcal{H}_i \subseteq \bigcup_{i \in [d_l]} \mathcal{F}_i^1(h_{\boldsymbol{\theta}}^{l:L})$, where on the left-hand side we have a union of d_l hyperplanes, and on the right-hand side we have a union of at most d_l hyperplanes. It follows that $\bigcup_{i \in [d_l]} \mathcal{H}_i = \bigcup_{i \in [d_l]} \mathcal{F}_i^1(h_{\boldsymbol{\theta}}^{l:L})$, and by applying the same argument to $h_{\boldsymbol{\eta}}$, we get $\bigcup_{i \in [d_l]} \mathcal{F}_i^1(h_{\boldsymbol{\theta}}^{l:L}) = \bigcup_{i \in [d_l]} \mathcal{F}_i^1(h_{\boldsymbol{\eta}}^{l:L})$. Therefore there must exist a permutation $\pi : [d_l] \to [d_l]$ such that $\mathcal{F}_i^1(h_{\boldsymbol{\theta}}^{l:L}) = \mathcal{F}_{\pi(i)}^1(h_{\boldsymbol{\eta}}^{l:L})$ for all i. Then by Lemma A.19, there exist scalars $m_1, \ldots m_{d_l}$, such that

$$(\mathbf{W}_l[i,:], b_l[i]) = m_i(\mathbf{W}'_l[\pi(i),:], b'_l[\pi(i)]). \tag{A.68}$$

We know that $m_i \neq 0$ because the folds $\mathcal{F}_i^1(h_{\boldsymbol{\theta}}^{l:L}), \mathcal{F}_i^1(h_{\boldsymbol{\eta}}^{l:L})$, are nonempty; otherwise $\bigcup_{i \in [d_l]} \mathcal{H}_i$ could not be a union of d_l hyperplanes. We have thus shown that there exists a permutation matrix $\mathbf{P}_l \in \mathbb{R}^{d_l \times d_l}$ and a nonzero-entry diagonal matrix $\mathbf{M}_l \in \mathbb{R}^{d_l \times d_l}$ such that $\mathbf{W}_l = \mathbf{M}_l\mathbf{P}_l\mathbf{W}'_l$ and $\mathbf{b}_l = \mathbf{M}_l\mathbf{P}_l\mathbf{b}'_l$.

Next, we show that the diagonal entries of $\mathbf{M}_l$ are positive. Let $\mathbf{x}^-, \mathbf{x}^+ \in \operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}$ be such that $\mathbf{I}^{\boldsymbol{\theta}}(\mathbf{x}^-)$ and $\mathbf{I}^{\boldsymbol{\theta}}(\mathbf{x}^+)$ differ only in $I_l^{\boldsymbol{\theta}}[i,i]$. Wlog, let $I_l^{\boldsymbol{\theta}}[i,i](\mathbf{x}^-) = 0$ and $I_l^{\boldsymbol{\theta}}[i,i](\mathbf{x}^+) = 1$,

and denote the row span of $\mathbf{I}_l^{\boldsymbol{\theta}}(\mathbf{x}^-)\mathbf{W}_l$ by $\mathcal{W}$. Then

$$\nabla_{\mathbf{x}} h_{\boldsymbol{\theta}}^{l:L}(\mathbf{x}^-) = \mathbf{W}_L \mathbf{I}_{L-1}^{\boldsymbol{\theta}}(\mathbf{x}^-) \cdots \mathbf{W}_{l+1} \mathbf{I}_l^{\boldsymbol{\theta}}(\mathbf{x}^-) \mathbf{W}_l \in \mathcal{W},$$
$$\nabla_{\mathbf{x}} h_{\boldsymbol{\theta}}^{l:L}(\mathbf{x}^+) = \mathbf{W}_L \mathbf{I}_{L-1}^{\boldsymbol{\theta}}(\mathbf{x}^+) \cdots \mathbf{W}_{l+1} \mathbf{I}_l^{\boldsymbol{\theta}}(\mathbf{x}^+) \mathbf{W}_l \in \operatorname{span}(\mathcal{W} \cup \mathbf{W}_l[i,:]).$$

Since $h_{\boldsymbol{\theta}}$ is general and $d_l \leq d_{l-1}$, the matrix $\mathbf{W}_l$ has full row rank. This means that the rows of $\mathbf{W}_l$ form a basis, in which the representation of $\nabla_{\mathbf{x}} h_{\boldsymbol{\theta}}^{l:L}(\mathbf{x}^-)$ has one more zero coefficient compared to $\nabla_{\mathbf{x}} h_{\boldsymbol{\theta}}^{l:L}(\mathbf{x}^+)$. In other words, for two points $\mathbf{x}^-, \mathbf{x}^+ \in \operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}$ whose indicators differ only in $I_l^{\boldsymbol{\theta}}[i,i]$, the point for which $I_l^{\boldsymbol{\theta}}[i,i](\mathbf{x}) = 0$ is also the one for which $\nabla_{\mathbf{x}} h_{\boldsymbol{\theta}}^{l:L}(\mathbf{x})$ has more zero coefficients when expressed in the row basis of $\mathbf{W}_l$. Now, observe that if $\mathbf{I}^{\boldsymbol{\theta}}(\mathbf{x}^-)$ and $\mathbf{I}^{\boldsymbol{\theta}}(\mathbf{x}^+)$ differ only in $I_l^{\boldsymbol{\theta}}[i,i]$, then $\mathbf{I}^{\boldsymbol{\eta}}(\mathbf{x}^-)$ and $\mathbf{I}^{\boldsymbol{\eta}}(\mathbf{x}^+)$ differ only in $I_l^{\boldsymbol{\eta}}[\pi(i),\pi(i)]$. Because $\mathbf{W}_l = \mathbf{M}_l \mathbf{P}_l \mathbf{W}_l'$, the number of zero coefficients of $\nabla_{\mathbf{x}} h_{\boldsymbol{\eta}}^{l:L}(\mathbf{x}^-) = \nabla_{\mathbf{x}} h_{\boldsymbol{\theta}}^{l:L}(\mathbf{x}^-)$ in the row basis of $\mathbf{W}_l'$ is the same as the number of zero coefficients of $\nabla_{\mathbf{x}} h_{\boldsymbol{\theta}}^{l:L}(\mathbf{x}^-)$ in the row basis of $\mathbf{W}_l$. It follows that $I_l^{\boldsymbol{\eta}}[\pi(i),\pi(i)](\mathbf{x}^-) = 0$ and $I_l^{\boldsymbol{\eta}}[\pi(i),\pi(i)](\mathbf{x}^+) = 1$. Hence, m_i is positive.

For the inductive step, let $l \in \{2, \ldots, L-1\}$, and assume that there exist permutation matrices $\mathbf{P}_1, \ldots, \mathbf{P}_{l-1}$, and positive-entry diagonal matrices $\mathbf{M}_1, \ldots, \mathbf{M}_{l-1}$, such that (A.65) holds up to layer $l-1$. Then $h_{\boldsymbol{\theta}}^{1:l-1} \equiv \mathbf{M}_{l-1}\mathbf{P}_{l-1} h_{\boldsymbol{\eta}}^{1:l-1}$. Since $h_{\boldsymbol{\theta}}^{1:L} \equiv h_{\boldsymbol{\eta}}^{1:L}$, it follows that

$$h_{\boldsymbol{\theta}}^{l:L}|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}} \equiv \left(h_{\boldsymbol{\eta}}^{l:L} \circ \mathbf{P}_{l-1}^{-1}\mathbf{M}_{l-1}^{-1} \right)\Big|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}} \equiv h_{\tilde{\boldsymbol{\eta}}}^{l:L}|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}}, \tag{A.69}$$

where $\tilde{\boldsymbol{\eta}} := (\mathbf{W}_l' \mathbf{P}_{l-1}^{-1}\mathbf{M}_{l-1}^{-1}, \mathbf{b}_l', \mathbf{W}_{l+1}', \mathbf{b}_{l+1}', \ldots, \mathbf{W}_L', b_L')$. We can therefore apply the same argument to $h_{\boldsymbol{\theta}}^{l:L}|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}}$ and $h_{\tilde{\boldsymbol{\eta}}}^{l:L}|_{\operatorname{int} \mathcal{X}_{\boldsymbol{\theta}}^{l-1}}$ as we presented above for the case $l = 1$. We obtain that there exists a permutation matrix $\mathbf{P}_l \in \mathbb{R}^{d_l \times d_l}$ and a positive-entry diagonal matrix $\mathbf{M}_l \in \mathbb{R}^{d_l \times d_l}$ such that

$$\mathbf{W}_l = \mathbf{M}_l \mathbf{P}_l \mathbf{W}_l' \mathbf{P}_{l-1}^{-1}\mathbf{M}_{l-1}^{-1}, \qquad \mathbf{b}_l = \mathbf{M}_l \mathbf{P}_l \mathbf{b}_l'. \tag{A.70}$$

Finally, consider the last layer. We know that $h_{\boldsymbol{\theta}}^{1:L-1} \equiv \mathbf{M}_{L-1}\mathbf{P}_{L-1} h_{\boldsymbol{\eta}}^{1:L-1}$, which implies $h_{\boldsymbol{\theta}}^{L} \equiv h_{\boldsymbol{\eta}}^{L} \circ \mathbf{P}_{L-1}^{-1}\mathbf{M}_{L-1}^{-1}$, i.e. $h_{\boldsymbol{\theta}}^{L}$ and $h_{\boldsymbol{\eta}}^{L} \circ \mathbf{P}_{L-1}^{-1}\mathbf{M}_{L-1}^{-1}$ are identical linear functions supported on the full-dimensional domain $\mathcal{X}_{\boldsymbol{\theta}}^{L-1}$. It follows that $\mathbf{W}_L = \mathbf{W}_L' \mathbf{P}_{L-1}^{-1}\mathbf{M}_{L-1}^{-1}$ and $b_L = b_L'$. $\square$

APPENDIX B

Proofs for Chapter 3

B.1 Basic lemmas

This section collects a few lemmas useful for proofs. We assume the same setting and notation as in Sections 3.1 and 3.3. In addition, we denote by $\mathbf{P}_{\mathbf{w}}$ the orthogonal projection onto $\operatorname{span}\{\mathbf{x}_i |\, \mathbf{w}^\intercal \mathbf{x}_i = 0\}^\perp$, and by $g : \mathbb{R}^d \to \mathbb{R}^d$,

$$g(\mathbf{w}) \triangleq -\sum_{i=1}^{n} \ell_i'(0) \cdot \mathbb{1}\{\mathbf{w}^\intercal \mathbf{x}_i > 0\}\, \mathbf{P}_{\mathbf{w}} \mathbf{x}_i. \tag{B.1}$$

B.1.1 Lemmas about sectors

The following lemma gives a necessary condition for a vector to be an extremal direction.

Lemma B.1. *If $\mathbf{w} \in \mathbb{S}^{d-1}$ is an extremal direction, then $g(\mathbf{w}) = C\mathbf{w}$ for some constant C.*

Proof. Let $\hat{\mathbf{w}} \in \mathbb{S}^{d-1}$ be a positive extremal direction (the negative case is analogous), and let $\hat{\mathbf{w}} \in \mathcal{S}_{\hat{\boldsymbol{\sigma}}}$. A sector is called *open* if $\sigma_i \neq 0$ for all $i \in [n]$. Denote by $\mathcal{A}(\hat{\boldsymbol{\sigma}})$ the set of all open sectors adjacent to $\hat{\boldsymbol{\sigma}}$,

$$\mathcal{A}(\hat{\boldsymbol{\sigma}}) := \left\{ \boldsymbol{\sigma} \in \{\pm 1\}^n \,\middle|\, \max_{i \in [n]} |\sigma_i - \hat{\sigma}_i| \leq 1 \right\}. \tag{B.2}$$

Since $\hat{\mathbf{w}}$ is a local maximum of G and G is sector-wise linear, $\hat{\mathbf{w}}$ maximises G when constrained to (the closure of) any adjacent sector, i.e. for any $\boldsymbol{\sigma} \in \mathcal{A}(\hat{\boldsymbol{\sigma}})$,

$$\begin{aligned} \hat{\mathbf{w}} = \arg\max_{\mathbf{w}} G(\mathbf{w}), \text{ subject to } \|\mathbf{w}\|^2 = 1, \\ \sigma_i \mathbf{w}^\intercal \mathbf{x}_i \geq 0 \text{ for all } i \in [n]. \end{aligned} \tag{B.3}$$

For $\mathbf{w}$ in the feasible region, G can be treated as a linear function with $\nabla G(\mathbf{w}) = g(\mathbf{w}_{\boldsymbol{\sigma}})$ where $\mathbf{w}_{\boldsymbol{\sigma}}$ is any vector such that $\mathbf{w}_{\boldsymbol{\sigma}} \in \mathcal{S}_{\boldsymbol{\sigma}}$. Hence, the necessary first-order KKT conditions for the problem (B.3) are

$$g(\mathbf{w}_{\boldsymbol{\sigma}}) = C\hat{\mathbf{w}} - \sum_{i=1}^{n} \lambda_i \sigma_i \mathbf{x}_i, \tag{B.4}$$

where $\lambda_i \geq 0$ for all i, but $\lambda_i \neq 0$ requires that the corresponding constraint is tight, $\sigma_i \hat{\mathbf{w}}^\intercal \mathbf{x}_i = 0$. It follows that $\mathbf{P}_{\hat{\mathbf{w}}} \lambda_i \sigma_i \mathbf{x}_i = \mathbf{0}$. Multiplying eq. (B.4) from the left by $\mathbf{P}_{\hat{\mathbf{w}}}$ therefore yields

$$C\hat{\mathbf{w}} = \mathbf{P}_{\hat{\mathbf{w}}}\, g(\mathbf{w}_{\boldsymbol{\sigma}}) = -\sum_{i=1}^{n} \ell_i'(0) \cdot \mathbb{1}\{\sigma_i = 1\}\, \mathbf{P}_{\hat{\mathbf{w}}} \mathbf{x}_i. \tag{B.5}$$

By adjacency, $\sigma_i = \hat{\sigma}_i$ whenever $\hat{\sigma}_i \in \{\pm 1\}$, so they can differ only when $\hat{\sigma}_i = 0$, i.e. when $\mathbf{P}_{\hat{\mathbf{w}}} \mathbf{x}_i = \mathbf{0}$. It follows that

$$C\hat{\mathbf{w}} = -\sum_{i=1}^{n} \ell_i'(0) \cdot \mathbb{1}\{\hat{\sigma}_i = 1\}\, \mathbf{P}_{\hat{\mathbf{w}}} \mathbf{x}_i = g(\hat{\mathbf{w}}). \tag{B.6}$$

□

The following lemma describes the local behaviour of the function G (defined in eq. (3.12)).

Lemma B.2. *For $\mathbf{w} \in \mathbb{S}^{d-1}$ and $\mathbf{v} \in \mathbb{R}^d$, there exists $\epsilon_{\max} > 0$ such that for $\epsilon \in [0, \epsilon_{\max}]$,*

$$G\left(\frac{\mathbf{w} + \epsilon \mathbf{v}}{\|\mathbf{w} + \epsilon \mathbf{v}\|}\right) = \frac{1}{\|\mathbf{w} + \epsilon \mathbf{v}\|}\left(G(\mathbf{w}) - \epsilon \sum_{i=1}^{n} \ell_i'(0)\, \mathbb{1}\{(\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_i > 0\}\, \mathbf{v}^\intercal \mathbf{x}_i\right). \tag{B.7}$$

Proof. Let g be defined as in eq. (B.1); then

$$G\left(\frac{\mathbf{w} + \epsilon \mathbf{v}}{\|\mathbf{w} + \epsilon \mathbf{v}\|}\right) = \frac{1}{\|\mathbf{w} + \epsilon \mathbf{v}\|}(\mathbf{w} + \epsilon \mathbf{v})^\intercal g(\mathbf{w} + \epsilon \mathbf{v}). \tag{B.8}$$

We now analyse $\mathbf{w}^\intercal g(\mathbf{w} + \epsilon \mathbf{v})$ and $\epsilon \mathbf{v}^\intercal g(\mathbf{w} + \epsilon \mathbf{v})$ separately, starting with the former. Denote $I_i^\epsilon := \mathbb{1}\{(\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_i > 0\}$. Then $g(\mathbf{w} + \epsilon \mathbf{v})$ can be written as

$$g(\mathbf{w}+\epsilon\mathbf{v}) = -\sum_{i=1}^{n} \ell_i'(0) \cdot I_i^\epsilon I_i^0\, \mathbf{P}_{\mathbf{w}} \mathbf{x}_i - \sum_{i=1}^{n} \ell_i'(0) \cdot I_i^\epsilon (1 - I_i^0)\, \mathbf{P}_{\mathbf{w}} \mathbf{x}_i - \sum_{i=1}^{n} \ell_i'(0) \cdot I_i^\epsilon\, (\mathbf{P}_{\mathbf{w}+\epsilon\mathbf{v}} - \mathbf{P}_{\mathbf{w}}) \mathbf{x}_i. \tag{B.9}$$

Define

$$\epsilon_{\max} := \frac{1}{2} \max_{\epsilon > 0} \epsilon, \qquad \text{subject to:} \quad \operatorname{sign}\{(\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_i\} \operatorname{sign}\{\mathbf{w}^\intercal \mathbf{x}_i\} \geq 0 \quad \forall i. \tag{B.10}$$

For $\epsilon \in [0, \epsilon_{\max}]$, $I_i^0 = 1$ implies $I_i^\epsilon = 1$, so the first term in eq. (B.17) equals $g(\mathbf{w})$. Regarding the second term, $I_i^\epsilon (1 - I_i^0)$ is nonzero only if $I_i^\epsilon = 1, I_i^0 = 0$. For $\epsilon \in [0, \epsilon_{\max}]$, this can only happen if $(\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_i > 0$ and $\mathbf{w}^\intercal \mathbf{x}_i = 0$. In this scenario however, $\mathbf{P}_{\mathbf{w}} \mathbf{x}_i = \mathbf{0}$, so the second term in eq. (B.17) is zero. Regarding the third term, as long as $\epsilon \in [0, \epsilon_{\max}]$, $(\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_i = 0$ implies $\mathbf{w}^\intercal \mathbf{x}_i = 0$, so

$$\mathbf{w} \in \operatorname{span}\{\mathbf{x}_j |\, \mathbf{w}^\intercal \mathbf{x}_j = 0\}^\perp \subseteq \operatorname{span}\{\mathbf{x}_j |\, (\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_j = 0\}^\perp \tag{B.11}$$

and $\mathbf{w}^\intercal (\mathbf{P}_{\mathbf{w}+\epsilon\mathbf{v}} - \mathbf{P}_{\mathbf{w}}) = \mathbf{w}^\intercal - \mathbf{w}^\intercal = \mathbf{0}^\intercal$. It follows that

$$\mathbf{w}^\intercal g(\mathbf{w} + \epsilon \mathbf{v}) = \mathbf{w}^\intercal g(\mathbf{w}) = G(\mathbf{w}). \tag{B.12}$$

Turning to $\epsilon \mathbf{v}^\intercal g(\mathbf{w} + \epsilon \mathbf{v})$, we have that

$$\epsilon \mathbf{v}^\intercal g(\mathbf{w} + \epsilon \mathbf{v}) = -\epsilon \mathbf{v}^\intercal \sum_{i=1}^{n} \ell_i'(0) \cdot \mathbb{1}\{(\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_i > 0\}\, \mathbf{P}_{\mathbf{w}+\epsilon\mathbf{v}} \mathbf{x}_i, \tag{B.13}$$

where

$$\epsilon \mathbf{v}^\intercal \mathbf{P}_{\mathbf{w}+\epsilon\mathbf{v}} = (\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{P}_{\mathbf{w}+\epsilon\mathbf{v}} - \mathbf{w}^\intercal \mathbf{P}_{\mathbf{w}+\epsilon\mathbf{v}} = (\mathbf{w} + \epsilon \mathbf{v})^\intercal - \mathbf{w}^\intercal = \epsilon \mathbf{v}^\intercal. \tag{B.14}$$

Plugging eq. (B.12) and eq. (B.13) into eq. (B.8) yields the result. □

B.1.2 Training dynamics

In the following lemma, we prove a formula for the evolution of the parameters of a two-layer network trained by gradient flow (eq. (3.4)). The formula has appeared in Maennel et al. [2018] before (but without a proof).

Lemma B.3. *Assume that the training inputs with the zero vector $\{\mathbf{x}_i\}_i \cup \{\mathbf{0}\}$ are in general position.*[1] *Then a two-layer ReLU network trained by gradient flow on $(\mathbf{X}, \mathbf{y})$ satisfies for all $j \in [p]$ and almost all $t \geq 0$,*

$$\frac{\partial a_j}{\partial t} = -\sum_{i=1}^{n} \ell_i'(t) \cdot \rho(\mathbf{w}_j^\intercal \mathbf{x}_i), \tag{B.15}$$

$$\frac{\partial \mathbf{w}_j}{\partial t} = -\sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\Big\{\mathbf{w}_j^\intercal \mathbf{x}_i > 0\Big\}\, a_j \mathbf{P}_{\mathbf{w}_j} \mathbf{x}_i. \tag{B.16}$$

Proof. Fix $\boldsymbol{\theta}$, and denote by $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} \in \{-1, 0, 1\}^{p \times n}$ the activation matrix for $f_{\boldsymbol{\theta}}$, $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}[j,i] \triangleq \operatorname{sign} \mathbf{w}_j^\intercal \mathbf{x}_i$. Then for any sequence $\boldsymbol{\theta}_k \to \boldsymbol{\theta}$ such that $\{\nabla \ell(\boldsymbol{\theta}_k)\}$ exists and has a limit,

$$\lim_{k\to\infty} \boldsymbol{\Sigma}_{\boldsymbol{\theta}_k} \in \Big\{\boldsymbol{\Sigma} \in \{\pm 1\}^{p\times n} \,\Big|\, \boldsymbol{\Sigma}[j,i] = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}[j,i] \text{ if } \boldsymbol{\Sigma}_{\boldsymbol{\theta}}[j,i] \neq 0\Big\}. \tag{B.17}$$

Conversely, for any $\boldsymbol{\Sigma}$ in the set above, there exists a sequence $\boldsymbol{\theta}_k \to \boldsymbol{\theta}$ such that $\lim_{k\to\infty} \boldsymbol{\Sigma}_{\boldsymbol{\theta}_k} = \boldsymbol{\Sigma}$. To see this, observe that each $\mathbf{w}_j$ can be approached separately. Let $\mathbf{A}$ be the matrix whose rows are formed by those $\mathbf{x}_i$ for which $\mathbf{w}_j^\intercal \mathbf{x}_i = 0$. Then by the general position of inputs, $\mathbf{A}$ is a wide full-rank matrix and $\mathbf{A}\mathbf{w}_j = \mathbf{0}$. It follows that for any $\boldsymbol{\epsilon}$, $\mathbf{A}\mathbf{w} = \boldsymbol{\epsilon}$ has a solution, which can be chosen convergent to $\mathbf{w}_j$ as $\boldsymbol{\epsilon} \to \mathbf{0}$.

We deduce that

$$\partial \ell(\boldsymbol{\theta}(t)) = \operatorname{conv}\Big\{\mathbf{g}(\boldsymbol{\Sigma}) \,|\, \boldsymbol{\Sigma}[j,i] \in \{\pm 1\},\, \boldsymbol{\Sigma}[j,i] = \boldsymbol{\Sigma}_{\boldsymbol{\theta}(t)}[j,i] \text{ if } \boldsymbol{\Sigma}_{\boldsymbol{\theta}(t)}[j,i] \neq 0\Big\}, \tag{B.18}$$

where we define $\mathbf{g}(\boldsymbol{\Sigma})$ coordinate-wise as

$$\begin{aligned} \mathbf{g}(\boldsymbol{\Sigma})[a_j] &:= \sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\{\boldsymbol{\Sigma}[j,i] = 1\}\, \mathbf{w}_j^\intercal \mathbf{x}_i, \\ \mathbf{g}(\boldsymbol{\Sigma})[\mathbf{w}_j] &:= \sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\{\boldsymbol{\Sigma}[j,i] = 1\}\, a_j \mathbf{x}_i. \end{aligned} \tag{B.19}$$

Since the value of $g(\boldsymbol{\Sigma})[a_j]$ is independent of $\boldsymbol{\Sigma}$, this proves eq. (B.15).

The proof of eq. (B.16) is slightly more complicated, as we need to pin down a single member of $\partial \ell(\boldsymbol{\theta}(t))$. To do that, we recall a result by Davis et al. [2020], who show that for a large class of deep learning scenarios (which includes ours), the objective ℓ admits a chain rule, i.e.

$$\ell'(t) = \left\langle \partial \ell(\boldsymbol{\theta}(t)), \frac{\partial \boldsymbol{\theta}}{\partial t} \right\rangle \qquad \text{for almost all } t \geq 0, \tag{B.20}$$

where the right-hand side above should be interpreted as the only element of the set $\{\mathbf{q}^\intercal \partial\boldsymbol{\theta}/\partial t \,|\, \mathbf{q} \in \partial \ell(\boldsymbol{\theta}(t))\}$. For t such that both eq. (3.4) and eq. (B.20) hold,

$$0 = \left\langle \partial \ell(\boldsymbol{\theta}(t)) - \partial \ell(\boldsymbol{\theta}(t)), \frac{\partial \boldsymbol{\theta}}{\partial t} \right\rangle, \tag{B.21}$$

[1] That is, no k of these points lie on a $(k-2)$-dimensional hyperplane, for all $k \geq 2$.

implying that

$$\frac{\partial \boldsymbol{\theta}}{\partial t} \in \operatorname{span}\{\partial \ell(\boldsymbol{\theta}(t)) - \partial \ell(\boldsymbol{\theta}(t)\}^{\perp}. \tag{B.22}$$

Suppose $\mathbf{q}_1, \mathbf{q}_2$ satisfy both eq. (3.4) and eq. (B.22) (taking the role of $\partial\boldsymbol{\theta}/\partial t$). Then

$$\mathbf{q}_1 - \mathbf{q}_2 \in (\partial \ell(\boldsymbol{\theta}(t)) - \partial \ell(\boldsymbol{\theta}(t))) \cap \operatorname{span}\{\partial \ell(\boldsymbol{\theta}(t)) - \partial \ell(\boldsymbol{\theta}(t)\}^{\perp} = \{\mathbf{0}\}. \tag{B.23}$$

It follows that $\partial\boldsymbol{\theta}/\partial t$ is the unique member of both $\operatorname{span}\{\partial \ell(\boldsymbol{\theta}(t)) - \partial \ell(\boldsymbol{\theta}(t)\}^{\perp}$ and $-\partial \ell(\boldsymbol{\theta}(t))$. By eqs. (B.18) and (B.19),

$$\operatorname{span}\{\partial \ell(\boldsymbol{\theta}(t)) - \partial \ell(\boldsymbol{\theta}(t)\} \supseteq \operatorname{span}\Big\{\boldsymbol{\xi}_{ij} \,|\, \mathbf{w}_j^\intercal \mathbf{x}_i = 0\Big\}, \tag{B.24}$$

where $\boldsymbol{\xi}_{ij}[\mathbf{w}_j] = \mathbf{x}_i$ and all other elements of $\boldsymbol{\xi}_{ij}$ are zero. Since

$$\frac{\partial \boldsymbol{\theta}}{\partial t} \in \operatorname{span}\{\partial \ell(\boldsymbol{\theta}(t)) - \partial \ell(\boldsymbol{\theta}(t)\}^{\perp} \subseteq \operatorname{span}\Big\{\boldsymbol{\xi}_{ij} \,|\, \mathbf{w}_j^\intercal \mathbf{x}_i = 0\Big\}^{\perp}, \tag{B.25}$$

we obtain that for all (i, j) with $\mathbf{w}_j^\intercal \mathbf{x}_i = 0$, $\partial \mathbf{w}_j/\partial t \perp \mathbf{x}_i$. In other words,

$$\frac{\partial \mathbf{w}_j}{\partial t} = \mathbf{P}_{\mathbf{w}_j} \frac{\partial \mathbf{w}_j}{\partial t} \in \operatorname*{conv}_{\boldsymbol{\Sigma}} \left\{ -\sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\{\boldsymbol{\Sigma}[j,i] = 1\}\, a_j \mathbf{P}_{\mathbf{w}_j} \mathbf{x}_i \right\}, \tag{B.26}$$

where the inclusion follows from $\partial\boldsymbol{\theta}/\partial t \in -\partial \ell(\boldsymbol{\theta}(t))$ and eqs. (B.18) and (B.19). Now observe that by definition, $\mathbf{P}_{\mathbf{w}_j}\mathbf{x}_i = \mathbf{0}$ for all (i, j) with $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}[j, i] = 0$, hence the set in eq. (B.26) is a singleton whose only element equals eq. (B.16). □

The following lemma shows that a balanced two-layer network remains balanced and neurons keep their signs.

Lemma B.4. *If a two-layer neural network is balanced at initialisation and trained by gradient flow with a loss whose derivative is bounded, then for* $t \geq 0$*,*

$$a_j(t) = \operatorname{sign} a_j(0) \cdot \|\mathbf{w}_j(t)\|. \tag{B.27}$$

Proof. By Lemma B.3, for almost all $t \geq 0$,

$$\frac{\partial \|\mathbf{w}_j\|^2}{\partial t} = -\sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\Big\{\mathbf{w}_j^\intercal \mathbf{x}_i > 0\Big\}\, a_j \mathbf{w}_j^\intercal \mathbf{x}_i = \frac{\partial a_j^2}{\partial t}, \tag{B.28}$$

i.e. a_j and $\mathbf{w}_j$ grow equally fast. Since $|a_j(0)| = \|\mathbf{w}_j(0)\|$ at initialisation, $|a_j(t)| = \|\mathbf{w}_j(t)\|$ throughout training.

Next denote by $B, V > 0$ some scalars such that $|\ell_i'(u)| \leq B$ for all $i \in [n]$ and $u \in \mathbb{R}$, and $\|\mathbf{x}_i\| \leq V$ for all $i \in [n]$. Then $\left|\partial a_j^2/\partial t\right| \leq nBa_j^2V$, or equivalently $\left|\partial \log a_j^2/\partial t\right| \leq nBV$. It follows that $a_j^2(t)$ lies between $a_j^2(0)\exp(-nBVt)$ and $a_j^2(0)\exp(nBVt)$, and hence a_j cannot cross zero in finite time, proving eq. (B.27). □

B.2 Proofs of main results

Lemma 3.2. *In the setting of Theorem 3.1, there is exactly one positive extremal direction and exactly one negative extremal direction. The positive extremal sector $\boldsymbol{\sigma}^+$ is given by*

$$\sigma_j^+ = \begin{cases} 1, & \text{if } y_j = 1, \\ -1, & \text{if } y_j = -1 \text{ and } \mathbf{x}_j^\intercal \mathbf{x}_i < 0 \text{ for some } i \text{ with } y_i = 1, \\ 0, & \text{if } y_j = -1 \text{ and } \mathbf{x}_j^\intercal \mathbf{x}_i = 0 \text{ for all } i \text{ with } y_i = 1, \end{cases} \tag{3.13}$$

and the negative extremal sector $\boldsymbol{\sigma}^-$ is given by

$$\sigma_j^- = \begin{cases} 1, & \text{if } y_j = -1, \\ -1, & \text{if } y_j = 1 \text{ and } \mathbf{x}_j^\intercal \mathbf{x}_i < 0 \text{ for some } i \text{ with } y_i = -1, \\ 0, & \text{if } y_j = 1 \text{ and } \mathbf{x}_j^\intercal \mathbf{x}_i = 0 \text{ for all } i \text{ with } y_i = -1. \end{cases} \tag{3.14}$$

Proof. We will prove the positive case; the negative case follows by inverting all labels. Because G is a continuous function on a compact domain, it has a maximum. At least one maximum must moreover be strict, or otherwise G would have to be constant. This shows that a positive extremal direction exists; we now show there is no more than one such direction.

By Lemma B.1, there cannot be more than one extremal direction per sector; it therefore suffices to show that no sector except one, $\boldsymbol{\sigma}^+$, admits a positive extremal direction. We will show that if $\mathbf{w} \in \mathbb{S}^{d-1}$ lies in any sector other than $\boldsymbol{\sigma}^+$, then $\mathbf{w}$ is not positive extremal; in particular we show that $G(\mathbf{w})$ can be locally increased.

Let $\boldsymbol{\sigma} \neq \boldsymbol{\sigma}^+$ and let $\mathbf{w} \in \mathcal{S}_{\boldsymbol{\sigma}} \cap \mathbb{S}^{d-1}$. By Lemma B.2, for any $\mathbf{v} \in \mathbb{R}^d$ there exists $\epsilon_{\max} > 0$ such that for $\epsilon \in (0, \epsilon_{\max}]$,

$$G\left(\frac{\mathbf{w} + \epsilon \mathbf{v}}{\|\mathbf{w} + \epsilon \mathbf{v}\|}\right) = \frac{G(\mathbf{w}) + \epsilon\alpha}{\|\mathbf{w} + \epsilon \mathbf{v}\|}, \tag{B.29}$$

where

$$\alpha := -\sum_{i=1}^{n} \ell_i'(0)\, \mathbb{1}\{(\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_i > 0\} \mathbf{v}^\intercal \mathbf{x}_i. \tag{B.30}$$

We now analyse the different possible realisations of $\boldsymbol{\sigma}$, and for each we find $\mathbf{v} \in \mathbb{R}^d$ such that $(G(\mathbf{w}) + \epsilon\alpha)/\|\mathbf{w} + \epsilon \mathbf{v}\| > G(\mathbf{w})$ for small ϵ.

Suppose first that $\sigma_j = -1$ for some example with $y_j = 1$, or that $\sigma_j = 1$ for some example with $y_j = -1$. Then set $\mathbf{v} := y_j \mathbf{x}_j / \|\mathbf{x}_j\|$. By orthogonal separability, we have that $\alpha \geq 0$. Also, $\sigma_j \triangleq \operatorname{sign} \mathbf{w}^\intercal \mathbf{x}_j = -y_j$ implies $\mathbf{w}^\intercal \mathbf{v} < 0$, therefore $\|\mathbf{w} + \epsilon \mathbf{v}\| < \|\mathbf{w}\| = 1$ for ϵ small enough. It follows that $(G(\mathbf{w}) + \epsilon\alpha)/\|\mathbf{w} + \epsilon \mathbf{v}\| > G(\mathbf{w})$.

Next suppose that $\sigma_j = 0$ for some example with $y_j = 1$, and set $\mathbf{v} := \mathbf{x}_j / \|\mathbf{x}_j\|$. Then $\alpha > 0$, because each term in eq. (B.30) is non-negative by orthogonal separability, and the term corresponding to $i = j$ is strictly positive:

$$-\ell_j'(0)\, \mathbb{1}\{(\mathbf{w} + \epsilon \mathbf{v})^\intercal \mathbf{x}_j > 0\}\, \mathbf{v}^\intercal \mathbf{x}_j = -\ell_j'(0)\, \mathbb{1}\{\epsilon \|\mathbf{x}_j\| > 0\} \|\mathbf{x}_j\| > 0. \tag{B.31}$$

From $\sigma_j \triangleq \operatorname{sign} \mathbf{w}^\intercal \mathbf{x}_j = 0$ it further follows that $\|\mathbf{w} + \epsilon \mathbf{v}\| = \sqrt{1 + \epsilon^2}$. Hence, $(G(\mathbf{w}) + \epsilon\alpha)/\|\mathbf{w} + \epsilon \mathbf{v}\| = (G(\mathbf{w}) + \epsilon\alpha)/(1 + O(\epsilon^2))$, which strictly exceeds $G(\mathbf{w})$ for ϵ small enough.

We have thus shown that if $\boldsymbol{\sigma}$ is positive extremal, then necessarily $\sigma_i = 1$ for all examples with $y_i = 1$, and $\sigma_i \in \{0, -1\}$ for examples with $y_i = -1$. Suppose now that $\sigma_j = 0$ for an example with $y_j = -1$ that satisfies $\mathbf{x}_j^\intercal \mathbf{x}_k < 0$ for some k with $y_k = 1$. Taking $\mathbf{v} := -\mathbf{x}_j / \|\mathbf{x}_j\|$ will make α strictly positive, as the term corresponding to $i = k$ in eq. (B.30) will be strictly positive (this term's indicator equals 1, as we know from the above that $\sigma_k = 1$). Like in the previous paragraph, $\|\mathbf{w} + \epsilon\mathbf{v}\| = 1 + O(\epsilon^2)$, which suffices to show $G(\mathbf{w})$ is locally submaximal.

Finally, let $\mathbf{x}_j$ be such that $y_j = -1$ and $\mathbf{x}_j^\intercal \mathbf{x}_i = 0$ for all i with $y_i = 1$, and suppose that $\sigma_j = -1$. With $\mathbf{v} := \mathbf{P}_{\mathbf{w}} \mathbf{x}_j / \|\mathbf{P}_{\mathbf{w}} \mathbf{x}_j\|$ (we know that $\mathbf{P}_{\mathbf{w}} \mathbf{x}_j \neq \mathbf{0}$ because $\mathbf{w}^\intercal \mathbf{x}_j \neq 0$ by $\sigma_j = -1$), we have

$$\alpha = -\frac{1}{\|\mathbf{P}_{\mathbf{w}} \mathbf{x}_j\|} \sum_{i=1}^{n} \ell_i'(0)\, \mathbb{1}\{(\mathbf{w} + \epsilon\mathbf{v})^\intercal \mathbf{x}_i > 0\}\, \mathbf{x}_j^\intercal \mathbf{P}_{\mathbf{w}} \mathbf{x}_i. \tag{B.32}$$

We claim that each term in eq. (B.32) is zero: For terms with $\sigma_i = -1$, the indicator $\mathbb{1}\{(\mathbf{w} + \epsilon\mathbf{v})^\intercal \mathbf{x}_i > 0\}$ is zero. For terms with $\sigma_i = 0$, $\mathbf{P}_{\mathbf{w}} \mathbf{x}_i = \mathbf{0}$. (Also notice that such terms necessarily satisfy $y_i = -1$ and $\mathbf{x}_i^\intercal \mathbf{x}_k = 0$ for all k with $y_k = 1$, which we will need shortly.) Lastly, for terms with $\sigma_i = 1$, we know $y_i = 1$, and hence $\mathbf{x}_i^\intercal \mathbf{x}_l = 0$ for all l with $\sigma_l = 0$. In other words, $\mathbf{x}_i \perp \operatorname{span}\{\mathbf{x}_l \,|\, \mathbf{w}^\intercal \mathbf{x}_l = 0\}$, implying $\mathbf{P}_{\mathbf{w}} \mathbf{x}_i = \mathbf{x}_i$. In the context of eq. (B.32), we obtain $\mathbf{x}_j^\intercal \mathbf{P}_{\mathbf{w}} \mathbf{x}_i = \mathbf{x}_j^\intercal \mathbf{x}_i = 0$, concluding the proof that $\alpha = 0$. Since $\sigma_j = -1$, $\|\mathbf{w} + \epsilon\mathbf{v}\| < \|\mathbf{w}\| = 1$ for small enough ϵ, and $(G(\mathbf{w}) + \epsilon\alpha)/\|\mathbf{w} + \epsilon\mathbf{v}\| > G(\mathbf{w})$. We have thus ruled out all sectors except $\boldsymbol{\sigma}^+$, proving that for orthogonally separable datasets there is a unique positive extremal sector. □

Lemma 3.3. *Assume the setting of Theorem 3.1. If at time T the neuron $(a_j, \mathbf{w}_j)$ satisfies $a_j(T) > 0$ and $\mathbf{w}_j(T) \in \mathcal{S}_{\boldsymbol{\sigma}}$, where $\boldsymbol{\sigma}$ is the positive extremal sector (eq.* (3.13)*), then for $t \geq T$, $\mathbf{w}_j(t) \in \mathcal{S}_{\boldsymbol{\sigma}}$. The same holds if $a_j(T) < 0$ and $\boldsymbol{\sigma}$ is the negative extremal sector (eq.* (3.14)*).*

Proof. We omit the neuron index, and only prove the positive case; the negative case is analogous. Denote $\boldsymbol{\sigma}(t) := \operatorname{sign}(\mathbf{X}^\intercal \mathbf{w}(t))$. We proceed by contradiction. Suppose there exists a time $T_1 > T$ such that $\boldsymbol{\sigma}(T_1) \neq \boldsymbol{\sigma}(T)$. Wlog, take T_1 such that $\boldsymbol{\sigma}(t)$ is constant on (T, T_1) and denote this constant sector $\bar{\boldsymbol{\sigma}}$; by continuity $\bar{\sigma}_k = \sigma_k(T)$ if $\sigma_k(T) \neq 0$.

Now consider $\sigma_k(T) = 0$. By the gradient flow differential inclusion, for almost all $t \in (T, T_1)$,

$$\frac{\partial \mathbf{w}^\intercal \mathbf{x}_k}{\partial t} \in \operatorname*{conv}_{\boldsymbol{\sigma}'} \left\{ -\sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\{\sigma_i' = 1\}\, a \mathbf{x}_i^\intercal \mathbf{x}_k \right\}, \tag{B.33}$$

where each $\boldsymbol{\sigma}'$ in the definition of the convex hull satisfies $\sigma_i' = \sigma_i(T)$ if $\sigma_i(T) \neq 0$, implying

$$\begin{aligned} \{i \,|\, \sigma_i' = 1\} &\subseteq \{i \,|\, \sigma_i(T) = 1\} \cup \{i \,|\, \sigma_i(T) = 0\} \\ &= \{i \,|\, y_i = 1\} \cup \{i \,|\, y_i = -1 \text{ and } \mathbf{x}_i^\intercal \mathbf{x}_j = 0 \text{ for all } j \text{ with } y_j = 1\}. \end{aligned} \tag{B.34}$$

Denote the two sets in the last expression $\mathcal{I}^+$ and $\mathcal{I}^0$, and consider the gradient corresponding to some $\boldsymbol{\sigma}'$ in eq. (B.33). The gradient terms corresponding to $i \in \mathcal{I}^+$ are zero (because $k \in \mathcal{I}^0$ and so $\mathbf{x}_i^\intercal \mathbf{x}_k = 0$) and the terms corresponding to $i \in \mathcal{I}^0$ (if there are any) are negative. The total gradient for $\boldsymbol{\sigma}'$ is therefore non-positive, which is preserved under taking convex hulls, and so we obtain $\frac{\partial}{\partial t} \mathbf{w}^\intercal \mathbf{x}_k \leq 0$. It follows that $\bar{\sigma}_k \neq 1$.

By Lemma B.3, for almost all $t \in (T, T_1)$ and any $k \in [n]$,

$$\frac{\partial \mathbf{w}^\intercal \mathbf{x}_k}{\partial t} = -\sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\{\bar{\sigma}_i = 1\}\, a \mathbf{x}_i^\intercal \mathbf{P}_{\mathbf{w}} \mathbf{x}_k, \tag{B.35}$$

where $\mathbb{1}\{\bar{\sigma}_i = 1\} = \mathbb{1}\{y_i = 1\}$ as we have shown above. Observe that for $\mathbf{x}_i$ with $y_i = 1$, we have $\mathbf{P}_{\mathbf{w}} \mathbf{x}_i = \mathbf{x}_i$. This is because $\mathbf{P}_{\mathbf{w}}$ projects onto

$$\operatorname{span}\{\mathbf{x}_i \,|\, \bar{\sigma}_i = 0\}^\perp \supseteq \operatorname{span}\{\mathbf{x}_i \,|\, \sigma_i(T) = 0\}^\perp \tag{B.36}$$

and $\mathbf{x}_i$ lies in the right-hand side by the definition of positive extremal sector (eq. (3.13)). Therefore

$$\frac{\partial \mathbf{w}^\intercal \mathbf{x}_k}{\partial t} = -\sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\{y_i = 1\}\, a \mathbf{x}_i^\intercal \mathbf{x}_k. \tag{B.37}$$

One can easily check that if $\sigma_k(T) = 1$ then $\frac{\partial}{\partial t} \mathbf{w}^\intercal \mathbf{x}_k > 0$, if $\sigma_k(T) = -1$ then $\frac{\partial}{\partial t} \mathbf{w}^\intercal \mathbf{x}_k < 0$, and if $\sigma_k(T) = 0$ then $\frac{\partial}{\partial t} \mathbf{w}^\intercal \mathbf{x}_k = 0$. It follows that $\boldsymbol{\sigma}(T_1) = \boldsymbol{\sigma}(T)$, which is a contradiction. □

Lemma B.5. *Assume the setting of Theorem 3.1. If at time T the neuron $(a_j, \mathbf{w}_j)$ satisfies $a_j(T) > 0$ and $\mathbf{w}_j(T)^\intercal \mathbf{x}_k > 0$ for some $k \in [n]$ with $y_k = 1$, then for $t \geq T$, $\mathbf{w}_j(t)^\intercal \mathbf{x}_k > 0$. The same holds if instead $a_j(T) < 0$ and $y_k = -1$.*

Proof. We omit the neuron index, and only prove the positive case; the negative case is analogous. We will show that for almost all $t \in (T, \infty)$, $\partial \mathbf{w}^\intercal \mathbf{x}_k / \partial t \geq 0$. By the gradient flow differential inclusion, for almost all $t \in (T, \infty)$,

$$\frac{\partial \mathbf{w}^\intercal \mathbf{x}_k}{\partial t} \in \operatorname*{conv}_{\boldsymbol{\sigma}'} \left\{ -\sum_{i=1}^{n} \ell_i'(t)\, \mathbb{1}\{\sigma_i' = 1\}\, a \mathbf{x}_i^\intercal \mathbf{x}_k \right\}. \tag{B.38}$$

Fix any $\boldsymbol{\sigma}'$ and consider the summand corresponding to example i. If $y_i = 1$, then $-\ell_i'(t) > 0$ and $\mathbf{x}_i^\intercal \mathbf{x}_k > 0$, so the summand is non-negative. If $y_i = -1$, then $-\ell_i'(t) < 0$ and $\mathbf{x}_i^\intercal \mathbf{x}_k \leq 0$, so the summand is again non-negative. It follows that the sum is non-negative irrespective of $\boldsymbol{\sigma}'$, hence $\partial \mathbf{w}^\intercal \mathbf{x}_k / \partial t \geq 0$. □

Corollary 3.1. *Under the conditions of Theorem 3.1, there exist constants $u, z \geq 0$ such that*

$$\frac{h_{\boldsymbol{\theta}(t)}(\mathbf{x})}{\|\boldsymbol{\theta}(t)\|^2} \to u\rho(\mathbf{w}_+^\intercal \mathbf{x}) - z\rho(\mathbf{w}_-^\intercal \mathbf{x}), \qquad \text{as } t \to \infty. \tag{3.10}$$

Proof. By Lemma B.4, $\|\boldsymbol{\theta}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{W}\|_F^2 = 2\|\mathbf{a}\|^2 = 2\,\|\mathbf{W}\|_F^2$. Then for any $\mathbf{x} \in \mathbb{R}^d$,

$$\frac{2h_{\boldsymbol{\theta}(t)}(\mathbf{x})}{\|\boldsymbol{\theta}(t)\|^2} = \frac{\mathbf{a}(t)^\intercal}{\|\mathbf{a}(t)\|}\, \rho\left(\frac{\mathbf{W}(t)\mathbf{x}}{\|\mathbf{W}(t)\|_F}\right). \tag{B.39}$$

Denote $\tilde{\mathbf{a}} := \lim_{t\to\infty} \mathbf{a}(t)/\|\mathbf{a}(t)\|$. Then by Theorem 3.1, as $t \to \infty$,

$$\frac{2h_{\boldsymbol{\theta}(t)}(\mathbf{x})}{\|\boldsymbol{\theta}(t)\|^2} \to \tilde{\mathbf{a}}^\intercal \rho(\mathbf{u}\mathbf{w}_+^\intercal \mathbf{x} + \mathbf{z}\mathbf{w}_-^\intercal \mathbf{x}) \tag{B.40}$$

$$= \tilde{\mathbf{a}}^\intercal \rho(\mathbf{u}\mathbf{w}_+^\intercal \mathbf{x}) + \tilde{\mathbf{a}}^\intercal \rho(\mathbf{z}\mathbf{w}_-^\intercal \mathbf{x}) \tag{B.41}$$

$$= \tilde{\mathbf{a}}^\intercal \mathbf{u}\, \rho(\mathbf{w}_+^\intercal \mathbf{x}) + \tilde{\mathbf{a}}^\intercal \mathbf{z}\, \rho(\mathbf{w}_-^\intercal \mathbf{x}), \tag{B.42}$$

where in eq. (B.41) we used the fact that for all $i \in [p]$, either $u_i = 0$ or $z_i = 0$, and in eq. (B.42) we used $\mathbf{u}, \mathbf{z} \geq \mathbf{0}$. Finally, since $\tilde{\mathbf{a}} = \mathbf{u}\|\mathbf{w}_+\| - \mathbf{z}\|\mathbf{w}_-\|$, we have

$$\tilde{\mathbf{a}}^\intercal \mathbf{u} = \|\mathbf{w}_+\|\mathbf{u}^\intercal\mathbf{u} \geq 0, \tag{B.43}$$

$$\tilde{\mathbf{a}}^\intercal \mathbf{z} = -\|\mathbf{w}_-\|\mathbf{z}^\intercal\mathbf{z} \leq 0, \tag{B.44}$$

which completes the proof. □

B.3 Relationship to nonlinear max-margin

Lemma B.6. *Let $(\mathbf{X}, \mathbf{y})$ be an orthogonally separable dataset, let $\mathbf{w}_+, \mathbf{w}_-$ be defined as in eqs.* (3.6) *and* (3.7) *and let*

$$\widetilde{\mathbf{W}} := \mathbf{u}\mathbf{w}_+^\intercal + \mathbf{z}\mathbf{w}_-^\intercal, \tag{B.45}$$

$$\tilde{\mathbf{a}} := \|\mathbf{w}_+\|\mathbf{u} - \|\mathbf{w}_-\|\mathbf{z}, \tag{B.46}$$

for some $\mathbf{u}, \mathbf{z} \in \mathbb{R}^p_+$ such that $u_i = 0$ or $z_i = 0$ for all $i \in [p]$. Also let $\mathbf{u}, \mathbf{z}$ be normalised such that $\|\mathbf{u}\| = \|\mathbf{w}_+\|^{-1/2}$ and $\|\mathbf{z}\| = \|\mathbf{w}_-\|^{-1/2}$. Then $\tilde{\boldsymbol{\theta}} \triangleq (\widetilde{\mathbf{W}}, \tilde{\mathbf{a}})$ is a KKT point of the following constrained optimisation problem:

$$\min \frac{1}{2}\|\boldsymbol{\theta}\|^2, \qquad \textit{s.t.} \qquad y_i h_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq 1, \quad i \in [n]. \tag{B.47}$$

Proof. By standard linear max-margin considerations (e.g. Hastie et al. [2008, Section 4.5.2]), we know that

$$\mathbf{w}_+ = \sum_{i:y_i=1} \alpha_i \mathbf{x}_i, \qquad \mathbf{w}_- = \sum_{i:y_i=-1} \alpha_i \mathbf{x}_i, \tag{B.48}$$

for some $\alpha_i \geq 0$ such that $\alpha_i = 0$ if $\mathbf{x}_i$ is a non-support vector. It follows by orthogonal separability that for i with $y_i = 1$,

$$\mathbf{w}_+^\intercal \mathbf{x}_i > 0, \qquad y_i \mathbf{w}_-^\intercal \mathbf{x}_i \leq 0, \tag{B.49}$$

and for i with $y_i = -1$,

$$\mathbf{w}_+^\intercal \mathbf{x}_i \leq 0, \qquad y_i \mathbf{w}_-^\intercal \mathbf{x}_i > 0; \tag{B.50}$$

we will need these properties shortly.

Let us now turn to checking the KKT status of $\tilde{\boldsymbol{\theta}}$ wrt. eq. (B.47). We start by showing that $\tilde{\boldsymbol{\theta}}$ is feasible. Let $\mathbf{x}_i$ be such that $y_i = 1$; then

$$y_i h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_i) = \tilde{\mathbf{a}}^\intercal \rho(\widetilde{\mathbf{W}}\mathbf{x}_i) \tag{B.51}$$

$$= \tilde{\mathbf{a}}^\intercal \rho(\mathbf{u}\mathbf{w}_+^\intercal \mathbf{x}_i + \mathbf{z}\mathbf{w}_-^\intercal \mathbf{x}_i) \tag{B.52}$$

$$= \tilde{\mathbf{a}}^\intercal \mathbf{u}\, \mathbf{w}_+^\intercal \mathbf{x}_i \tag{B.53}$$

$$= \|\mathbf{w}_+\|\|\mathbf{u}\|^2 \mathbf{w}_+^\intercal \mathbf{x}_i \tag{B.54}$$

$$= \mathbf{w}_+^\intercal \mathbf{x}_i \geq 1, \tag{B.55}$$

where the last inequality follows from the definition of $\mathbf{w}_+$, eq. (3.6). Similarly, if $\mathbf{x}_i$ is such that $y_i = -1$, then

$$y_i h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_i) = -\tilde{\mathbf{a}}^\intercal \mathbf{z}\, \mathbf{w}_-^\intercal \mathbf{x}_i \tag{B.56}$$

$$= \|\mathbf{w}_-\| \|\mathbf{z}\|^2 \mathbf{w}_-^\intercal \mathbf{x}_i \tag{B.57}$$

$$= \mathbf{w}_-^\intercal \mathbf{x}_i \geq 1. \tag{B.58}$$

This shows that $\tilde{\boldsymbol{\theta}}$ is feasible.

Next, we show that $\tilde{\boldsymbol{\theta}}$ is a KKT point, i.e. we show that there exist $\lambda_1, \ldots, \lambda_n \geq 0$ such that

1. for all $i \in [n]$, $\lambda_i (y_i h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_i) - 1) = 0$, and
2. $\tilde{\boldsymbol{\theta}} \in \sum_{i=1}^n \lambda_i y_i \partial_{\boldsymbol{\theta}} h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_i)$,

where $\partial_{\boldsymbol{\theta}} h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x})$ denotes the Clarke subdifferential of $h_{\boldsymbol{\theta}}(\mathbf{x})$ wrt. $\boldsymbol{\theta}$, evaluated at $\tilde{\boldsymbol{\theta}}$. Specifically, we show that the choice

$$\lambda_i = \begin{cases} \alpha_i / \|\mathbf{w}_+\|, & \text{if } \; y_i = 1, \\ \alpha_i / \|\mathbf{w}_-\|, & \text{if } \; y_i = -1, \end{cases} \tag{B.59}$$

satisfies both conditions above.

As for the first condition, observe that if i is a non-support example, then $\lambda_i = \alpha_i = 0$ and the condition holds. If i is a support example, then by eqs. (B.55) and (B.58), $y_i h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_i) = 1$ and the condition holds as well.

As for the second condition, denote

$$g_i(\boldsymbol{\theta}) := [\mathbf{I}_{\boldsymbol{\theta}}(\mathbf{x}_i) \mathbf{a} \mathbf{x}_i^\intercal ;\; \rho(\mathbf{W}\mathbf{x}_i)], \tag{B.60}$$

where $\mathbf{I}_{\boldsymbol{\theta}}(\mathbf{x}) = \operatorname{diag}[\mathbb{1}\{\mathbf{W}\mathbf{x}_i > \mathbf{0}\}] \in \mathbb{R}^{p \times p}$ is the diagonal matrix whose (i,i)-element is one if $\mathbf{w}_i^\intercal \mathbf{x} > 0$ and zero otherwise. It holds that $g_i(\tilde{\boldsymbol{\theta}}) \in \partial_{\boldsymbol{\theta}} h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_i)$ and

$$\begin{aligned}
\sum_{i=1}^n \lambda_i y_i g_i\big(\tilde{\boldsymbol{\theta}}\big) &= \sum_{i:y_i=1} \frac{\alpha_i}{\|\mathbf{w}_+\|} [\operatorname{diag}[\mathbb{1}\{\mathbf{u} > \mathbf{0}\}] \tilde{\mathbf{a}} \mathbf{x}_i^\intercal ;\; \rho(\mathbf{u}\mathbf{w}_+^\intercal \mathbf{x}_i + \mathbf{z}\mathbf{w}_-^\intercal \mathbf{x}_i)] \\
&\quad - \sum_{i:y_i=-1} \frac{\alpha_i}{\|\mathbf{w}_-\|} [\operatorname{diag}[\mathbb{1}\{\mathbf{z} > \mathbf{0}\}] \tilde{\mathbf{a}} \mathbf{x}_i^\intercal ;\; \rho(\mathbf{u}\mathbf{w}_+^\intercal \mathbf{x}_i + \mathbf{z}\mathbf{w}_-^\intercal \mathbf{x}_i)] \\
&= \sum_{i:y_i=1} \frac{\alpha_i}{\|\mathbf{w}_+\|} [\|\mathbf{w}_+\| \mathbf{u} \mathbf{x}_i^\intercal ;\; \mathbf{u}\mathbf{w}_+^\intercal \mathbf{x}_i] - \sum_{i:y_i=-1} \frac{\alpha_i}{\|\mathbf{w}_-\|} [-\|\mathbf{w}_-\| \mathbf{z} \mathbf{x}_i^\intercal ;\; \mathbf{z}\mathbf{w}_-^\intercal \mathbf{x}_i] \\
&= \frac{1}{\|\mathbf{w}_+\|} [\|\mathbf{w}_+\| \mathbf{u}\mathbf{w}_+^\intercal ;\; \mathbf{u}\mathbf{w}_+^\intercal \mathbf{w}_+] - \frac{1}{\|\mathbf{w}_-\|} [-\|\mathbf{w}_-\| \mathbf{z}\mathbf{w}_-^\intercal ;\; \mathbf{z}\mathbf{w}_-^\intercal \mathbf{w}_-] \\
&= [\mathbf{u}\mathbf{w}_+^\intercal + \mathbf{z}\mathbf{w}_-^\intercal ;\; \mathbf{u}\|\mathbf{w}_+\| - \mathbf{z}\|\mathbf{w}_-\|] \\
&\triangleq \tilde{\boldsymbol{\theta}}.
\end{aligned}$$

This proves the second condition, and shows that $\tilde{\boldsymbol{\theta}}$ is a KKT point of eq. (B.47). □

APPENDIX C

Proofs and derivations for Chapter 4

C.1 The infomax inequality

Theorem 4.1. *For any two random variables* z *and* x *distributed according to the joint distribution with a density* $p(\mathrm{z}, \mathrm{x})$ *and for any conditional density function* $r(\mathbf{z}|\mathbf{x})$ *we have*

$$I_p(\mathbf{z}, \mathbf{x}) \geq H_p(\mathbf{z}) + \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r(\mathbf{z}|\mathbf{x})]. \tag{4.8}$$

(The inequality was originally proved by Barber and Agakov [2003]. Here we provide an alternative direct proof.)

Proof. Using the positivity of KL divergence, we obtain

$$\begin{aligned}
& D_{\mathrm{KL}}(p(\mathbf{z}, \mathbf{x}) \,\|\, r(\mathbf{z}|\mathbf{x})\, p(\mathbf{x})) \geq 0 && \text{(C.1)}\\
\Leftrightarrow \quad & \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log p(\mathbf{z}|\mathbf{x}) + \log p(\mathbf{x}) - \log r(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x})] \geq 0 && \text{(C.2)}\\
\Leftrightarrow \quad & \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log p(\mathbf{z}|\mathbf{x})] \geq \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r(\mathbf{z}|\mathbf{x})]. && \text{(C.3)}
\end{aligned}$$

Plugging the above into the definition of mutual information proves the claim:

$$\begin{aligned}
I_p(\mathbf{z}, \mathbf{x}) &= H_p(\mathbf{z}) - H_p(\mathbf{z}|\mathbf{x}) && \text{(C.4)}\\
&= H_p(\mathbf{z}) + \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log p(\mathbf{z}|\mathbf{x})] && \text{(C.5)}\\
&\geq H_p(\mathbf{z}) + \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r(\mathbf{z}|\mathbf{x})]. && \text{(C.6)}
\end{aligned}$$

□

C.2 Derivation and properties of (4.12)

We will prove that

$$\nabla_p \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r_p^*(\mathbf{z}|\mathbf{x})] = \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}\left[(\nabla_p \log p(\mathbf{z}, \mathbf{x})) \cdot \log r_p^*(\mathbf{z}|\mathbf{x}) + \nabla_p \log r_p^*(\mathbf{z}|\mathbf{x})\right]. \tag{4.12}$$

Note that we treat $p(\mathbf{z})$ as constant and ∇_p denotes the gradient with respect to $p(\mathbf{x}|\mathbf{z})$ only.

$$\nabla_p \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r_p^*(\mathbf{z}|\mathbf{x})] = \nabla_p \mathop{\mathbb{E}}_{\mathbf{z}\sim p}\left[\sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}|\mathbf{z}) \log r_p^*(\mathbf{z}|\mathbf{x})\right] \tag{C.7}$$

$$= \mathop{\mathbb{E}}_{\mathbf{z}\sim p}\left[\sum_{x\in\mathcal{X}} (\nabla_p p(\mathbf{x}|\mathbf{z})) \log r_p^*(\mathbf{z}|\mathbf{x}) + p(\mathbf{x}|\mathbf{z})\Big(\nabla_p \log r_p^*(\mathbf{z}|\mathbf{x})\Big)\right] \tag{C.8}$$

Isolating $p(\mathbf{x}|\mathbf{z})$ and writing the resulting ratio as the derivative of the log yields

$$= \mathop{\mathbb{E}}_{\mathbf{z}\sim p}\left[\sum_{x\in\mathcal{X}} p(\mathbf{x}|\mathbf{z})\Big\{(\nabla_p \log p(\mathbf{x}|\mathbf{z})) \cdot \log r_p^*(\mathbf{z}|\mathbf{x}) + \nabla_p \log r_p^*(\mathbf{z}|\mathbf{x})\Big\}\right] \tag{C.9}$$

$$= \mathop{\mathbb{E}}_{\mathbf{z}\sim p}\left[\sum_{x\in\mathcal{X}} p(\mathbf{x}|\mathbf{z})\Big\{(\nabla_p \log p(\mathbf{z},\mathbf{x})) \cdot \log r_p^*(\mathbf{z}|\mathbf{x}) + \nabla_p \log r_p^*(\mathbf{z}|\mathbf{x})\Big\}\right] \tag{C.10}$$

$$= \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}\Big[(\nabla_p \log p(\mathbf{z},\mathbf{x})) \cdot \log r_p^*(\mathbf{z}|\mathbf{x}) + \nabla_p \log r_p^*(\mathbf{z}|\mathbf{x})\Big]. \tag{C.11}$$

Next, we show that the term $\mathbb{E}_{(\mathbf{z},\mathbf{x}\sim p)}\left[\nabla_p \log r_p^*(\mathbf{z}|\mathbf{x})\right]$ vanishes when $r_p^*(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$.

$$\mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\nabla_p \log p(\mathbf{z}|\mathbf{x})] = \mathop{\mathbb{E}}_{\mathbf{x}\sim p} \mathop{\mathbb{E}}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x})}\left[\frac{\nabla_p p(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}\right] \tag{C.12}$$

$$= \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\int_{\mathcal{Z}} \frac{\nabla_p p(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \cdot p(\mathbf{z}|\mathbf{x})\mathrm{d}\mathbf{z}\right] = \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\nabla_p \int_{\mathcal{Z}} p(\mathbf{z}|\mathbf{x})\mathrm{d}\mathbf{z}\right] = 0. \tag{C.13}$$

C.3 Gradient of the MAE objective

Recall the MAE objective,

$$\begin{aligned} L_m(p,q) &\triangleq L(p,q) - C\left|\hat{I}_p(\mathbf{z},\mathbf{x}) - M\right| \\ &= L(p,q) - C\left|H_p(\mathbf{z}) + \max_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})] - M\right|, \end{aligned} \tag{4.11}$$

and recall that for tractability reasons we approximate the mutual information by

$$\hat{I}(\boldsymbol{\theta}) \triangleq H_p(\mathbf{z}) + \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p_{\boldsymbol{\theta}}}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})], \tag{C.14}$$

where $r_{\boldsymbol{\omega}}$ is independently trained to approach $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

We will now derive the gradient of $C\left|\hat{I}(\boldsymbol{\theta}) - M\right|$ with respect to the model parameters $\boldsymbol{\theta}$, treating $\boldsymbol{\omega}$ as fixed:

$$\nabla_{\boldsymbol{\theta}}\, C\left|\hat{I}(\boldsymbol{\theta}) - M\right| = C \cdot \operatorname{sign}\Big(\hat{I}(\boldsymbol{\theta}) - M\Big) \cdot \nabla_{\boldsymbol{\theta}} \hat{I}(\boldsymbol{\theta}) \tag{C.15}$$

$$= C \cdot \operatorname{sign}\Big(\hat{I}(\boldsymbol{\theta}) - M\Big) \cdot \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\mathbf{z}\sim p}\left[\mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}[\log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})]\right] \tag{C.16}$$

$$= C \cdot \operatorname{sign}\Big(\hat{I}(\boldsymbol{\theta}) - M\Big) \cdot \mathop{\mathbb{E}}_{\mathbf{z}\sim p}\left[\sum_{x\in\mathcal{X}} (\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})) \cdot \log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})\right]. \tag{C.17}$$

Making use of the derivative of the log, we get

$$= C \cdot \operatorname{sign}\Big(\hat{I}(\boldsymbol{\theta}) - M\Big) \cdot \mathop{\mathbb{E}}_{\mathbf{z}\sim p}\left[\sum_{x\in\mathcal{X}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})(\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})) \cdot \log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})\right] \tag{C.18}$$

$$= C \cdot \operatorname{sign}\Big(\hat{I}(\boldsymbol{\theta}) - M\Big) \cdot \mathop{\mathbb{E}}_{(\mathbf{z},\mathbf{x})\sim p_{\boldsymbol{\theta}}}[(\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})) \cdot \log r_{\boldsymbol{\omega}}(\mathbf{z}|\mathbf{x})]. \tag{C.19}$$

The gradient in (C.19) can be estimated using Monte Carlo sampling of both the expectation and the $\hat{I}(\boldsymbol{\theta})$ term inside the sign. The sign estimate will inevitably be wrong sometimes, making our gradient estimator biased. However, note that only the norm is affected, while the direction of the gradient estimate is still correct in expectation.

APPENDIX D

Proofs for Chapter 5

We first define some notation in addition to that of Section 5.2 in the main text. We denote

$$\ell^1(\mathbf{w}) \triangleq \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^\intercal \mathbf{x}_i), \tag{D.1}$$

$$\ell_i(u) \triangleq -y_i \log \sigma(u) - (1-y_i)\log(1-\sigma(u)) - \ell_i^*, \tag{D.2}$$

where ℓ_i^* are constants chosen such that the minimum of ℓ_i is 0, namely $\ell_i^* = -y_i \log y_i - (1-y_i)\log(1-y_i)$.

Slightly abusing notation, we write $\ell(\tau) \triangleq \ell^1(\mathbf{w}(\tau)) \triangleq \ell(\mathbf{W}_1(\tau), \dots, \mathbf{W}_N(\tau))$ for the objective value at time τ.

Finally, for a full-rank matrix $\mathbf{A} \in \mathbb{R}^{d\times m}$ $(m \geq 1)$, we denote by $\mathbf{P}_\mathbf{A} \in \mathbb{R}^{d\times d}$ the matrix of projection onto the span of $\mathbf{A}$,

$$\mathbf{P}_\mathbf{A} = \begin{cases} \mathbf{I}, & m \geq d, \\ \mathbf{A}(\mathbf{A}^\intercal \mathbf{A})^{-1}\mathbf{A}^\intercal, & m < d. \end{cases} \tag{D.3}$$

D.1 Properties of the cross-entropy loss

Theorem D.1 (Gradient)**.** *The gradient of the cross-entropy loss (D.1) takes the form*

$$\nabla \ell^1(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} (\sigma(\mathbf{w}^\intercal \mathbf{x}_i) - y_i)\cdot \mathbf{x}_i. \tag{D.4}$$

It always lies in the data span, $\nabla \ell^1(\mathbf{w}) \in \operatorname{span}(\mathbf{X})$.

Proof. Straightforward calculation. □

Theorem D.2 (Global minima)**.** *The global minimum of the cross-entropy loss (D.1) is 0 and the set of global minimisers is*

$$\left\{\mathbf{w} \in \mathbb{R}^d \mid \mathbf{X}^\intercal \mathbf{w} = \mathbf{X}^\intercal \mathbf{w}_*\right\}. \tag{D.5}$$

Proof. We know that $\ell^1 \geq 0$ and $\ell^1(\mathbf{w}_*) = 0$, so 0 is the optimal objective value, and the set of global optima consists of all $\mathbf{w}$ such that $\ell^1(\mathbf{w}) = 0$. The last condition is equivalent to $\forall_i : \ell_i(\mathbf{w}) = 0$, which in turn is equivalent to $\forall_i : \sigma(\mathbf{w}^\intercal \mathbf{x}_i) = \sigma(\mathbf{w}_*^\intercal \mathbf{x}_i)$. By monotonicity of σ, this is further equivalent to $\forall_i : \mathbf{w}^\intercal \mathbf{x}_i = \mathbf{w}_*^\intercal \mathbf{x}_i$, which is a restatement of (D.5). □

Theorem D.3 (Restricted strong convexity)**.** *Assume $\mathbf{X}$ is full-rank. For any sublevel set $\mathcal{W} = \{\mathbf{w} \,|\, \ell^1(\mathbf{w}) \leq l\}$, there exists $\mu > 0$ such that*

$$\ell^1(\mathbf{v}) \geq \ell^1(\mathbf{w}) + \nabla \ell^1(\mathbf{w})^\intercal (\mathbf{v} - \mathbf{w}) + \frac{\mu}{2} \|\mathbf{v} - \mathbf{w}\|^2 \tag{D.6}$$

for all $\mathbf{w}, \mathbf{v} \in \mathcal{W}$ satisfying $\mathbf{v} - \mathbf{w} \in \operatorname{span}(\mathbf{X})$.

Proof. Consider the 2nd-order Taylor expansion of ℓ^1 around $\mathbf{w}$,

$$\ell^1(\mathbf{v}) = \ell^1(\mathbf{w}) + \nabla \ell^1(\mathbf{w})^\intercal (\mathbf{v} - \mathbf{w}) + \frac{1}{2} (\mathbf{v} - \mathbf{w})^\intercal [\nabla^2 \ell^1(\bar{\mathbf{w}})] (\mathbf{v} - \mathbf{w}), \tag{D.7}$$

where $\nabla^2 \ell^1(\bar{\mathbf{w}})$ is the Hessian of ℓ^1 evaluated at $\bar{\mathbf{w}}$, a point lying between $\mathbf{v}$ and $\mathbf{w}$. A straightforward calculation shows that the Hessian takes the form

$$\nabla^2 \ell^1(\bar{\mathbf{w}}) = \mathbf{X} \mathbf{D}_{\bar{\mathbf{w}}} \mathbf{X}^\intercal, \tag{D.8}$$

where

$$\mathbf{D}_{\bar{\mathbf{w}}} = \operatorname{diag}[\sigma(\bar{\mathbf{w}}^\intercal \mathbf{x}_1)(1 - \sigma(\bar{\mathbf{w}}^\intercal \mathbf{x}_1)), \ldots, \sigma(\bar{\mathbf{w}}^\intercal \mathbf{x}_n)(1 - \sigma(\bar{\mathbf{w}}^\intercal \mathbf{x}_n))]. \tag{D.9}$$

We will now show that there is a constant $\omega > 0$ such that

$$\sigma(\bar{\mathbf{w}}^\intercal \mathbf{x}_i)(1 - \sigma(\bar{\mathbf{w}}^\intercal \mathbf{x}_i)) \geq \omega \tag{D.10}$$

for all $\bar{\mathbf{w}} \in \mathcal{W}$ and $i \in \{1, \ldots, n\}$, so that we can claim $\mathbf{D}_{\bar{\mathbf{w}}} \succeq \omega \mathbf{I}$ and consequently $\nabla^2 \ell^1(\bar{\mathbf{w}}) \succeq \omega \mathbf{X}\mathbf{X}^\intercal$.

Let $\mathbf{w} \in \mathcal{W}$. The bound on $\ell^1(\mathbf{w})$ implies a bound on $\ell_i(\mathbf{w}^\intercal \mathbf{x}_i)$ for all i,

$$\ell_i(\mathbf{w}^\intercal \mathbf{x}_i) \leq n \ell^1(\mathbf{w}) \leq nl. \tag{D.11}$$

Because ℓ_i is convex and $\ell_i(u) \to \infty$ as $u \to \pm\infty$, we know that $\ell_i^{-1}((-\infty, nl])$ is a bounded interval, and the finite union $\cup_{i=1}^n \ell_i^{-1}((-\infty, nl])$ is also a bounded interval, whose size depends only on nl and the data. Hence, there exists $K > 0$ such that $\mathbf{w}^\intercal \mathbf{x}_i \in [-K, K]$ for all $\mathbf{w} \in \mathcal{W}$ and $i \in \{1, \ldots, n\}$. The existence of $\omega > 0$ satisfying (D.10) follows.

Now, let us apply $\nabla^2 \ell^1(\mathbf{w}) \succeq \omega \mathbf{X}\mathbf{X}^\intercal$ to lower-bound (D.7):

$$\ell^1(\mathbf{v}) \geq \ell^1(\mathbf{w}) + \nabla \ell^1(\mathbf{w})^\intercal (\mathbf{v} - \mathbf{w}) + \frac{\omega}{2} (\mathbf{v} - \mathbf{w})^\intercal \mathbf{X}\mathbf{X}^\intercal (\mathbf{v} - \mathbf{w}). \tag{D.12}$$

Consider two cases. If $n \geq d$, $\mathbf{X}\mathbf{X}^\intercal$ is full-rank and $\mathbf{X}\mathbf{X}^\intercal \succeq \lambda_{\min} \mathbf{I}$ holds, where $\lambda_{\min} > 0$ is the smallest eigenvalue of $\mathbf{X}\mathbf{X}^\intercal$. Combined with (D.12), this proves the claim for $n \geq d$ and $\mu = \omega \lambda_{\min}$.

If $n < d$, $\mathbf{X}^\intercal \mathbf{X}$ is full rank. We can use the assumption $\mathbf{v} - \mathbf{w} \in \operatorname{span}(\mathbf{X})$ to deduce

$$\begin{aligned} \|\mathbf{v} - \mathbf{w}\|^2 &= \|\mathbf{P}_{\mathbf{X}} (\mathbf{v} - \mathbf{w})\|^2 \\ &= (\mathbf{v} - \mathbf{w})^\intercal \mathbf{X} (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal (\mathbf{v} - \mathbf{w}) \\ &\leq \lambda_{\max} (\mathbf{v} - \mathbf{w})^\intercal \mathbf{X}\mathbf{X}^\intercal (\mathbf{v} - \mathbf{w}), \end{aligned} \tag{D.13}$$

where $\lambda_{\max} > 0$ is the largest eigenvalue of $(\mathbf{X}^\intercal \mathbf{X})^{-1}$. Combined with (D.12), this proves the claim for $n < d$ and $\mu = \omega / \lambda_{\max}$. □

Corollary D.1 (Restricted Polyak-Lojasiewicz)**.** *Assume* $\mathbf{X}$ *is full-rank. For any sublevel set* $\mathcal{W} = \{\mathbf{w} \,|\, \ell^1(\mathbf{w}) \leq l\}$*, there exists* $c > 0$ *such that*

$$c\ell^1(\mathbf{w}) \leq \frac{1}{2}\left\|\nabla \ell^1(\mathbf{w})\right\|^2 \tag{D.14}$$

for all $\mathbf{w} \in \mathcal{W}$*.*

Proof. Let $\mathbf{w} \in \mathcal{W}$. (If $\mathcal{W}$ is empty, the claim is trivially true.) Theorem D.3 applied to $\mathcal{W}$ implies that for some $\mu > 0$,

$$\ell^1(\mathbf{v}) \geq \ell^1(\mathbf{w}) + \nabla \ell^1(\mathbf{w})^\intercal(\mathbf{v} - \mathbf{w}) + \frac{\mu}{2}\|\mathbf{v} - \mathbf{w}\|^2 \tag{D.15}$$

for all $\mathbf{v} \in \mathcal{W} \cap \mathcal{V}$ where $\mathcal{V} = \{\mathbf{v} \,|\, \mathbf{v} - \mathbf{w} \in \operatorname{span}(\mathbf{X})\}$. Taking $\min_{\mathbf{v} \in \mathcal{W} \cap \mathcal{V}}$ on both sides, then relaxing part of the constraint on the right-hand side yields

$$\begin{aligned} \min_{\mathbf{v} \in \mathcal{W} \cap \mathcal{V}} \ell^1(\mathbf{v}) \;&\geq \min_{\mathbf{v} \in \mathcal{W} \cap \mathcal{V}} \ell^1(\mathbf{w}) + \nabla \ell^1(\mathbf{w})^\intercal(\mathbf{v} - \mathbf{w}) + \frac{\mu}{2}\|\mathbf{v} - \mathbf{w}\|^2 \\ &\geq \min_{\mathbf{v} \in \mathcal{V}} \ell^1(\mathbf{w}) + \nabla \ell^1(\mathbf{w})^\intercal(\mathbf{v} - \mathbf{w}) + \frac{\mu}{2}\|\mathbf{v} - \mathbf{w}\|^2. \end{aligned} \tag{D.16}$$

Now, the minimum on the left-hand side is equal to 0 and is attained at $\mathbf{v} = \mathbf{w} + \mathbf{P}_{\mathbf{X}}(\mathbf{w}_* - \mathbf{w})$, as can be seen from Theorem D.2. For the right-hand side, we can substitute $\mathbf{v} = \mathbf{w} + \mathbf{X}\mathbf{a}$ for $\mathbf{a} \in \mathbb{R}^n$ and find the unconstrained minimum with respect to $\mathbf{a}$. We get

$$\begin{aligned} 0 &\geq \ell^1(\mathbf{w}) - \frac{1}{2\mu}\nabla \ell^1(\mathbf{w})^\intercal \mathbf{X}(\mathbf{X}^\intercal \mathbf{X})^{-1}\mathbf{X}^\intercal \nabla \ell^1(\mathbf{w}) \\ &\geq \ell^1(\mathbf{w}) - \frac{\lambda_{\max}}{2\mu}\left\|\nabla \ell^1(\mathbf{w})\right\|^2, \end{aligned} \tag{D.17}$$

where $\lambda_{\max} > 0$ is the largest eigenvalue of $\mathbf{X}(\mathbf{X}^\intercal \mathbf{X})^{-1}\mathbf{X}^\intercal$. This yields the result with $c = \mu/\lambda_{\max}$. □

D.2 Proof of Theorem 5.1

We will prove a supporting lemma, and then the theorem.

Lemma D.1. *Assume the student is a directly parameterised linear classifier* $(L = 1)$ *initialised at zero,* $\mathbf{w}(0) = \mathbf{0}$*. Then,* $\mathbf{w}(\tau) \in \operatorname{span}(\mathbf{X})$ *for* $\tau \in [0, \infty)$*.*

Proof. Let $\mathbf{q} \in \mathbb{R}^d$ be any vector orthogonal to the span of $\mathbf{X}$. It suffices to show that $\mathbf{q}^\intercal \mathbf{w}(\tau) = 0$. For that, notice that $\mathbf{q}^\intercal \mathbf{w}(0) = 0$ and

$$\frac{\mathrm{d}}{\mathrm{d}\tau}(\mathbf{q}^\intercal \mathbf{w}(\tau)) = -\mathbf{q}^\intercal \nabla \ell^1(\mathbf{w}(\tau)) = 0, \tag{D.18}$$

where the last equality follows from the fact that $\nabla \ell^1(\mathbf{w}(\tau)) \in \operatorname{span}(\mathbf{X})$ (Theorem D.1). The claim follows. □

Theorem 5.1. *Assume the student is a directly parameterised linear classifier* $(L = 1)$ *initialised such that* $\mathbf{w}(0) = \mathbf{0}$*. Then, the student's weight vector fulfills almost surely*

$$\mathbf{w}(t) \to \hat{\mathbf{w}}, \tag{5.5}$$

for $t \to \infty$*, where*

$$\hat{\mathbf{w}} = \begin{cases} \mathbf{w}_*, & n \geq d, \\ \mathbf{X}(\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{w}_*, & n < d. \end{cases} \tag{5.6}$$

Proof. Recall the time-derivative of ℓ,

$$\ell'(\tau) = -\left\| \nabla \ell^1(\mathbf{w}(\tau)) \right\|^2. \tag{D.19}$$

The data matrix $\mathbf{X}$ is almost surely (wrt. $\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}$) full-rank, we can therefore apply Corollary D.1 to $\mathcal{W} = \{\mathbf{w} \,|\, \ell^1(\mathbf{w}) \leq \ell^1(\mathbf{0})\}$ and $\mathbf{w}(\tau)$ to lower-bound the gradient norm on the right-hand side of (D.19). We obtain $\ell'(\tau) \leq -c\ell(\tau)$ for some $c > 0$ and all $\tau \in [0, \infty)$, or equivalently,

$$(\log \ell(\tau))' \leq -c. \tag{D.20}$$

Integrating over $[0, t]$ yields $\ell(t) \leq \ell(0) \cdot e^{-ct}$, which proves global convergence in the objective: $\ell(t) \to 0$ as $t \to \infty$.

Now invoke Theorem D.3 with $\mathcal{W}$ as above, $\mathbf{v} = \mathbf{w}(t)$ and $\mathbf{w} = \hat{\mathbf{w}}$ (we know that both $\mathbf{w}(\tau), \hat{\mathbf{w}} \in \mathcal{W} \cap \operatorname{span}(\mathbf{X})$, partly by Lemma D.1):

$$\ell(t) \geq \frac{\mu}{2} \|\mathbf{w}(t) - \hat{\mathbf{w}}\|^2. \tag{D.21}$$

Since $\ell(t) \to 0$ as $t \to \infty$, the theorem follows. □

D.3 Proof of Theorem 5.2

Theorem 5.2. *Let* $\hat{\mathbf{w}}$ *be defined as in eq.* (5.6)*. Assume the student is a deep linear network, initialised such that for some* $\epsilon > 0$*,*

$$\|\mathbf{w}(0)\| < \min\left\{ \|\hat{\mathbf{w}}\|, \epsilon^L \left(\epsilon^2 \|\hat{\mathbf{w}}\|^{-\frac{2}{L}} + \|\hat{\mathbf{w}}\|^{2-\frac{2}{L}} \right)^{-\frac{L}{2}} \right\}, \tag{5.11}$$

$$\ell^1(\mathbf{w}(0)) < \ell^1(\mathbf{0}), \tag{5.12}$$

$$\mathbf{W}_{l+1}(0)^\intercal \mathbf{W}_{l+1}(0) = \mathbf{W}_l(0) \mathbf{W}_l(0)^\intercal \tag{5.13}$$

for $l \in [L-1]$*. Then, for* $n \geq d$*, the student's weight vector fulfills almost surely*

$$\mathbf{w}(t) \to \hat{\mathbf{w}}, \tag{5.14}$$

and for $n < d$*,*

$$\|\mathbf{w}(t) - \hat{\mathbf{w}}\| \leq \epsilon, \tag{5.15}$$

for all t *large enough.*

For the proof, we will need a result by Arora et al. [2018], which characterises the induced flow on $\mathbf{w}(\tau)$ when running gradient descent on the component matrices $\{\mathbf{W}_l\}$.

Lemma D.2 ([Arora et al., 2018, Claim 2])**.** *If the balancedness condition (5.13) holds, then*

$$\frac{\partial \mathbf{w}(\tau)}{\partial \tau} = -\|\mathbf{w}(\tau)\|^{\frac{2(L-1)}{L}} \left(\nabla \ell^1(\mathbf{w}(\tau)) + (L-1)\cdot \mathbf{P}_{\mathbf{w}(\tau)} \nabla \ell^1(\mathbf{w}(\tau))\right). \tag{D.22}$$

Proof of Theorem 5.2. Similarly to the case $L = 1$, we start by looking at the time-derivative of ℓ,

$$\begin{aligned}
\ell'(\tau) &= \nabla \ell^1(\mathbf{w}(\tau))^\intercal \left(\frac{\partial \mathbf{w}(\tau)}{\partial \tau}\right) \\
&= -\|\mathbf{w}(\tau)\|^{\frac{2(L-1)}{L}} \left(\left\|\nabla \ell^1(\mathbf{w}(\tau))\right\|^2 + (L-1)\cdot \left\|\mathbf{P}_{\mathbf{w}(\tau)} \nabla \ell^1(\mathbf{w}(\tau))\right\|^2\right) \\
&\leq -\|\mathbf{w}(\tau)\|^{\frac{2(L-1)}{L}} \cdot \left\|\nabla \ell^1(\mathbf{w}(\tau))\right\|^2.
\end{aligned} \tag{D.23}$$

It is non-positive, so $\mathbf{w}(\tau)$ stays within the $\ell(0)$-sublevel set throughout optimisation,

$$\mathbf{w}(\tau) \in \mathcal{W} = \left\{\mathbf{w} \,|\, \ell^1(\mathbf{w}) \leq \ell(0)\right\}. \tag{D.24}$$

Also, $\mathcal{W}$ is convex and by Assumption (5.12) it does not contain $\mathbf{0}$. We can therefore take $\delta > 0$ to be the distance between $\mathcal{W}$ and $\mathbf{0}$, and it follows that $\|\mathbf{w}(\tau)\| \geq \delta$ for $\tau \in [0, \infty)$.

Now, noting that $\mathbf{X}$ is almost surely full-rank, apply Corollary D.1 to $\mathcal{W}$ and $\mathbf{w}(\tau)$ to upper-bound the right-hand side of (D.23),

$$\ell'(\tau) \leq -c\delta^{\frac{2(L-1)}{L}} \ell(\tau). \tag{D.25}$$

Letting $\tilde{c} = c\delta^{\frac{2(L-1)}{L}}$, we get $(\log \ell(\tau))' \leq -\tilde{c}$ and consequently $\ell(t) \leq \ell(0)\cdot e^{-\tilde{c}t}$. This proves convergence in the objective, $\ell(t) \to 0$ as $t \to \infty$.

To prove convergence in parameters, we decompose the 'error' $\mathbf{w}(\tau) - \hat{\mathbf{w}}$ into orthogonal components and bound each of them separately,

$$\|\mathbf{w}(\tau) - \hat{\mathbf{w}}\|^2 = \|\mathbf{P}_{\mathbf{X}}(\mathbf{w}(\tau) - \hat{\mathbf{w}})\|^2 + \|\mathbf{P}_{\mathbf{Q}}(\mathbf{w}(\tau) - \hat{\mathbf{w}})\|^2, \tag{D.26}$$

where the columns of $\mathbf{Q} \in \mathbb{R}^{d\times(d-n)}$ orthogonally complement those of $\mathbf{X}$. If $n \geq d$, we simply bound the first term and disregard the second one.

To bound the first term, invoke Theorem D.3 with $\mathcal{W}$, $\mathbf{v} = \mathbf{P}_{\mathbf{X}}\mathbf{w}(\tau)$ and $\mathbf{w} = \mathbf{P}_{\mathbf{X}}\hat{\mathbf{w}}$. One can check that $\ell^1(\mathbf{P}_{\mathbf{X}}\mathbf{u}) = \ell^1(\mathbf{u})$ for all $\mathbf{u} \in \mathbb{R}^d$, so $\mathbf{P}_{\mathbf{X}}\mathbf{w}(\tau) \in \mathcal{W}$ and our use of the theorem is valid. We obtain

$$\ell(\tau) \geq \frac{\mu}{2}\|\mathbf{P}_{\mathbf{X}}(\mathbf{w}(\tau) - \hat{\mathbf{w}})\|^2. \tag{D.27}$$

Since $\ell(\tau) \to 0$, it follows that

$$\|\mathbf{P}_{\mathbf{X}}(\mathbf{w}(\tau) - \hat{\mathbf{w}})\|^2 \to 0 \tag{D.28}$$

as $\tau \to \infty$.

For the second term, notice that $\hat{\mathbf{w}} \in \operatorname{span}(\mathbf{X})$, so $\mathbf{P}_{\mathbf{Q}}\hat{\mathbf{w}}$ vanishes and we are left with $\|\mathbf{P}_{\mathbf{Q}}\mathbf{w}(\tau)\|^2$. Denote this quantity $q(\tau)$. Its time derivative is

$$\begin{aligned}
q'(\tau) &= 2(\mathbf{P}_{\mathbf{Q}}\mathbf{w}(\tau))^\intercal \left(\frac{\partial \mathbf{w}(\tau)}{\partial \tau}\right) \\
&= -2\|\mathbf{w}(\tau)\|^{\frac{2(L-1)}{L}} \left(\mathbf{w}(\tau)^\intercal \mathbf{P}_{\mathbf{Q}} \nabla \ell^1(\mathbf{w}(\tau)) + \frac{(L-1)}{\|\mathbf{w}(\tau)\|^2}\cdot \mathbf{w}(\tau)^\intercal \mathbf{P}_{\mathbf{Q}}\mathbf{w}(\tau)\cdot \mathbf{w}(\tau)^\intercal \nabla \ell^1(\mathbf{w}(\tau))\right) \\
&= -2q(\tau)(L-1)\|\mathbf{w}(\tau)\|^{-2/L} \mathbf{w}(\tau)^\intercal \nabla \ell^1(\mathbf{w}(\tau)),
\end{aligned} \tag{D.29}$$

where we have used the fact that $\nabla \ell^1(\mathbf{w}(\tau)) \in \operatorname{span}(\mathbf{X})$ (Theorem D.1) and $\mathbf{Q}$ is orthogonal to $\mathbf{X}$. Rearranging, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\left(\frac{\log q(\tau)}{2(L-1)}\right) = -\|\mathbf{w}(\tau)\|^{-2/L} \cdot \mathbf{w}(\tau)^\intercal \nabla \ell^1(\mathbf{w}(\tau)). \tag{D.30}$$

It turns out that the right-hand side expression is integrable in yet another way, namely

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\left(\frac{1}{2L}\log\|\mathbf{w}(\tau)\|^2\right) = -\|\mathbf{w}(\tau)\|^{-2/L} \cdot \mathbf{w}(\tau)^\intercal \nabla \ell^1(\mathbf{w}(\tau)). \tag{D.31}$$

Equating the two and integrating over $[0, t]$ yields

$$\log\frac{q(t)}{q(0)} = \frac{L-1}{L} \cdot \log\frac{\|\mathbf{w}(t)\|^2}{\|\mathbf{w}(0)\|^2}, \tag{D.32}$$

which implies

$$\frac{q(t)}{\|\mathbf{w}(t)\|^2} \le \left(\frac{\|\mathbf{w}(0)\|}{\|\mathbf{w}(t)\|}\right)^{2/L}, \tag{D.33}$$

because $q(0) \le \|\mathbf{w}(0)\|^2$.

We now bound the norm of $\mathbf{w}(t)$. Starting from an orthogonal decomposition similar to (D.26) and applying (D.28) with (D.33), we get

$$\begin{aligned} \|\mathbf{w}(t)\|^2 =& \|\mathbf{P}_{\mathbf{X}}\mathbf{w}(t)\|^2 + \|\mathbf{P}_{\mathbf{Q}}\mathbf{w}(t)\|^2 \\ \limsup_{t\to\infty}\|\mathbf{w}(t)\|^2 \le& \|\hat{\mathbf{w}}\|^2 + \|\mathbf{w}(0)\|^{\frac{2}{L}} \limsup_{t\to\infty}\|\mathbf{w}(t)\|^{2-\frac{2}{L}}. \end{aligned} \tag{D.34}$$

Denote $\nu := \limsup_{t\to\infty}\|\mathbf{w}(t)\|$. By the same orthogonal decomposition, we also know that $\nu^2 \ge \limsup_{t\to\infty}\|\mathbf{P}_{\mathbf{X}}\mathbf{w}(t)\|^2 = \|\hat{\mathbf{w}}\|^2 > 0$, so we can divide both sides above by ν^2,

$$1 \le \frac{\|\hat{\mathbf{w}}\|^2}{\nu^2} + \frac{\|\mathbf{w}(0)\|^{2/L}}{\nu^{2/L}} =: f(\nu). \tag{D.35}$$

On the right-hand side, we now have a decreasing function of ν that goes to zero as $\nu \to \infty$. However, evaluated at our specific ν, it is lower-bounded by 1, implying an implicit upper bound for ν.

How do we find this bound? Suppose we find some constant K such that $f(K) \le 1$. Then, because f is decreasing, it must be the case that $\nu \le K$. One such candidate for K is

$$K = \|\hat{\mathbf{w}}\| \cdot \left(1 - \frac{\|\mathbf{w}(0)\|^{2/L}}{\|\hat{\mathbf{w}}\|^{2/L}}\right)^{\frac{-L}{2(L-1)}}. \tag{D.36}$$

(Here we have used condition (5.11): $\|\mathbf{w}(0)\| < \|\hat{\mathbf{w}}\|$.) To check that indeed $f(K) \le 1$, start from the inequality

$$(\|\hat{\mathbf{w}}\|/K)^{\frac{2(L-1)}{L}} + \frac{\|\mathbf{w}(0)\|^{2/L}}{\|\hat{\mathbf{w}}\|^{2/L}} = 1 \le \left(1 - \frac{\|\mathbf{w}(0)\|^{2/L}}{\|\hat{\mathbf{w}}\|^{2/L}}\right)^{\frac{-1}{L-1}} = (\|\hat{\mathbf{w}}\|/K)^{-\frac{2}{L}}. \tag{D.37}$$

Taking the leftmost and rightmost expression and multiplying by $(\|\hat{\mathbf{w}}\|/K)^{2/L}$ yields

$$f(K) = \frac{\|\hat{\mathbf{w}}\|^2}{K^2} + \frac{\|\mathbf{w}(0)\|^{2/L}}{K^{2/L}} \le 1. \tag{D.38}$$

Hence,

$$\limsup_{t\to\infty} \|\mathbf{w}(t)\| \leq \|\hat{\mathbf{w}}\| \cdot \left(1 - \frac{\|\mathbf{w}(0)\|^{2/L}}{\|\hat{\mathbf{w}}\|^{2/L}}\right)^{\frac{-L}{2(L-1)}}. \tag{D.39}$$

Finally, let us turn back to our original goal of bounding $\|\mathbf{w}(\tau) - \hat{\mathbf{w}}\|^2$. With (D.26), (D.28), (D.33) and (D.39), we now know that

$$\limsup_{t\to\infty} \|\mathbf{w}(\tau) - \hat{\mathbf{w}}\|^2 \leq \|\mathbf{w}(0)\|^{\frac{2}{L}} \|\hat{\mathbf{w}}\|^{\frac{2(L-1)}{L}} \left(1 - \frac{\|\mathbf{w}(0)\|^{\frac{2}{L}}}{\|\hat{\mathbf{w}}\|^{\frac{2}{L}}}\right)^{-1} \tag{D.40}$$

$$= \frac{\|\hat{\mathbf{w}}\|^{2+2/L}}{\|\hat{\mathbf{w}}\|^{2/L} - \|\mathbf{w}(0)\|^{2/L}} - \|\hat{\mathbf{w}}\|^2. \tag{D.41}$$

Hence, if we initialise close enough to zero, as specified by condition (5.11), we can ensure that

$$\limsup_{t\to\infty} \|\mathbf{w}(\tau) - \hat{\mathbf{w}}\|^2 < \epsilon^2. \tag{D.42}$$

This concludes the proof. □

D.4 Theorem 5.3 for approximate distillation

We extend Theorem 5.3 to the setting where the student learns $\hat{\mathbf{w}} = \mathbf{X}(\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{w}_*$ only ϵ-approximately, as is the case for deep linear networks initialised as in Theorem 5.2. When $n \geq d$, the teacher's weight vector is recovered exactly and the transfer risk is zero, even when the student is deep. The following theorem therefore only covers the case $n < d$.

Theorem D.4 (Risk bound for approximate distillation)**.** *Let $n < d$. For any training set $\mathbf{X} \in \mathbb{R}^{d\times n}$, let $\hat{h}_{\mathbf{X}}(\mathbf{x}) = \mathbb{1}\{\hat{\mathbf{w}}_\epsilon^\intercal\mathbf{x} \geq 0\}$ be a linear classifier whose weight vector is ϵ-close to the distillation solution $\hat{\mathbf{w}}$, i.e. $\|\hat{\mathbf{w}}_\epsilon - \hat{\mathbf{w}}\| \leq \epsilon$, where ϵ is a positive constant such that $\epsilon \leq \frac{1}{2}\|\hat{\mathbf{w}}\|$. Define $\delta := \sqrt{\frac{2\pi\epsilon}{\|\hat{\mathbf{w}}\|}}$. Then, it holds for any $\beta \in [0, \pi/2 - \delta]$ that*

$$\mathop{\mathbb{E}}_{\mathbf{X}\sim P_{\mathbf{x}}^{\otimes n}} \left[R\Big(\hat{h}_{\mathbf{X}}\Big|P_{\mathbf{x}}, \mathbf{w}_*\Big)\right] \leq p(\beta) + p(\pi/2 - \delta - \beta)^n. \tag{D.43}$$

The result is very similar to Theorem 5.3 in the main text, the only difference is the constant δ which compensates for the imprecision in learning $\hat{\mathbf{w}}$ by pushing the bound up (recall that p is decreasing). However, as ϵ goes to zero, so does δ and we recover the original bound.

For the proof, we start with a tool for controlling the angle between $\hat{\mathbf{w}}$ and $\hat{\mathbf{w}}_\epsilon$. Recall that the angle is defined as

$$\alpha(\mathbf{w}, \mathbf{v}) = \cos^{-1}\left(\frac{\mathbf{w}^\intercal\mathbf{v}}{\|\mathbf{w}\| \cdot \|\mathbf{v}\|}\right) \tag{D.44}$$

for $\mathbf{w}, \mathbf{v} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$.

Lemma D.3. *Let $\mathbf{w}, \mathbf{v} \in \mathbb{R}^d$ be such that $\|\mathbf{w} - \mathbf{v}\| \leq \epsilon$, where $\epsilon \leq \frac{1}{2}\|\mathbf{w}\|$. Then $\alpha(\mathbf{w}, \mathbf{v}) \leq \sqrt{\frac{2\pi\epsilon}{\|\mathbf{w}\|}}$.*

Proof of Lemma D.3. The first step is to lower-bound the inner product $\mathbf{w}^\intercal \mathbf{v}$. To that end, we expand and rearrange $\|\mathbf{w} - \mathbf{v}\|^2 \leq \epsilon^2$ to obtain

$$2\mathbf{w}^\intercal \mathbf{v} \geq \|\mathbf{w}\|^2 + \|\mathbf{v}\|^2 - \epsilon^2. \tag{D.45}$$

Now use the triangle relation $\|\mathbf{v}\| \geq \|\mathbf{w}\| - \epsilon$ squared to lower-bound the right-hand side of (D.45) and get

$$2\mathbf{w}^\intercal \mathbf{v} \geq 2\|\mathbf{w}\|^2 - 2\epsilon\|\mathbf{w}\|, \tag{D.46}$$

which implies

$$\frac{\mathbf{w}^\intercal \mathbf{v}}{\|\mathbf{w}\| \cdot \|\mathbf{v}\|} \geq \frac{\|\mathbf{w}\| - \epsilon}{\|\mathbf{v}\|} \geq \frac{\|\mathbf{w}\| - \epsilon}{\|\mathbf{w}\| + \epsilon} \geq 1 - \frac{2\epsilon}{\|\mathbf{w}\|}. \tag{D.47}$$

Thus,

$$1 - \frac{2\epsilon}{\|\mathbf{w}\|} \leq \frac{\mathbf{w}^\intercal \mathbf{v}}{\|\mathbf{w}\| \cdot \|\mathbf{v}\|} = \cos(\alpha(\mathbf{w}, \mathbf{v})). \tag{D.48}$$

The left-hand side is by assumption non-negative, so we have $\alpha(\mathbf{w}, \mathbf{v}) \in [-\pi/2, \pi/2]$. On this domain,

$$\cos x \leq 1 - \frac{x^2}{\pi}, \tag{D.49}$$

which lets us deduce

$$1 - \frac{2\epsilon}{\|\mathbf{w}\|} \leq 1 - \frac{\alpha(\mathbf{w}, \mathbf{v})^2}{\pi}. \tag{D.50}$$

Rearranging yields the result. □

Proof of Theorem D.4. We decompose the expected risk as follows:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} \left[R\big(\hat{h}_{\mathbf{X}} \big| P_{\mathbf{x}}, \mathbf{w}_*\big)\right] &= \mathop{\mathbb{P}}_{\substack{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n} \\ \mathbf{x} \sim P_{\mathbf{x}}}} [\mathbf{w}_*^\intercal \mathbf{x} \cdot \hat{\mathbf{w}}_\epsilon^\intercal \mathbf{x} < 0] \\
&= \int_{\mathbf{x}: \bar{\alpha}(\mathbf{w}_*, \mathbf{x}) \geq \beta} \mathop{\mathbb{P}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} [\mathbf{w}_*^\intercal \mathbf{x} \cdot \hat{\mathbf{w}}_\epsilon^\intercal \mathbf{x} < 0 | \mathbf{x}] \, \mathrm{d}P_{\mathbf{x}} \\
&\quad + \int_{\mathbf{x}: \bar{\alpha}(\mathbf{w}_*, \mathbf{x}) < \beta,\, \mathbf{w}_*^\intercal \mathbf{x} > 0} \mathop{\mathbb{P}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} [\hat{\mathbf{w}}_\epsilon^\intercal \mathbf{x} < 0 | \mathbf{x}] \, \mathrm{d}P_{\mathbf{x}} \\
&\quad + \int_{\mathbf{x}: \bar{\alpha}(\mathbf{w}_*, \mathbf{x}) < \beta,\, \mathbf{w}_*^\intercal \mathbf{x} < 0} \mathop{\mathbb{P}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} [\hat{\mathbf{w}}_\epsilon^\intercal \mathbf{x} > 0 | \mathbf{x}] \, \mathrm{d}P_{\mathbf{x}}.
\end{aligned} \tag{D.51}$$

Let us fix some $\mathbf{x}$ for which $\bar{\alpha}(\mathbf{w}_*, \mathbf{x}) < \beta$ and $\mathbf{w}_*^\intercal \mathbf{x} > 0$; for this $\mathbf{x}$ we have $\alpha(\mathbf{w}_*, \mathbf{x}) = \bar{\alpha}(\mathbf{w}_*, \mathbf{x})$. Consider the situation where $\bar{\alpha}(\mathbf{w}_*, \mathbf{x}_i) < \pi/2 - \beta - \delta$ for some i. Then by the triangle inequality, Lemma D.3 and Lemma 5.1,

$$\begin{aligned}
\alpha(\hat{\mathbf{w}}_\epsilon, \mathbf{x}) &\leq \alpha(\hat{\mathbf{w}}_\epsilon, \hat{\mathbf{w}}) + \alpha(\mathbf{w}_*, \hat{\mathbf{w}}) + \alpha(\mathbf{w}_*, \mathbf{x}) && \text{(D.52)} \\
&\leq \delta + \bar{\alpha}(\mathbf{w}_*, \mathbf{x}_i) + \bar{\alpha}(\mathbf{w}_*, \mathbf{x}) && \text{(D.53)} \\
&< \pi/2, && \text{(D.54)}
\end{aligned}$$

which implies $\hat{\mathbf{w}}_\epsilon^\intercal \mathbf{x} > 0$, i.e. a correct prediction (same as the teacher's). Conversely, an error can occur only if $\bar{\alpha}(\mathbf{w}_*, \mathbf{x}_i) \geq \pi/2 - \delta - \beta$ for all i. Because $\mathbf{x}_i$ are independent, we have

$$\begin{aligned}
\mathop{\mathbb{P}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} [\hat{\mathbf{w}}_\epsilon^\intercal \mathbf{x} < 0 | \, \bar{\alpha}(\mathbf{w}_*, \mathbf{x}) < \beta, \, \mathbf{w}_*^\intercal \mathbf{x} > 0] &\leq \mathop{\mathbb{P}}_{\mathbf{X} \sim P_{\mathbf{x}}^{\otimes n}} [\forall_i : \bar{\alpha}(\mathbf{w}_*, \mathbf{x}_i) \geq \pi/2 - \delta - \beta] \\
&= p(\pi/2 - \delta - \beta)^n.
\end{aligned} \tag{D.55}$$

By a symmetric argument, one can show that

$$\mathop{\mathbb{P}}_{\mathbf{X}\sim P_{\mathbf{x}}^{\otimes n}}[\hat{\mathbf{w}}_\epsilon^\intercal \mathbf{x} > 0 |\, \bar{\alpha}(\mathbf{w}_*, \mathbf{x}) < \beta,\, \mathbf{w}_*^\intercal \mathbf{x} < 0] \leq p(\pi/2 - \delta - \beta)^n. \tag{D.56}$$

Combining (D.51), (D.55) and (D.56) yields the result. □

APPENDIX E

Additional experiments for Chapter 6

We provide additional results/observations regarding the choice of teachers (Section E.1), the semi-supervised setting with very few labels (Section E.2), and inference times for different modes and exits (Section E.3).

E.1 Choice of teachers

We performed exploratory experiments with a different choice of teacher sets $\mathcal{T}$. In particular, we let each exit learn from all later exits, $\mathcal{T}(m) = \{m+1, \dots, M\}$ for $m < M$ and $\mathcal{T}(M) = \emptyset$. The intuition behind this choice is that learning from an ensemble of good exits might be better than learning from a single good exit. However, as the results show, this turns out not to be the case.

Figure E.1 shows the accuracy curve of a model trained by distillation from all later exits (yellow), as well as curves for distillation from only the last exit (green) and exit-wise training (blue) for comparison. We observe that the two teacher-set choices yield models of very similar accuracy, so there seems to be little benefit in adopting more complicated teacher-student setups.

E.2 Semi-supervised distillation with few labels

In the main text, we show that using additional unlabelled data with distillation-based training may improve accuracy. However, the gains are relatively modest, and only significant for the late exits in the case of ImageNet(100) and the early exits in the case of CIFAR(150) and CIFAR(250). One could speculate why the gains are not larger, and a potential explanation would be that in the considered settings, the amount of labelled data is too large. To test this hypothesis, we ran the experiment from Section 6.3.1 on CIFAR(80), i.e. using only 30 labelled images per class for training and 50 for validation. However, the gains remain rather small (see Figure E.2), at most 1%, similar to the case of CIFAR(150) and CIFAR(250).

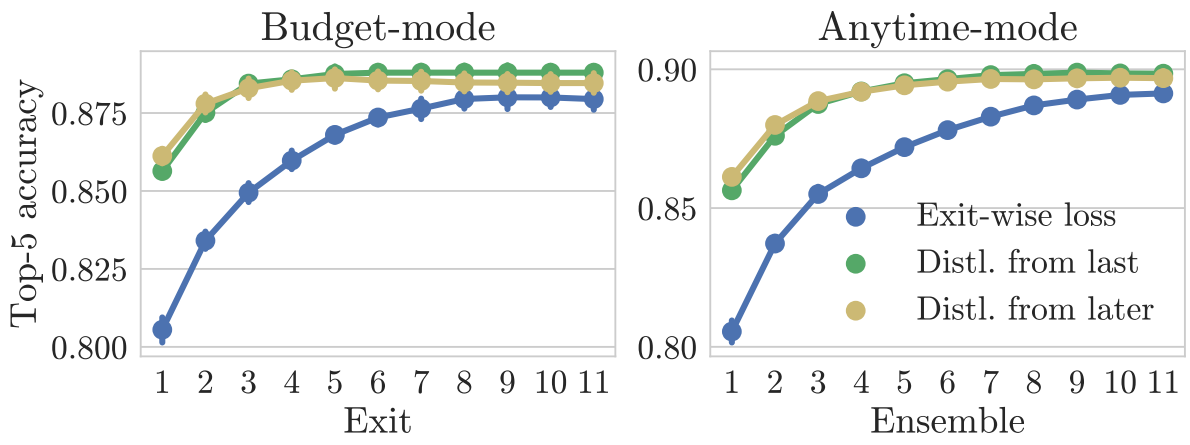


Figure E.1: Top-5 accuracy of a model trained by distillation from all later exits (yellow) vs. trained by distillation from the last exit (green) vs. trained by the exit-wise loss (blue), on CIFAR(250). Results shown for different computational budgets in both the budget-mode (left) and the anytime-mode (right).

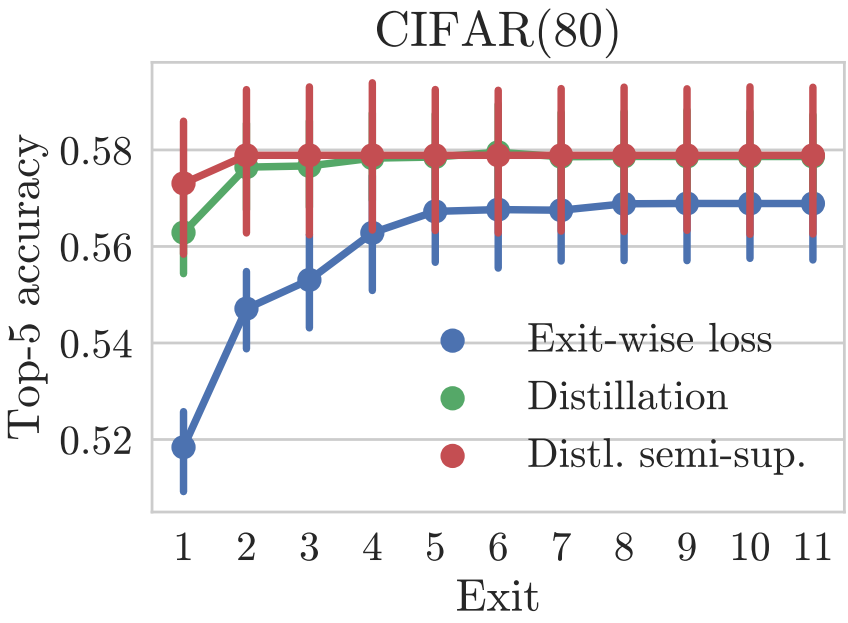


Figure E.2: Top-5 accuracy as a function of computational budget (denominated in available exits). MSDNet trained by the exit-wise loss (blue) vs. trained by distillation (green) vs. trained by semi-supervised distillation (red) on CIFAR(80).

	CPU timings [s]		GPU timings [s]	
	Budget-mode	Anytime-mode	Budget-mode	Anytime-mode
Exit / Ensemble 1	0.024 ± 0.022	0.026 ± 0.029	0.007 ± 0.000	0.007 ± 0.000
Exit / Ensemble 2	0.034 ± 0.029	0.043 ± 0.037	0.011 ± 0.000	0.011 ± 0.000
Exit / Ensemble 3	0.043 ± 0.030	0.062 ± 0.040	0.015 ± 0.000	0.015 ± 0.000
Exit / Ensemble 4	0.051 ± 0.035	0.081 ± 0.049	0.018 ± 0.000	0.019 ± 0.000
Exit / Ensemble 5	0.062 ± 0.041	0.099 ± 0.055	0.022 ± 0.001	0.023 ± 0.001
Exit / Ensemble 6	0.067 ± 0.034	0.121 ± 0.064	0.025 ± 0.001	0.027 ± 0.001
Exit / Ensemble 7	0.071 ± 0.046	0.139 ± 0.066	0.029 ± 0.001	0.032 ± 0.001
Exit / Ensemble 8	0.085 ± 0.035	0.164 ± 0.078	0.032 ± 0.001	0.035 ± 0.001
Exit / Ensemble 9	0.093 ± 0.035	0.169 ± 0.085	0.036 ± 0.001	0.039 ± 0.001
Exit / Ensemble 10	0.103 ± 0.040	0.196 ± 0.078	0.039 ± 0.001	0.043 ± 0.001
Exit / Ensemble 11	0.105 ± 0.038	0.219 ± 0.085	0.041 ± 0.001	0.045 ± 0.001

Table E.1: Inference times for the CIFAR MSDNet operating either in the budget-mode or the anytime-mode, evaluated either on CPU or GPU. We report the mean $\pm$ stdev over 1000 runs.

	CPU timings [s]		GPU timings [s]	
	Budget-mode	Anytime-mode	Budget-mode	Anytime-mode
Exit / Ensemble 1	0.159 ± 0.048	0.229 ± 0.079	0.027 ± 0.001	0.027 ± 0.002
Exit / Ensemble 2	0.339 ± 0.099	0.373 ± 0.094	0.038 ± 0.001	0.038 ± 0.001
Exit / Ensemble 3	0.471 ± 0.131	0.519 ± 0.118	0.048 ± 0.001	0.048 ± 0.002
Exit / Ensemble 4	0.559 ± 0.138	0.647 ± 0.117	0.055 ± 0.001	0.056 ± 0.001
Exit / Ensemble 5	0.665 ± 0.125	0.703 ± 0.145	0.057 ± 0.001	0.059 ± 0.001

Table E.2: Inference times for the ImageNet MSDNet operating either in the budget-mode or the anytime-mode, evaluated either on CPU or GPU. We report the mean $\pm$ stdev over 1000 runs.

E.3 Example inference times

To give the reader a sense of the efficiency gains achievable by anytime inference, we provide some example inference times for CIFAR MSDNet (Table E.1) and ImageNet MSDNet (Table E.2), for the two inference modes (budget-mode and anytime-mode) and different exits. We made no effort to optimise these timings. They are obtained by simply running our code, without any changes, and measuring the time elapsed. We expect the results to generalise qualitatively to different hardware and implementations, though the exact numbers are likely to vary.