



Rahim Entezari, Msc

Optimization and Generalization of Neural Networks at the Edge

Doctoral Thesis

to achieve the university degree of
Doctor of Science

PhD degree program: Engineering Sciences, Telecommunication Engineering

submitted to

Graz University of Technology

Supervisor and First Examiner:
Prof. Olga Saukh

Second and Third Examiners:
Prof. Lothar Thiele and Prof. Ludwig Schmidt

Institute for Technical Informatics

July 2023

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Datum

Unterschrift

Abstract

The growing ubiquity of deep neural networks in daily life highlights the need to address the challenges and limitations they face when deployed on edge devices. These devices impose constraints on resources, accessibility, and scalability, and are subject to dynamic changes in their working environment. This makes the optimization and generalization of neural networks a critical area of research. In this thesis, we adopt an empirical science approach, treating deep learning as an observable and experimental phenomenon. Our primary goal is to understand when neural networks work and when they do not, and we focus on examining the optimization and generalization of neural networks in the context of edge environments.

As we strive to optimize neural network performance under resource constraints, we explore the delicate balance between sparsity and robustness. Our investigation uncovers that incorporating sparsity bolsters network robustness without compromising accuracy, even in the face of edge constraints. To address data and class imbalance challenges, we devise an end-to-end sparsity method that demonstrates remarkable effectiveness in a variety of real-world edge applications.

We then further improve our understanding of the vital part that parameter efficiency contributes to the model generalization, particularly in edge environments. By examining the complexity of the neural network loss landscape, we gain valuable insights into the geometry of solutions in the neural networks' loss landscape. Our investigations not only help us understand the loss landscape of neural networks but also illuminate how different solutions are connected. Moreover, we propose the permutation invariance conjecture, which offers a novel perspective on the shape of the loss landscape. We provide theoretical and empirical support for our conjecture on permutation invariance in the loss landscape and introduce the REPAIR method to enhance the performance of aligned interpolated networks.

We then move towards a more comprehensive understanding of neural network generalization by exploring a complementary and crucial facet: the role of data in the generalization of neural networks. We concentrate on the crucial role that pre-training data distribution plays in transfer performance, taking into account the unique challenges present in edge environments *e.g.*, limited access to labeled data, constantly changing data sources, and resource constraints. By addressing these challenges, we aim to enhance transfer learning, enabling neural networks to adapt more effectively to

the dynamic conditions of edge environments. Our findings reveal that variations in pre-training data distributions and as well as methods can lead to differences in downstream transfer accuracy, especially in the few-shot transfer regime. We discover that leveraging more pre-training data may help bridge the performance gap between different training methods, such as supervised and contrastive approaches.

We hope that this research contributes to a deeper understanding of neural networks at the edge and provides valuable insights for future work on neural network optimization and deployment in edge environments. The empirical findings and methodologies presented in this thesis can serve as a foundation for the development of more efficient, scalable, and robust deep learning models, ultimately enabling their successful integration into edge devices across a wide range of applications.

Kurzfassung

Die zunehmende Verbreitung von tiefen neuronalen Netzen im täglichen Leben macht deutlich, dass die Herausforderungen und Einschränkungen, mit denen sie beim Einsatz auf Edge-Geräten konfrontiert sind, angegangen werden müssen. Diese Geräte unterliegen Beschränkungen in Bezug auf Ressourcen, Zugänglichkeit und Skalierbarkeit und sind dynamischen Veränderungen in ihrer Arbeitsumgebung ausgesetzt. Dies macht die Optimierung und Verallgemeinerung neuronaler Netze zu einem wichtigen Forschungsbereich. In dieser Arbeit wird ein empirisch-wissenschaftlicher Ansatz verfolgt, bei dem Deep Learning als ein beobachtbares und experimentelles Phänomen betrachtet wird. Unser primäres Ziel ist es, zu verstehen, wann neuronale Netze funktionieren und wann sie nicht funktionieren, und wir konzentrieren uns auf die Untersuchung der Optimierung und Generalisierung neuronaler Netze im Kontext von Edge-Umgebungen.

In unserem Bestreben, die Leistung neuronaler Netze bei eingeschränkten Ressourcen zu optimieren, untersuchen wir das empfindliche Gleichgewicht zwischen dünnbesetzten Matrizen (Sparsity) und Robustheit. Unsere Untersuchung zeigt, dass die Einbeziehung von Sparsity die Robustheit des Netzwerks erhöht, ohne die Genauigkeit zu beeinträchtigen, selbst angesichts der von Edge-Umgebung bedingten Beschränkungen. Um die Herausforderungen von Daten- und Klassenungleichgewicht zu bewältigen, entwickeln wir eine End-to-End-Sparsity-Methode, die in einer Vielzahl von realen Edge-Anwendungen bemerkenswert effektiv ist.

Anschließend verbessern wir unser Verständnis der essentiellen Rolle, welche die Parametereffizienz bei der Modellgeneralisierung spielt, insbesondere in Edge-Umgebungen. Durch die Untersuchung der Komplexität der Loss Landscape neuronaler Netze gewinnen wir wertvolle Erkenntnisse über die Geometrie der Lösungen in der Loss Landscape neuronaler Netze. Unsere Untersuchungen helfen uns nicht nur, die Loss Landscape neuronaler Netze zu verstehen, sondern beleuchten auch, wie verschiedene Lösungen miteinander verbunden sind. Darüber hinaus schlagen wir die Permutationsinvarianz-Vermutung vor, die eine neue Perspektive auf die Form der Loss Landscape bietet. Wir liefern theoretische und empirische Unterstützung für unsere Vermutung zur Permutationsinvarianz in der Loss Landscape und stellen die REPAIR-Methode vor, um die Leistung von interpolierten Netzen zu verbessern.

Anschließend gehen wir zu einem umfassenderen Verständnis der Gener-

alisierung neuronaler Netze über, indem wir eine ergänzende und entscheidende Facette untersuchen: die Rolle der Daten bei der Generalisierung neuronaler Netze. Wir konzentrieren uns auf die entscheidende Rolle, die die Verteilung der Daten vor dem Training für die Leistungsfähigkeit nach einem Transfer spielt, und berücksichtigen dabei die einzigartigen Herausforderungen, die in Edge-Umgebungen bestehen: begrenzter Zugang zu gelabelten Daten, sich ständig ändernde Datenquellen und Ressourcenbeschränkungen. Durch die Bewältigung dieser Herausforderungen wollen wir das Transfer-Lernen verbessern und neuronale Netze in die Lage versetzen, sich effektiver an die dynamischen Bedingungen von Edge-Umgebungen anzupassen. Unsere Ergebnisse zeigen, dass Variationen in der Verteilung der Pre-Training-Daten und -Methoden zu Unterschieden in der Downstream-Transfergenauigkeit führen können, insbesondere bei dem Schema für Few-Shot Transfer. Wir stellen fest, dass die Nutzung von mehr Pre-Trainingsdaten dazu beitragen kann, die Leistungslücke zwischen verschiedenen Trainingsmethoden, wie z. B. überwachte und kontrastive Ansätze, zu schließen.

Wir hoffen, dass diese Forschung zu einem tieferen Verständnis von neuronalen Netzwerken in Edge-Umgebungen beiträgt und wertvolle Erkenntnisse für zukünftige Arbeiten zur Optimierung und zum Einsatz neuronaler Netzwerke in Edge-Umgebungen liefert. Die in dieser Arbeit vorgestellten empirischen Erkenntnisse und Methoden können als Grundlage für die Entwicklung effizienterer, skalierbarer und robuster Deep-Learning-Modelle dienen und letztlich deren erfolgreiche Integration in Edge-Geräte für eine Vielzahl von Anwendungen ermöglichen

Acknowledgment

This PhD thesis, a testament to several years of diligence and perseverance, is not a solitary achievement. Instead, it's an embodiment of the love, support, and guidance I have received from many pivotal people in my life. I pause here to express my profound gratitude towards them.

First and foremost, to my amazing wife, Neshat. Your ceaseless love and support have been my North Star, guiding me throughout this journey. During the stormy days and the sunny ones, you've stood by my side, sharing in both my struggles and my triumphs. This endeavor would not have been possible without your patience and your unwavering belief in me. I am deeply grateful for your love and for your devoted partnership.

I am enormously grateful to my PhD supervisor, Prof. Olga Saukh. Your wisdom and mentorship have been instrumental throughout this journey. You've guided me not only in my academic work but also taught me how to tackle challenges with a critical mind and a determined spirit. The soft skills I've learned from you are just as valuable as the technical ones, and for that, I am deeply appreciative.

This thesis is lovingly dedicated to my late father, a man who stood as a shining example of true integrity. His lessons on doing what's right continue to guide me every step of the way. His sudden loss lingers — an unexpected call one night, by his side the very next day, and merely a week later, the unforgiving grip of COVID had tragically taken him from us. Despite this profound sorrow, I find solace in the belief that he would have been brimming with pride at this achievement. I ardently hope that my work serves as a fitting homage to his cherished memory.

My mother deserves special recognition for her enduring support that began from my earliest days in elementary school. Her limitless love has been the bedrock upon which I've built my academic pursuits. Her strength and kindness continue to inspire me, and I'm deeply thankful for her. A special mention to my brothers, my family, and my friends who have been my pillars of strength throughout this journey.

I owe a tremendous amount of gratitude to my collaborators, especially Dr. Behnam Neyhabur and Dr. Hanie Sedghi. Working with you has been an incredible learning experience. Your generous sharing of knowledge and your dedication to high-level research have profoundly enriched my professional development. Your mentorship has been invaluable, and I am deeply thankful for the opportunity to learn and grow with you.

Last but not least, I dedicate this work to the Woman-Life-Freedom movement and to a young hero, Kian Pirlakal, whose life was cut short at the age of 9 by the Iranian regime. His soul serves as a poignant reminder of the struggles for freedom and justice in my country. His story inspires me and strengthens my commitment to contribute towards a better world through

my research.

This journey, while long and sometimes challenging, has been one of the most rewarding experiences of my life. Every person I've mentioned here has played a significant role in shaping my path. As I embark on the next chapter of my life, I carry their love, support, and wisdom with me. To all of you, my heartfelt thanks.

Contents

1	Introduction	1
1.1	Resource demands of neural networks	3
1.2	Parameter efficiency of neural networks	4
1.2.1	Sparsity	5
1.2.2	Ensembles	8
1.3	Data efficiency of neural networks	12
1.3.1	Data augmentation	13
1.3.2	Pre-training and transfer learning	15
1.4	Thesis outline and contributions	16
2	Pruning and Generalization	19
2.1	Sparsity and robustness of neural networks	20
2.1.1	Methodology	20
2.1.2	Robustness to weight perturbation	22
2.1.3	Robustness to data corruption	24
2.1.4	Robustness to adversarial attacks	24
2.1.5	Discussion	25
2.2	Impact of supervision on sparsity	25
2.2.1	Methodology	26
2.2.2	Impact of sparsity on distribution of PIEs	28
2.2.3	Impact of sparsity on representation quality	29
2.2.4	Impact of sparsity on sample difficulty	29
2.2.5	Discussion	30
2.3	Class-dependent pruning of deep neural networks	31
2.3.1	Methodology	33
2.3.2	Experimental setup	36
2.3.3	Evaluation of the proposed method	37
2.3.4	Discussion	38
2.4	Deep neural network pruning for nuclei instance segmentation	38
2.4.1	Methodology	39
2.4.2	Segmentation and regression models	41
2.4.3	Network-wide and layer-wise pruning	42
2.4.4	Evaluation of the sparse instance segmentation model	43
2.4.5	Discussion	45
2.5	Conclusion	46

Contents

3 Loss Landscape and Generalization	48
3.1 Background	48
3.2 Mode connectivity of neural networks	51
3.2.1 Loss barrier	53
3.3 Empirical investigation of barriers	53
3.3.1 Effect of width	54
3.3.2 Effect of depth	55
3.3.3 Effect of task difficulty and architecture choice	55
3.4 Role of invariance in loss barriers	56
3.4.1 Permutation invariance	57
3.5 Permutation invariance conjecture	58
3.6 A theoretical result	59
3.7 Direct empirical evaluation	59
3.8 Search algorithms for finding a winning permutation	60
3.8.1 Simulated Annealing (SA)	60
3.8.2 Functional Difference (FD)	62
3.8.3 Optimal Transport Fusion (OPT)	62
3.8.4 Correlation of activations	63
3.9 Identifying the problem: Variance Collapse	64
3.9.1 Why does the variance collapse phenomenon occur? .	66
3.10 REPAIR	66
3.10.1 Closed-form approximate variant	67
3.10.2 Forward-pass exact variant	68
3.11 Effectiveness of REPAIR	70
3.11.1 REPAIR for ImageNet	70
3.11.2 Split data training	71
3.12 Conclusion	72
4 Pre-training and Generalization	74
4.1 Background	74
4.1.1 Multimodal architectures	78
4.1.2 Multimodal datasets	79
4.2 The role of pre-training data in transfer learning	82
4.3 Related works	83
4.4 Methodology	84
4.5 Research questions	85
4.5.1 What is the impact of different pre-training data sources on transfer learning?	86
4.5.2 Which data distribution is better for transfer learning? .	86
4.5.3 How much pre-training contributes to downstream performance as opposed to training from scratch?	87
4.5.4 Do well-curated pre-training datasets lead to better transfer?	87

Contents

4.5.5	How does the effectiveness of ImageNet pre-training compare to that of LAION pre-training?	89
4.5.6	How does the downstream performance improve as more data is available for pre-training?	90
4.5.7	Effect of pre-training loss	91
4.6	Conclusion	94
5	Conclusion and Outlook	96
	Bibliography	98

1 Introduction

In recent years the success of neural networks has been a significant driving force behind the rapid growth of artificial intelligence. The capabilities of deep learning models in image recognition were first demonstrated almost 10 years ago (Krizhevsky et al., 2012), paving the way for many subsequent advances. One example is the Transformer architecture (Vaswani et al., 2017), which has become the foundation for numerous state-of-the-art models in natural language processing such as GPT (Radford et al., 2018).

The impressive achievements of neural networks can be largely attributed to advancements in training techniques and optimization algorithms, along with the significant role of computation and large-scale datasets. As the size and complexity of neural network models grow, the computational resources required for their training and operation increase substantially. Powerful hardware such as GPUs and TPUs, and efficient distributed computing techniques have made it possible to train these large-scale models. Additionally, the availability of vast, high-quality datasets has been crucial, as these models typically perform better with more data. The data serves as the 'fuel' for the learning process, enabling the models to recognize patterns, make predictions, and improve their performance. This symbiosis between improved algorithms, advanced computation capabilities, and large-scale datasets has been a driving force in the success of neural networks.

During training of a neural network, an optimization algorithm is used to find the best set of parameters that minimize the training error (Duchi et al., 2011; Kingma & Ba, 2014; Robbins & Monro, 1951; Tieleman & Hinton, 2012). However, the goal is not only to achieve low training error but also to achieve good generalization performance on unseen data (LeCun et al., 2015; Neyshabur et al., 2017; C. Zhang et al., 2017). The generalization error, also known as the test error, is the error that the model makes on new, unseen data. In general, the goal is to find a set of parameters that results in low training and test errors. However, in practice, it is often the case that models with low training error do not necessarily have low test error.

In contrast to classical machine learning, where increasing model complexity results in overfitting (low generalization), neural networks trained with a large number of parameters compared to the amount of training data still demonstrate good generalization performance (Neyshabur et al., 2014; C. Zhang et al., 2017). Overfitting occurs when a model memorizes the training data, rather than learning the underlying patterns, which leads

to poor performance on new, unseen data. The study of generalization can be divided into two main components: in-distribution generalization and out-of-distribution generalization.

In-distribution generalization (ID) refers to the neural network's ability to perform well on new, unseen data that is drawn from the same distribution as the training data (Arora et al., 2018). One of the main challenges in training neural networks is ensuring that they generalize well to in-distribution data. This is particularly difficult due to a large number of parameters in modern neural network architectures, that can lead to overfitting if not properly regularized.

There are several approaches to improving in-distribution generalization in neural networks. One popular approach is to use regularization techniques, such as weight decay, dropout (Srivastava et al., 2014), and early stopping (Raskutti et al., 2014), which help to prevent overfitting by reducing the capacity of the model. In practice, we split the dataset into three parts, including train, validation, and test set, and use the validation set to tune the learning parameters such as the early stopping threshold. Data augmentation is another approach that increases the diversity of the training set and reduces overfitting chances (Shorten & Khoshgoftaar, 2019). Additionally, the choice of architecture is essential, with Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) being well-suited for image and sequential data, respectively, and achieving good generalization performance on in-distribution data (Hochreiter & Schmidhuber, 1997; LeCun et al., 2015). In-distribution generalization can also benefit from robust architectures, such as those achieved through adversarial training (I. J. Goodfellow et al., 2014). Techniques like transfer learning, which involve pre-training a neural network on a large dataset and then fine-tuning it on a smaller target dataset, have also proven effective at improving in-distribution generalization.

Out-of-distribution generalization (OOD), on the other hand, refers to a scenario where the joint distribution of inputs and outputs differs between training and test sets (also referred to as distribution shift), *i.e.*, $P_{train}(x, y) \neq P_{test}(x, y)$. Neural networks learn to identify patterns and relationships in the training data, but when faced with new data from a different distribution, their performance may suffer (Hendrycks & Gimpel, 2016). This issue arises mostly because neural networks are highly expressive models capable of fitting the training data closely, which is advantageous for in-distribution data. However, this characteristic also makes the network sensitive to the peculiarities of training data, potentially hindering its ability to generalize well to new data with distinct properties (Hendrycks & Gimpel, 2016). Poor out-of-distribution generalization can also result from a lack of diversity in the training data. Therefore, a diverse training dataset is crucial when training a neural network.

Understanding the loss landscape of neural networks plays a crucial role in improving generalization. The loss landscape is a high-dimensional representation of the relationship between the model’s parameters and its objective function, *i.e.*, the loss, providing insights into the optimization process. The shape of the loss landscape can be affected by several factors, including the model architecture, the amount, quality, and distribution of the training data, and the choice of the optimization algorithm, and can have a significant impact on the generalization performance of neural networks. For instance, a flat loss landscape, which is characterized by many local minima that have similar training error values, is associated with more robust and generalizable solutions, as small perturbations in the model parameters result in less significant changes in the loss (Keskar et al., 2016). Consequently, developing optimization algorithms that favor flatter minima can lead to models with improved generalization capabilities.

On the other hand, a sharp loss landscape, which is characterized by a single global minimum with a low training error and many high error values around it, is generally considered to be less favorable for generalization. This is because it forces the optimization algorithm to converge to a single solution, which may not generalize well to unseen data. Understanding the loss landscape allows for the identification of areas with high loss values, facilitating the design of techniques that avoid these regions and ultimately leading to better-performing models.

1.1 Resource demands of neural networks

Neural networks are computationally intensive due to their complex architecture and the amount of data required for training (LeCun et al., 2015). A neural network is composed of multiple interconnected layers of neurons, which perform mathematical operations on the input data to produce an output. During training, the network adjusts the weights of these connections based on the input data to improve its accuracy. This process of adjusting the weights requires significant computational resources, including memory, processing power, and storage.

Resource demands of neural networks arise from their complexity and the massive amount of data needed for training. The number of neurons, layers, and connections in a neural network can be massive, particularly in deep neural networks with many layers (I. Goodfellow et al., 2016). Each layer adds more complexity to the network, requiring additional computations and memory. This complexity can also increase the time required for training and inference, making it challenging to use neural networks for real-time or time-sensitive applications.

Neural networks also require extensive training data to learn patterns and

relationships within the data (C. Zhang et al., 2017). The size of the training data can range from hundreds to millions or even billions of samples, depending on the complexity of the task and the size of the network. This large amount of data requires significant storage and processing power, particularly for training neural networks with larger and more complex datasets. High computational requirements limit the scalability and accessibility of neural networks. In addition, the high resource demands of neural networks can limit their deployment on resource-constrained devices, such as mobile phones or embedded systems, where energy efficiency and performance are vital. As an illustration, the MegatronLM has 8.3B parameters (Shoeybi et al., 2019), while T5 has 11B parameters (Raffel et al., 2020). The T-NLG model has even more parameters, with 17B (Rosset, 2020), and the current state-of-the-art model for natural language processing, GPT-3 (Brown et al., 2020) from OpenAI, contains a staggering 175 Billion Parameters (no available data for the number of parameters of GPT-4). Training and utilizing these models is achievable solely via extensive parallelization, necessitating the use of thousands of GPU units. This leads to considerable economic and ecological consequences. For example, the expense to train one instance of GPT-3 can be around 12 million (Wiggers, 2020).

1.2 Parameter efficiency of neural networks

One promising direction to circumvent the resource demands of neural networks is to enforce parameter efficiency. In Convolutional Neural Networks (CNNs), instead of connecting every pair of neurons between every two layers, we *prune* connections and keep only the local surrounding connections based on the convolution kernel size, padding, and dilation. The resulting Locally Connected Network (Ngiam et al., 2010) contains unique filters for each output neuron, specializing in different spatial regions. Another step of parameter efficiency is utilized by weight sharing across filters to provide translational equivariance (Hoefler et al., 2021). Figure 1.1 shows these two steps for convolutional operators as sparse fully-connected operators.

Previous research has explored various methods to decrease the size, training and inference cost of large-scale DNNs while maintaining performance levels, including quantization, knowledge distillation, neural architecture search, low-rank compression, and sparsity. The main focus of this thesis is to employ sparsity and therefore next section looks into sparsity in detail. For details on other methods refer to Entezari and Saukh (2019) and Hoefler et al. (2021).

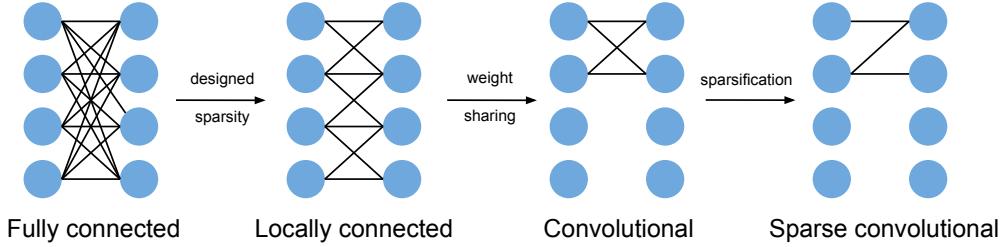


Figure 1.1: **Convolutional Neural Network as an example of parameter efficiency (Hoefler et al., 2021)** One-dimensional convolutional operators can be seen as sparse fully-connected operators for a single input and output channel. Additional sparsity can also be applied on top of the convolutional network.

1.2.1 Sparsity

Figure 1.1 shows that further parameter efficiency can also be introduced in CNNs by utilizing *sparsity*. Sparsity refers to having a high proportion of zero values in a matrix or tensor. The motivation behind sparsity in neural networks has evolved over time. Initially, the focus was on using model sparsification to enhance generalization (LeCun et al., 1990). There are various methods to encourage sparsity, including L1 regularization and incorporating sparsity constraints into the optimization objective (Olshausen & Field, 1997; Tibshirani, 1996). However, due to the increasing computational demands of neural networks, recent sparsity research (also known as pruning) has shifted toward reducing the cost of inference and, more recently, minimizing the cost of training (Frantar & Alistarh, 2023; Gale et al., 2019; Nikdan et al., 2023). Sparse neural networks can potentially offer reduced computational complexity and memory requirements, as well as increased efficiency (Han, Pool, et al., 2015). Last but not least, some researchers look into sparsity in order to gain a deeper understanding of the learning process in neural networks.

What can be sparsified?

Hoefler et al. (2021) provides a nice overview to answer this question. Figure 1.2 shows different options for sparsification of deep neural networks, per model or per example. Two main approaches to model sparsity include weight sparsity (also referred to as **unstructured**), and neuron sparsity. Weight sparsity leverages the empirical finding that preserving the output requires only a handful of salient weights within a layer, hence removing those weights below a certain score (Frankle & Carbin, 2019b; Guo et al., 2016; Han, Mao, et al., 2015; Hassibi & Stork, 1992; LeCun et al., 1989).

Weight sparsity has gained significant traction in the literature due to its ease of implementation and ability to achieve high pruning ratios while maintaining high accuracy (Frankle & Carbin, 2019b). The parameter budget for weight sparsification could be considered globally (*e.g.*, in magnitude pruning this translates to ranking all parameters and removing the smalleset parameters globally), or layer-wise. If the parameter budget is set globally, high levels of sparsity are achieved by removing most of the weights at a larger layer with more parameters *e.g.*, fully-connected layer. Nonetheless, certain layers with a minimal amount of parameters may be entirely maintained, as their impact on the parameter budget is negligible. These layers can possess substantial FLOPs, for instance, in an initial convolution layer where a small 3×3 kernel is applied to the entire image. Kusupati et al. (2020) induces sparsity while learning pruning thresholds thereby obtaining a non-uniform sparsity budget. Weight sparsity generally necessitates specialized hardware to expedite inference. Recently Mishra et al. (2021) and Zhou et al. (2021) have introduced pre-defined sparsity patterns supported by specialized GPUs to address this issue.

Pruning whole neurons and neuron-like components (such as filters, channels, and heads; also known as *structured sparsity*) effectively reduces the network size, resulting in decreased storage demands and enhanced real-time performance on any device (J. Chen et al., 2020; Dong, Huang, et al., 2017; Y. He et al., 2019; Kang & Han, 2020). Z. Liu et al. (2018) observed that pruning filters might be less effective if carried out layer-by-layer without taking into account the overall resulting structure.

In the context of structured sparsity, (Qu, 2022) explore on-device adaption, where the model, once deployed on an Internet of Things (IoT) device, continues to learn and adapt using local data. (Qu et al., 2022) Enforces structure-wise partial parameter updates to the last output layer of a DNN to achieve reasonable few-shot classification accuracy while still ensuring rapid generalization to unfamiliar tasks.

Ephemeral sparsity constitutes a secondary class of sparsification techniques that are exclusively used during the calculation of each example and is only relevant to that specific example. Structural sparsification is most apparent in activations *e.g.*, ReLU operator results in natural sparsification. Random activation sparsity can also be achieved through techniques like dropout (Hoefler et al., 2021).

Another set of ephemeral sparsity elements pertains to the *gradient-based* training values. The objective of gradient sparsification is to induce sparsity in parameter gradients during training. In *conditional computation*, the model determines a sparse computational pathway for each instance dynamically. This is accomplished by directing the computation through the network without engaging all of the weights (Mallya & Lazebnik, 2018; Wortsman et al., 2020).

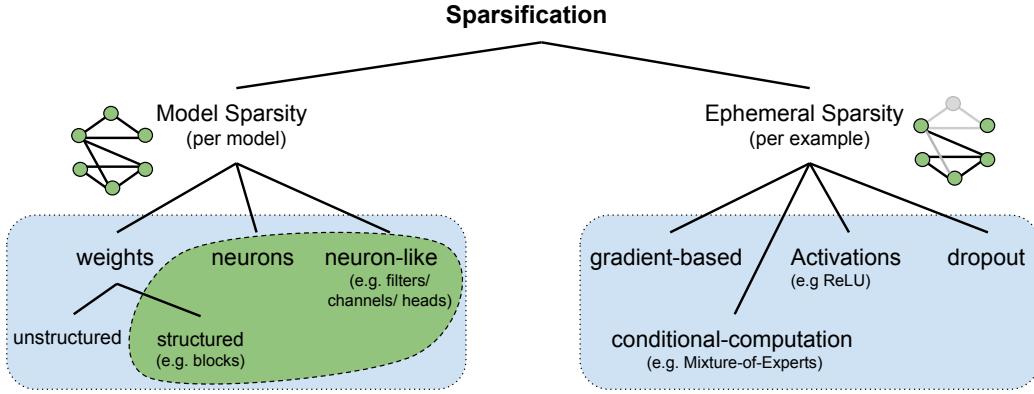


Figure 1.2: Classification of the sparsity methods (Hoefler et al., 2021).

Mixture of Experts (MoE) is another example of conditional computation that achieved impressive success in natural language processing and computer vision (Mustafa et al., 2022). In a mixture of experts (MoE) model, multiple sub-models or experts are combined to make predictions. Each expert is designed to handle a specific aspect of the input data distribution. The predictions of the experts are combined using a gating network, which decides which expert to use for a given input. Sparsity is an important concept in MoE because it allows the gating network to make efficient use of the available experts. The gating network outputs a sparse vector, meaning that only a small subset of the experts are selected for a given input. This is achieved by using a threshold or other mechanism to determine which experts are most appropriate for a given input.

Regularization-based sparsity applies regularization techniques that encourage small weights or small activations. This is accomplished by adding a regularization term to the loss function during training, which penalizes large weights or activations. The regularization term effectively imposes a constraint on the model's parameters and forcing them to remain close to zero. By doing so, the regularization term encourages the model to focus on a smaller subset of features or neurons, effectively inducing sparsity in the network. Examples of regularization techniques used for inducing sparsity include L₀, L₁, and L₂ regularization.

When to sparsify?

Model sparsity often comes with a pruning schedule. Figure 1.3 shows three different classes of pruning schedules.

Sparsity after training. The prevalent sparsification schedule, often referred to as one-shot, employs a conventional dense training method that converges after T iterations (green area in Figure 1.3). Subsequently, the fully trained model is sparsified. The resulting sparse model typically undergoes fine-

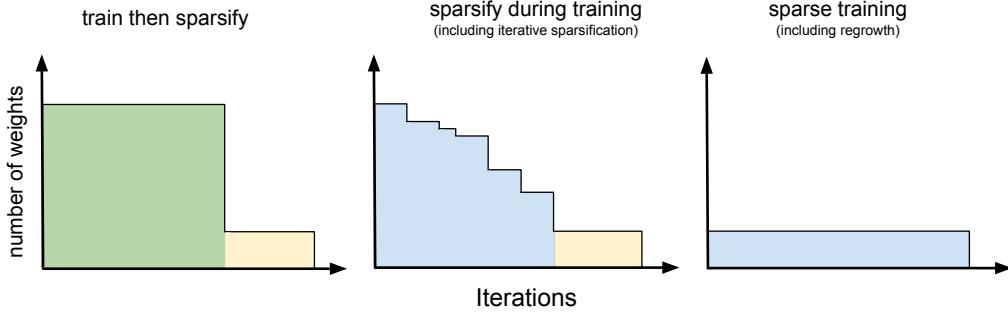


Figure 1.3: Overview of sparsification schedules (Hoefler et al., 2021).

tuning to achieve improved accuracy (yellow area in Figure 1.3).

Sparsity during training. This schedule starts the sparsification of the model before it has been trained to convergence. Compared to sparsification after training, it is cheaper (fewer training iterations) because it enforces sparsity while training is done. Regularization techniques are among this category. Sparsification during training could lead to less efficient convergence and is often more brittle to configure via hyperparameters (Ghosh & Tumer, 1994). To circumvent this challenge one may delay the starting point of pruning to later iterations (Corti et al., 2022) or use gradual schedules instead of fixed ones (Zhu & Gupta, 2017).

Sparse training. This schedule begins with a sparse model and trains it while potentially adding and removing elements during the training process. Evci et al. (2020) show that fully-sparse training can achieve ResNet-50 performance comparable to fully-dense training, albeit with extra iterations. This method is particularly beneficial for training high-dimensional models that would not fit into edge devices using dense representations.

1.2.2 Ensembles

An alternative approach to promoting parameter efficiency in neural networks is by utilizing ensemble learning. Ensemble learning combines multiple models to create a more accurate prediction or classification. By combining multiple models, an ensembled model can reduce the impact of individual model errors and provide a more reliable prediction or classification (Breiman, 1996, 2001; Hansen & Salamon, 1990). Ensemble learning can be used in a variety of machine learning applications, including regression, and classification. The prediction performance of an ensemble depends on both the individual performance of its members and their diversity (Dietterich, 2000; Z. Lu et al., 2010). Diversity in an ensemble can be achieved when the predictions of its members do not coincide on all the samples. Intuitively, when the members of an ensemble are diverse, they tend to make

independent errors, which can be beneficial for ensemble performance. This is because when the errors of diverse classifiers are aggregated, they tend to cancel each other out, leading to an improved ensemble prediction (Berend & Kontorovich, 2015; Ortega et al., 2022).

There are different types of ensemble learning methods, including bagging (Breiman, 1996, 2001), boosting (Freund, Schapire, et al., 1996), and stacking (Wolpert, 1992). Bagging entails training several models independently and then merging their results through a majority vote. Conversely, boosting involves training models in a sequential manner, with each successive model concentrating on the mistakes made by the prior one. Stacking incorporates training multiple models and utilizing their outputs as inputs for a higher-level model. (C. Zhang & Ma, 2012).

All ensemble methods encourage diversity among their individual models either implicitly or explicitly, *e.g.*, bagging and boosting implicitly encourage diversity by creating different configurations. Deep ensembles also resort to implicit techniques such as random initialization (Lakshminarayanan et al., 2017; Wen et al., 2020), tweaking the optimizer (Foret et al., 2020; Maddox et al., 2019; Wenzel et al., 2020; R. Zhang et al., 2019), or employing different hyperparameter settings (Wenzel et al., 2020; Wortsman et al., 2022).

Weight-space ensembling and output-space ensembling are two main approaches to ensemble learning for neural networks.

Output-space ensembling involves merging the predictions of multiple models to generate a final prediction. This can be achieved using techniques such as voting. One downside of this approach is that it requires saving and retrieving all individual models during inference, which can be a challenge on resource-constrained devices. This can limit the feasibility of using output-space ensembling in certain applications that require lightweight and efficient models.

Weight-space ensembling of neural networks refers to a technique in which multiple models are trained independently, and their weights are combined to produce a single model with improved performance. Unlike output-space, weight-space ensembling merges all models and therefore only one model is used for inference. Weight-space ensembling is done by averaging the weights of the independently trained models. This could be done either by averaging the weights after training is finished or during the training by using dropout. When a neural network uses dropout, every individual data point is only used to fit a random subset of the neurons. This can make the neural network more like an ensemble model of sub-networks (Srivastava et al., 2014). By combining the weights of multiple models, weight-space ensembling can reduce the risk of overfitting and improve the generalization of the final model. Training deep neural networks can benefit from the regularization provided by weight-space ensembling. weight-space ensembling requires the base models to reside in one *basin* (Chapter 3. provides

1 Introduction

the definition of basin based on Neyshabur et al. (2020a)). However, the optimization problem of training neural networks is nonconvex and can have multiple local minima, with each basin representing a different local minimum in the loss function. This can make it challenging to ensemble different models. Recent research has shed light on the properties of the loss landscape in neural network training. For instance, Liang et al. (2018) showed that the landscape is characterized by a large number of saddle points that can trap optimization algorithms, while Garipov et al. (2018) found that there exist regions in the loss landscape where a large number of solutions with similar loss values can be found. Neyshabur et al. (2020b) observe that fine-tuned models optimized independently from the same pre-training lie in the same basin of the loss landscape.

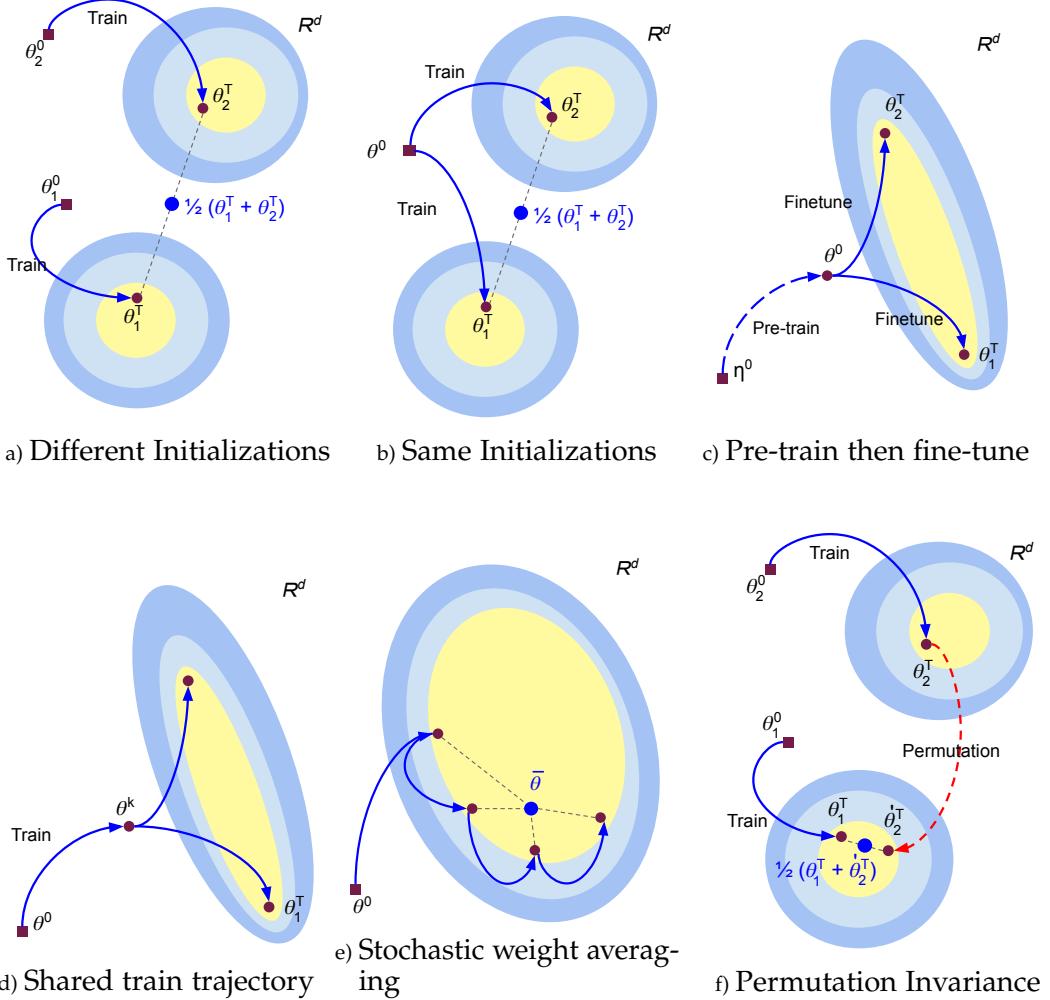


Figure 1.4: Ensemble Learning in neural networks based on mode connectivity. To create an ensemble, we need 1) functionally diverse solutions 2) All solutions need to reside in one basin. Frankle et al. (2020) show that naive weight averaging between two models trained from different initialization would end up in different basins, resulting in a low accuracy ensemble (Figure 1.4a). Neyshabur et al. (2020b) observed that fine-tuned models optimized independently from the same pre-training lie in the same basin of the loss landscape (Figure 1.4c). Frankle et al. (2020) showed that two SGD solutions will end up in one basin if their training trajectory is shared up to K iterations, where the value of K depends on both architecture and dataset (Figure 1.4d, Figure 1.4b). Stochastic Weight Averaging improves the performance of a single training trajectory. However, the solutions are not diverse for ensembling (Figure 1.4e). Entezari et al. (2021) conjecture that taking the permutation symmetries into account, we can make two different SGD solutions in one basin (Figure 1.4f).

Figure 1.4 illustrates six different efforts for ensemble learning in neural networks. Frankle et al. (2020) show that naive weight averaging between

two models trained from different initialization would end up in different basins, resulting in a low accuracy ensemble (see Figure 1.4a). Weight averaging during *a single training trajectory* (calculating the average of the weights over a certain number of iterations or epochs) has been demonstrated to reside in one basin (Figure 1.4e) and enhance the performance of models (Izmailov et al., 2019; Szegedy et al., 2016). However, because such individual models are trained from one single initialization, they lack the high diversity required for efficient ensembling. Similarly, Frankle et al. (2020) extended the results by (Neyshabur et al., 2020b) and showed that two SGD solutions will end up in one basin if their training trajectory is shared up to K iterations, where the value of K depends on both architecture and dataset (task complexity). Although the resulting models belong to a single basin, the average outcome does not enhance the performance of the individual models. Chapter 3 describes the importance of symmetries in neural networks and how permutation invariances help to make ensembles of models trained from different initialization (Figure 1.4f).

1.3 Data efficiency of neural networks

Data serves as the vital foundation for neural networks, providing the essential input needed for training and optimizing these powerful models. The importance of data for neural networks cannot be overstated, as it serves as the foundation upon which these models learn and improve their performance (LeCun et al., 2015). As the quantity and quality of data increase, the potential for neural networks to achieve remarkable results also increases drastically. From an optimization viewpoint, data is crucial in allowing neural networks to learn and fine-tune their parameters to minimize error and maximize accuracy (I. Goodfellow et al., 2016).

Neural networks are often described as being "data hungry," which speaks to the large volumes of data required to train these models effectively (Sun et al., 2017). This demand for data presents challenges in terms of both labelling and processing and highlights the importance of having robust datasets to ensure the success of neural networks in real-world applications.

Quality is another essential aspect of data for neural networks, as the performance of these models is heavily influenced by the quality of the input (C. Zhang et al., 2017). High-quality data provides a strong foundation for neural networks to effectively capture underlying relationships within the data, resulting in better generalization and performance. Conversely, low-quality data, which may be characterized by noise, inconsistencies, or biases, can negatively impact the learning process, causing the neural network to produce suboptimal or even harmful outcomes *e.g.*, in a facial recognition system if a neural network is trained on a biased dataset that predominantly

contains images of individuals from specific ethnic backgrounds, the system may perform poorly on recognizing faces from underrepresented groups. This can lead to misidentification, false accusations, or discrimination (Torralba & Efros, 2011; D. Wang et al., 2016).

The importance of data quality for neural networks becomes even more critical when considering edge devices, particularly in the context of the Internet of Things (IoT). IoT devices generate vast amounts of data, which can serve as a valuable resource for training neural networks. However, utilizing IoT data presents its own set of challenges. One notable issue is the distribution shift, as IoT devices operate in various environments under diverse conditions, causing the data collected to reflect this diversity. Consequently, the data distribution may change over time, leading to inaccuracies and inconsistencies in models trained on this data. Furthermore, data collected by IoT devices is often noisy and lacks accurate labeling, which can complicate the training process for neural networks (D. C. Nguyen et al., 2021; Tuli et al., 2019).

Several techniques have been developed to overcome these challenges, including data augmentation, pre-training, and then transfer learning. These methods aim to reduce the impact of distribution shifts by ensuring that models are trained on data that is representative for real-world distribution. Additionally, continuous monitoring of the distribution of data and adaptation of models is necessary to ensure that they remain accurate over time. In the following, we cover these methods in more detail.

1.3.1 Data augmentation

State-of-the-art models often rely on intensive data augmentation methods. This is heavily pronounced in image classification applications. Despite the well-established effectiveness of these methods, the underlying mechanism behind how these transformations operate is not well-understood. It is believed that data augmentation is effective as it simulates realistic samples from the true data distribution. As a result, it has been argued that augmentation strategies are reasonable since the transformed data closely resembles the original data, thus effectively increasing the quantity of training data available (Bellegarda et al., 1992).

Recent research results suggest that augmentation strategies are effective because they increase the diversity of samples seen by the model (Cubuk et al., 2018). Different examples of data augmentation include but are not limited to adding Gaussian noise (Ford et al., 2019; Lopes et al., 2019), erasing random patches of the training samples during training (DeVries & Taylor, 2017; Park et al., 2019; Zhong et al., 2020), and Mixup (H. Zhang et al., 2017). While each of these techniques can be effective in certain contexts, they have different strengths and weaknesses. Gaussian noise is a

simple technique that is easy to implement and has been shown effective in improving model robustness. However, it can sometimes result in overfitting if the noise is too strong. Erasing random patches can be particularly useful when working with large datasets, as it can help prevent overfitting by forcing the model to focus on important features. However, it can sometimes result in underfitting if too many patches are removed. The main idea behind MixUp is to create new training examples by blending two random images and their corresponding labels, thus encouraging the model to learn smooth transitions between classes and improving generalization. Mixup has been shown to be effective in improving model generalization. However, it can be computationally expensive, particularly when working with large datasets as it requires generating a new training sample for each pair of existing training samples, resulting in a significant increase in the size of the training set.

Gontijo-Lopes et al. (2020) argue that data augmentation improves model performance by increasing both the *affinity* between training samples and the *diversity* of the training set. Affinity quantifies how augmentation shifts data with respect to the decision boundary of the clean baseline model.

Affinity (Gontijo-Lopes et al., 2020): Let D_{train} and D_{val} be training and validation datasets drawn IID from the same clean data distribution, and let D'_{val} be derived from D_{val} by applying a stochastic augmentation strategy a , once to each image in D_{val} , $D'_{\text{val}} = \{(a(x), y) : \forall (x, y) \in D_{\text{val}}\}$. Further let m be a model trained on D_{train} and $\mathcal{A}(m, D)$ denote the model's accuracy when evaluated on dataset D . The Affinity; $\mathcal{T}[a; m; D_{\text{val}}]$, is given by:

$$\mathcal{T}[a; m; D_{\text{val}}] = \frac{\mathcal{A}(m, D'_{\text{val}})}{\mathcal{A}(m, D_{\text{val}})} \quad (1.1)$$

The affinity of one represents no shift. A smaller number suggests that the augmented data is out-of-distribution for the model. It is worth noting that affinity is easy to measure as it requires only clean training of the model in question.

Diversity, on the other hand, measures how hard augmented data is to fit. **Diversity (Gontijo-Lopes et al., 2020):** Let a be an augmentation and D'_{train} be the augmented training data resulting from applying the augmentation a . Further, let L_{train} be the training loss for a model m , trained on D'_{train} . We define the Diversity, $\mathcal{D}[a; m; D_{\text{train}}]$ as follows:

$$\mathcal{D}[a; m; D_{\text{train}}] = \frac{\mathbb{E}_{D'_{\text{train}}} [L_{\text{train}}]}{\mathbb{E}_{D_{\text{train}}} [L_{\text{train}}]} \quad (1.2)$$

Gontijo-Lopes et al. (2020) evaluate these metrics on several popular data augmentation techniques and show that explicitly optimizing along axes of both Affinity and Diversity yields better performance. These metrics can be

used to guide the development of new data augmentation techniques and to better understand the impact of data augmentation on model performance.

1.3.2 Pre-training and transfer learning

Pre-training and transfer learning are two complementary techniques in the field of deep learning that have revolutionized the way we approach machine learning problems. Pre-training involves training a model on a large dataset, which may consist of labeled or unlabeled data, to learn general features and representations. In the context of computer vision, pre-training is often performed on large labeled datasets, such as ImageNet (Deng et al., 2009; Krizhevsky et al., 2017) or recently on a large multimodal dataset, such as LAION (Schuhmann et al., 2022; Schuhmann et al., 2021). The pre-trained model can then be fine-tuned on a specific task using a smaller labeled dataset, leveraging the transfer learning paradigm.

Transfer learning improves generalization in machine learning models by allowing them to build upon the knowledge acquired during the pre-training phase. Intuitively, by learning from a large and diverse dataset, the model is exposed to a wider variety of patterns, which helps it to better generalize to unseen data. This is reminiscent of *Diversity* in data augmentation (Section 1.3.1). Neyshabur et al. (2020b) showed that feature re-use is one role player in successful transfer. One may also consider pre-training as a special case of data augmentation (Section 1.3.1), where the pre-training data covers data shifts with respect to the decision boundary of the model trained on the target task (Affinity).

Transfer learning is especially beneficial when the target task has limited labeled data, as the model can efficiently use the prior knowledge to achieve better performance compared to training from scratch. Transfer learning has demonstrated its effectiveness in various domains, including computer vision, natural language processing, and reinforcement learning (J. Howard & Ruder, 2018; Yosinski et al., 2014).

The effect of pre-training on the loss landscape shape is significant, as it results in more stable optimization landscapes compared to models trained from scratch. Pre-trained models exhibit a flatter and more convex loss landscape, which facilitates gradient-based optimization and helps the model to converge to a better local minimum (H. Li et al., 2018). Neyshabur et al. (2020b) explored the loss landscape of models that are trained from both pre-trained and randomly initialized weights. Their observations reveal that there is no notable performance barrier between the two instances of models that are initialized using pre-trained weights. This suggests that pre-trained weights effectively guide the optimization process towards a flat basin in the loss landscape (Figure 1.4c).

Geirhos et al. (2020) also showed that large-scale pre-training makes

classifiers less vulnerable to shortcut learning. In shortcut learning the model learns to exploit certain features or patterns in the data that allow it to achieve good performance on the training set but may not generalize well to unseen data. In other words, the model takes "shortcuts" by relying on spurious correlations in the data rather than learning the true underlying structure or meaningful features that are relevant to the task at hand (Geirhos et al., 2020).

Transfer learning can be generally categorized into zero-shot, few-shot, and full-shot learning, depending on the amount of labeled data available for the target task. In zero-shot learning, a model is expected to generalize to the target task without any labeled examples from that task (Xian et al., 2017). This is usually achieved by learning shared semantic representations across tasks, such as word embeddings in natural language processing or shared visual features in computer vision. The zero-shot transfer gives the ability to test the goodness of representations without finetuning. CLIP (Radford et al., 2021) and Flamingo (Alayrac et al., 2022) are two successful examples that have shown impressive results in zero-shot transfer learning tasks. Few-shot learning is particularly useful in scenarios where data is scarce, labeling data is expensive, or difficult *e.g.*, edge environments. Full-shot learning, on the other hand, involves using a larger labeled dataset for the target task, allowing the model to fine-tune and achieve better performance on the target task.

1.4 Thesis outline and contributions

In this thesis, we investigate the recent advancements in enhancing the generalization performance of neural networks on edge devices. Neural networks require substantial resources for both training and inference. This presents a challenge in edge environments where computational constraints are imposed by the device, yet data evolves over time, or deployment conditions may diverge from the measured data used in training. Our objective is to understand the deep learning phenomenon and identify strategies for improving the performance of neural networks for such resource-constrained settings.

In this journey, we primarily adopt an empirical science approach, treating deep learning as an observable and experimental phenomenon. As we navigate through real-world applications on the edge, we aim to identify consistent patterns in deep learning, transform these observations into conjectures, and carefully evaluate them through experimentation.

Chapter 2: In this chapter we dig into sparsity as a promising direction to achieve parameter efficiency of neural networks at the edge.

1. We investigate the relationship between sparsity and robustness of neural networks with respect to weight perturbation, data corruption, and adversarial attacks. We show that, up to a certain sparsity achieved by increasing network width and depth while keeping the network capacity fixed, sparsified networks consistently match and often outperform their initially dense versions. This section is based on the following paper:
Timpl, L., Entezari, R.*, Sedghi, H., Neyshabur, B., Saukh, O. (2021), Understanding the effect of sparsity on neural networks robustness, ICML Workshop on "Over-parameterization: Pitfalls and Opportunities.*
(shared first co-authorship, proposed the main idea, supervised the first author master student).
2. We look into the impact of different sparsity techniques on the representation learned by deep neural networks trained with different loss functions (supervised vs. contrastive self-supervised). We show that at high sparsity levels, contrastive learning results in a higher number of misclassified examples relative to models trained with traditional cross-entropy loss. This section is based on the following paper:
Corti, F., Entezari, R.*, Hooker, H., Bacciu, D., Saukh, O. (2022), Studying the impact of magnitude pruning on contrastive learning methods, ICML Workshop on "Hardware Aware Efficient Training*
(shared first co-authorship, supervised the first author master student).
3. Motivated by the applications of neural networks, we proposed a new loss function to train neural networks to compensate for data imbalance (*i.e.*, when the number of labeled instances of one class considerably outweighs the number of labeled instances of the other class) and class imbalance (*i.e.*, higher number of false positives may be tolerable, yet the number of false negatives must stay low). This section is based on the following paper:
Entezari, R., Saukh, O. (2020), Class-dependent pruning of deep neural networks, Second Workshop on Machine Learning on Edge in Sensor Systems (SenSys-ML)
(the main author, proposed the main idea, design and running of experiments, and wrote parts of the paper).
4. We also investigate the impact of different sparsity techniques (*i.e.*, layer-wise and network-wide magnitude pruning), on the nuclei instance segmentation performance in medical images. This section is based on the following paper:
Mahbod, A., Entezari, R.*, Ellinger, I, Saukh, O. (2022), Neural Network Pruning for Nuclei Instance Segmentation in Hematoxylin & Eosin-Stained Histological Images, MICCAI Workshop on Applications of Medical AI*
(shared first co-authorship, proposed the main idea, design and running of experiments, close collaboration with domain experts).

Chapter 3: As discussed in Chapter 1, studying the loss landscape of neural networks helps to improve generalization of these models. In this chapter, we explore the connectivity of the minimas based on the definition of the basin and investigate the impact of overparameterization on the formation of the loss landscape. We further study the symmetries in the loss landscape of neural networks and conjecture that by taking permutation invariance into account, the loss landscape can be simplified significantly resulting in a linear mode connectivity between SGD solutions. We show how extensive empirical attempts fall short of refuting it and provide a preliminary theoretical result to support our conjecture. Moreover, we identify the "variance collapse" phenomenon in interpolated networks and introduce an algorithm to address this issue, referred to as REPAIR.

This chapter is based on the following papers:

1. *Entezari, R., Sedghi, H., Saukh, O., Neyshabur, B. (2022), The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks, International Conference on Learning Representations*
(the main author, proposed the main idea, design and running of experiments, and wrote parts of the paper).
2. *Jordan, K., Sedghi, H., Saukh O., Entezari, R., Neyshabur, B. (2023), REPAIR: Linear Mode Connectivity of Deep Neural Networks via Permutation Invariance and Renormalization, International Conference on Learning Representations*
(discussions and design of experiments, wrote parts of the paper).

Chapter 4: In Chapter 4, we will have a closer look at the role of data in improving the generalization of neural networks. Specifically, we investigate the impact of pre-training data distribution on transfer learning performance. Furthermore, motivated by recent advances in large-scale multimodal pre-training, we compare the pre-training loss function (supervised vs. contrastive semi-supervised) in few-shot and full-shot transfer performance on multiple vision tasks. This chapter is based on the following paper:

1. *Entezari, R., Wortsman, M., Saukh O., Sedghi, H., Schmidt, L. (2023), The Role of Pre-training Data in Transfer Learning, ICLR Workshop on Multi-modal Representation Learning*
(the main author, proposed the main idea, design and running of experiments, and wrote parts of the paper).

Chapter 5: In this chapter, we conclude and discuss limitations and outline future research directions.

2 Pruning and Generalization

Edge devices are often deployed in diverse environments with different conditions, such as varying temperatures. Therefore data collected by edge devices can be shifted over time. The data collected by edge devices may be shifted *e.g.*, sensors might malfunction over time due to factors such as high temperature or the distribution of the gathered data might change over time.

In addition, this data is often noisy, incomplete, or corrupted due to factors like sensor inaccuracies, communication errors, or environmental influences. These reasons motivate the implementation of robust machine learning models that can handle different variations, tolerate noise and still provide accurate predictions or decisions.

In this chapter, we first introduce sparsity as one way to run neural networks on edge devices and review some background on different sparsity techniques. We also explore the interplay between sparsity and robustness to answer *does sparsity degrade or improve model robustness?* (Section 2.1). We then examine the impact of various learning techniques *e.g.*, supervised, contrastive on the generalization capabilities of sparse neural networks (Section 2.2).

Driven by edge applications, we concentrate on two inherent challenges frequently encountered in real-world situations. Firstly, real-world data typically exhibit a long-tailed distribution, where a small number of classes make up the majority of data points, while numerous other classes contain significantly fewer samples (*data imbalance*). Secondly, in many edge applications, it is often acceptable to have a higher number of false positives during model training and performance optimization. However, it is crucial to maintain a low number of false negatives, such as in early warning systems (*class imbalance*). We provide an end-to-end sparsity method by proposing parameterized loss function to fight data imbalance and class imbalance (Section 2.3).

Edge computing offers improved privacy protection compared to cloud-based solutions, as user data typically remains on the device. This is particularly important for sensitive applications like those in the medical field. Furthermore, executing AI-based solutions on the cloud requires substantial bandwidth, which may be scarce in developing and underdeveloped countries. We focus on the application of sparse neural networks for medical imaging, showcasing the potential of edge computing in addressing

real-world challenges faced by healthcare professionals. We highlight the significance of our sparsity-driven approach in medical image segmentation and explain how sparsity improves generalization (Section 2.4).

2.1 Sparsity and robustness of neural networks

Related literature refers to robustness as the network generalization ability to small shifts in the distribution that humans are usually robust to. There is a growing body of work studying methods for building robust models. Recent studies (Recht et al., 2019; Shankar et al., 2020) found that image classification models show a consistent accuracy drop when evaluated on ImageNet (Deng et al., 2009) and ImageNetV2 (Recht et al., 2019), while humans achieve the same accuracy. Another line of research aims at minimizing the worst-case expected error over a set of probability distributions by applying distributionally robust optimization (Duchi et al., 2020; Sagawa et al., 2020; Shafieezadeh-Abadeh et al., 2015). A similar line of work focuses on finding models that have low performance drop on adversarial examples (Biggio & Roli, 2018; Madry et al., 2019).

Hooker et al. (2020) showed that model sparsity, and to a smaller extent quantization, result in tremendous robustness degradation. At the same time, Golubeva et al. (2021) found that wider networks of the same capacity (same number of parameters) yield better performance. Model compression leads simultaneously to sparser and lower capacity networks, yet the contribution of both effects is mixed. Understanding the impact of these effects on model robustness in isolation is crucial when optimizing machine learning models for resource-constrained devices. In this section, we try to answer the following question:

What is the effect of sparsity on model robustness? Does sparsity degrade or improve model robustness?

2.1.1 Methodology

We hypothesize that sparsity alone does not hurt model robustness when the network capacity is fixed and provide empirical evidence to support this hypothesis in a number of settings. We run our study on a range of robustness tests (weight perturbations, data corruptions, adversarial examples), network architectures (MLPs, VGG and ResNets), datasets (MNIST, CIFAR-10, CIFAR-100), and evaluate the overall and per class network performance. We observe that for randomly initialized models with a static sparsity pattern applied before or after training, network sparsification does not hurt or even improves robustness to a certain sparsity compared to a

dense network of the same capacity. Both robustness and precision deteriorate concurrently at extremely high sparsity levels, owing to the weakened connections between the network’s layers. We show that our hypothesis holds when introducing sparsity by increasing network width and depth in separate experiments, applied before and after training. These findings show that a rapid robustness drop caused by network compression observed in the literature is due to a reduced network capacity rather than sparsity.

Experimental Framework

Robustness measures We evaluate the impact of sparsity on model performance with respect to weight perturbations (von Oswald et al., 2021), data corruptions (Hendrycks & Dietterich, 2019), and natural adversarial examples (Hendrycks et al., 2021).

1. **Weight perturbation.** Similarly to von Oswald et al., 2021, we perturb model weights by applying Gaussian noise $z_i \sim \mathcal{N}(\mu, w_i^2 \sigma_i^2)$ in proportion to the magnitude of each weight $w_i, i \in L$, and then measure the difference in the loss $\delta\mathcal{L} = \mathbb{E}_z[\mathcal{L}(w_i + z) - \mathcal{L}(w_i)]$. Accuracy drop due to model perturbation is related to the flatness of the loss landscape around the obtained optimum. Robustness to weight perturbation could also represent a proxy for quantization error. This error is introduced in neural network compression by weight quantization in the literature (Novac et al., 2021).
2. **Corrupted data.** We apply numerous algorithmically generated corruptions, similar to the ones evaluated in Hooker et al., 2020 (e.g., blur, contrast, pixelation) to all datasets used in this paper. This allows us to investigate how sensitive the sparsified models are to data corruption of different severity which humans are oblivious to. Our corrupted datasets are MNIST-C Mu and Gilmer, 2019, CIFAR10-C and CIFAR100-C Hendrycks and Dietterich, 2019.
3. **Natural adversarial examples.** We use Torchattacks (H. Kim, 2020) to generate a diverse range of adversarial attacks for different combinations of mentioned architectures and datasets. This include FGSM (I. J. Goodfellow et al., 2014), BIM (Kurakin et al., 2016), APGD (Croce & Hein, 2020), and PGD (Madry et al., 2019).

Sparsification methods. Existing literature covers multiple ways to make use of sparsity during and after model training including static and dynamic sparsity (e.g., β -Lasso (Neyshabur, 2020b)), iterative hard thresholding (e.g., Lottery Ticket Hypothesis with various pruning strategies (Frankle & Carbin, 2019b; Renda et al., 2020)) and others. Sparsification without changing the number of parameters was investigated in (Golubeva et al., 2021). In their study, static sparsity showed the most prominent impact on network

performance and is thus adopted in this work. We sparsify a network while preserving its capacity by changing the network’s width or depth. When sparsifying by increasing width, we leverage the approach introduced in Golubeva et al., 2021: every layer of the network is sparsified by removing weights at random in proportion to the layer size, using a static mask generated at initialization. This approach is referred to as *static sparsity*. We build on its publicly available implementation (Golubeva et al., 2021). Sparsifying by increasing network depth involves duplicating layers and then applying a static random mask to sparsify the weight tensors. When sparsifying by increasing depth, we consider MLP with 2^9 hidden units in each layer, and add layers of the same size. For VGG and ResNet we build architecture families VGG11, VGG13, VGG16 and ResNet18, ResNet34, ResNet50 all enjoying the default width of 64.

Sparsification schedules. In addition to static sparsity applied prior to network training, we also investigate network pruning after training by removing a certain amount of weights with the lowest magnitude to match the required sparsity level. Note that no finetuning is applied. When applying sparsity, we evaluate both the overall model performance and its performance in the most sensitive class. We follow the methodology introduced in Hooker et al., 2020 *i.e.*, evaluate the change to class level recall compared to the overall model accuracy.

Datasets and architectures. The datasets used in the experiments include MINST (LeCun & Cortes, 2010), CIFAR-10 (Krizhevsky et al., 2009), and CIFAR-100 (Krizhevsky et al., 2009). We fix the number of weights in each network architecture (one layer MLP, VGG16 (Simonyan & Zisserman, 2015), ResNet18 (K. He et al., 2015)) throughout all experiments, by increasing the width or depth and introducing the proper corresponding sparsity. See *sparsification methods* for more details. We use one layer MLP with 2^7 hidden units, VGG with 11 layers, and ResNet18 as base architectures. We refer to these vanilla architectures as to 100 %-networks before sparsification. Note that for both ResNet and VGG, our vanilla implementation uses the layer width of 16 as the base architecture, which is lower than the width of 64 used in the original architecture. We use width to set the number of output channels for the first layer and use the same width ratios as the respective vanilla architectures for the following layers. All networks were trained using SGD with a momentum of 0.9.

2.1.2 Robustness to weight perturbation

We first investigate the networks that were sparsified while growing the width to keep their capacity fixed. Figure 2.1 shows that as we move towards higher sparsity levels, the test performance first increases then decreases in extreme sparsity levels. We note that such increase is happening earlier for

2 Pruning and Generalization

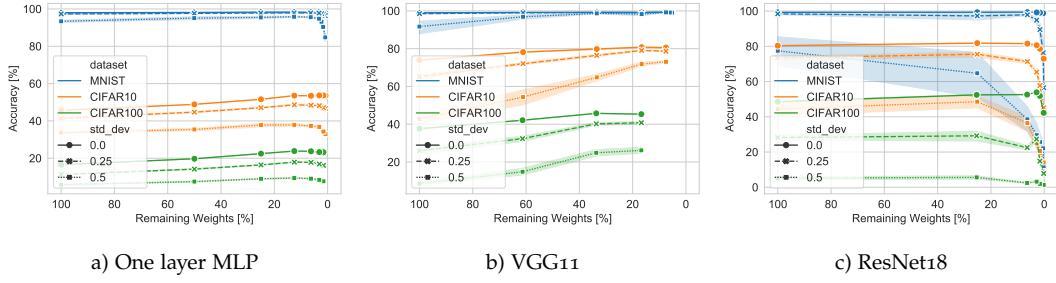


Figure 2.1: Robustness to weight perturbations, sparsification by increasing width (Timpl et al., 2021). We add multiplicative Gaussian noise $z_i \sim \mathcal{N}(\mu, w_i^2 \sigma_i^2)$ to each weight and evaluate model performance. We observe that as we move towards higher sparsity levels, the performance first increases then decreases in extreme sparsity levels. We note that such increase is happening earlier for simpler tasks like MNIST. This performance improvement indicates a flatter loss landscape around the minima suggesting better generalization.

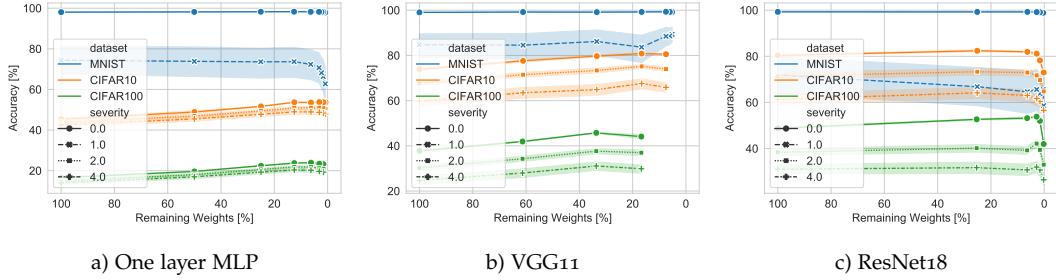


Figure 2.2: Robustness to data corruption, sparsification by increasing width (Timpl et al., 2021). We evaluate the performance of the models on corrupted datasets MNIST-C, CIFAR10-C, and CIFAR100-C. We observe that as we move towards higher sparsity levels, the performance first increases and then decreases in extreme sparsity levels. We note that such an increase is happening earlier for simpler tasks like MNIST.

simpler tasks like MNIST. We observe that sparse configurations are indeed in flatter regions of weight space as $\delta\mathcal{L}$ increases more slowly with δz_i . This suggests better robustness and generalization around the minima (Jiang et al., 2019; Pittorino et al., 2020). Each point in this plot shows the mean over five networks trained from different initializations. When sparsification is applied while increasing network depth, the maximum accuracy, and robustness are achieved for smaller depth values in all experiments. Note that keeping a network connected while increasing its depth, in contrast to width, becomes difficult with higher sparsity. For results on increasing depth refer to Timpl et al. (2021). The outcome across all experiments consistently suggests that sparsification alone does not undermine network robustness to weight perturbations as long as sufficient network connectivity is maintained.

2 Pruning and Generalization

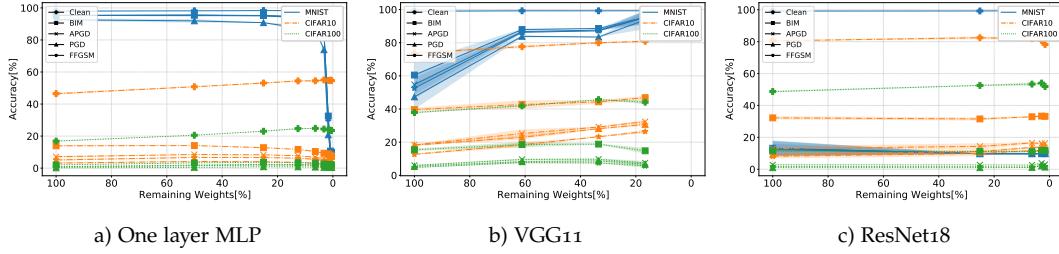


Figure 2.3: **Robustness to adversarial attacks. Sparsification by increasing width** (Timpl et al., 2021). Robustness to all adversarial attacks (BIM (Kurakin et al., 2016), APGD (Croce & Hein, 2020), PGD (Madry et al., 2019), FFGSM (I. J. Goodfellow et al., 2014)) is improved as we have less remaining weights and decreases for extreme sparsity levels where overall network accuracy (clean) drops.

2.1.3 Robustness to data corruption

Figure 2.2 evaluates the performance of the models on corrupted datasets MNIST-C (Mu & Gilmer, 2019), CIFAR10-C (Hendrycks & Dietterich, 2018), and CIFAR100-C (Hendrycks & Dietterich, 2018). We observe that as we move towards higher sparsity levels, the test performance first increases and then decreases in extreme sparsity levels. We note that such an increase is happening earlier for simpler tasks like MNIST. Each point in Figure 2.2 is the mean performance over three trained networks. For each network, we randomly sample 1000 examples from a dataset and add five noise samples in each run. On CIFAR10-C and CIFAR100-C our evaluation considers corruption severity of two and four as classified by Hendrycks and Dietterich, 2019. For detailed results on individual corruption types as well as achieved performance of networks sparsified by increasing depth refer to Timpl et al. (2021). We note that VGG networks experience convergence issues as the network sparsity approaches 10% due to lacking connectivity between layers. This is not the case for MLP and ResNet which also converge for lower percentage of remaining weights. We attribute these differences to the power of skip connections in ResNet and low overall tested network depths (1,2,4 and 8) for MLP.

2.1.4 Robustness to adversarial attacks

Figure 2.3 shows the robustness of sparsified networks when applying adversarial attacks to perturb test data. We observe a consistent trend for robustness to all adversarial attacks (BIM (Kurakin et al., 2016), APGD (Croce & Hein, 2020), PGD (Madry et al., 2019), FGSM (I. J. Goodfellow et al., 2014)). Similar to perturbed model weights and corrupted data, as we have less remaining weights, test performance for adversarial examples is first

improved and then decreases for extreme sparsity levels where the overall (clean) network accuracy drops. Dense VGG networks trained on MNIST show the highest accuracy decline in the presence of all attacks, while sparsification helps to improve adversarial robustness.

We also extend our results for sparsification after training (one-shot). Our results indicate a similar trend to the experiments with static sparsity applied at initialization *i.e.*, model sparsification often helps to improve robustness compared to a dense mode and the performance decreases in extreme sparsity levels. For plots depicting these results refer to (Timpl et al., 2021).

2.1.5 Discussion

In this section, we hypothesized that sparsity, while keeping the number of parameters fixed, does not hurt network robustness. We provide experimental evidence to support this claim based on several standard architectures, datasets, sparsification methods, and measures of robustness. Our observation is that network sparsification often helps to improve robustness compared to a dense model, yet the benefits decline together with the overall model accuracy for high sparsity levels. This is due to the increasingly loose connectivity between layers which complicates optimization. Since network capacity rather than sparsity causes accuracy and robustness drop in compressed models, designing pruning methods that treat network capacity and sparsity separately can lead to better sparsification methods. In addition, our work emphasizes the need for training procedures that better support sparse operations, which would allow for faster and more memory-efficient training of sparse networks.

2.2 Impact of supervision on sparsity

Hooker et al. (2019) demonstrated that network pruning disproportionately affects different classes, meaning that some examples are more easily forgotten than others. These examples consist of data points in the distribution’s long tail, those with incorrect or imprecise labels, or those with multiple labels, among others. Given the uneven effects of pruning, examining diverse learning algorithms becomes crucial for ensuring robust network performance. Nakkiran et al. (2020) emphasizes the distinctions between various learning algorithms in the generalization of dense networks. They found that contrastive representation learning can achieve outstanding accuracy and enhance out-of-distribution generalization. This section focuses on the influence of supervision in the neural network pruning process. We aim to address the question:

How do sparsified neural networks differ when trained with supervised versus semi-supervised loss?

This section studies the impact of different versions of magnitude pruning on the representation learned by deep models trained with supervised cross-entropy loss (Sup) and supervised contrastive learning (SCL) (Khosla et al., 2020) methods. We investigate the impact of changing the pruning schedule, comparing the impact of post-training global one-shot pruning (One-Shot) (Paganini & Forde, 2020), Gradual Magnitude Pruning (GMP) (Zhu & Gupta, 2017) and a delayed version of GMP (Δ GMP) on the learned representation. Along with the obtained test set accuracy and the analysis of Pruning Identified Exemplars (PIEs) (Hooker et al., 2019), we also evaluate the Q-Score (Kalibhat et al., 2022), prediction depth PD-Score (Baldock et al., 2021) metrics and visually inspect obtained representations with UMAP (McInnes, Healy, Saul, et al., 2018) to draw our conclusions.

In this section, we explore the influence of various magnitude pruning variants on the representations learned by deep models using supervised cross-entropy loss (Sup) and supervised contrastive learning (SCL) (Khosla et al., 2020) approaches. We assess the effects of modifying the pruning schedule and compare the impacts of post-training global one-shot pruning (One-Shot) (Paganini & Forde, 2020), Gradual Magnitude Pruning (GMP) (Zhu & Gupta, 2017) and a delayed version of GMP (Δ GMP) on the learned representations. Along with test set accuracy and the examination of Pruning Identified Exemplars(PIEs) (Hooker et al., 2019), we also evaluate the Q-Score (Kalibhat et al., 2022), prediction depth PD-Score (Baldock et al., 2021) metrics, and visually inspect the resulting representations using UMAP (McInnes, Healy, Saul, et al., 2018) to derive our findings.

- We demonstrate that models trained using SCL are considerably more affected by all examined sparsification techniques compared to those trained with supervised learning at the same sparsity levels. The negative influence is most pronounced early in the training process
- We discovered that as sparsity increases, the quality of representations learned by SCL deteriorates more than in supervised learning. This highlights the need for pruning approaches that are better suited to SCL models.

2.2.1 Methodology

Pruning methods and schedules. We implement different versions of magnitude pruning to compare the relative merits and impact of different techniques. Global post-training one-shot pruning (One-Shot) (Paganini & Forde, 2020) is the simplest method operating on a fully learned representation. We use finetuning to recover accuracy loss due to One-Shot pruning.

finetuning is performed using a Supervised loss function. Gradual Magnitude Pruning (GMP) (Zhu & Gupta, 2017) is also used in (Hooker et al., 2019) to investigate how pruning impacts different classes. At each pruning iteration for GMP, a certain percentage of the least important weights (based on their absolute magnitude) are removed from the network. The remaining weights are then finetuned to recover any loss in accuracy. This process is repeated for several iterations until the desired sparsity level is reached. A delayed version of GMP (Δ GMP) starting from epoch 50 is evaluated for contrastive models. The Δ GMP pruning is used to compare earlier pruning against later pruning in terms of the impact on the quality of the compressed representation.

PIEs. We evaluate all methods based on their generalization performance, *i.e.*, test-set accuracy. To quantify the impact of sparsity on the model performance, we assess *the level of disagreement between the predictions of pruned and non-pruned networks* on a given example. As defined in (Hooker et al., 2019), for each sample image i and a set of trained models M , we denote the class predicted most frequently by the t -pruned models by $y_{i,t}^M$ and the class predicted most frequently by non-pruned models ($t = 0$) by $y_{i,0}^M$. The sample i is classified as a Pruning Identified Exemplar (PIE) iff the label is different between the set of t -pruned models and the set of dense models.

$$\text{PIE}_{i,t} = \begin{cases} 1 & \text{if } y_{i,0}^M \neq y_{i,t}^M \\ 0 & \text{otherwise.} \end{cases} \quad (2.1)$$

Pruning introduces selective forgetting, *i.e.*, the impact of sparsity is not equally distributed amongst classes (Hooker et al., 2019). Instead, some classes are more impacted than others. This unbalanced forgetting phenomenon is stronger with lower sparsity and tends to be amplified as sparsity increases. The PIEs framework helps to quantify the impact of sparsity on the classes affected by pruning by detecting the examples for which the average prediction changes due to model pruning compared to the non-pruned version of the same model.

Training methods. We use a WideResNet (Zagoruyko & Komodakis, 2016) architecture trained on CIFAR-10 (Krizhevsky et al., 2009) throughout our experiments for SCL and Sup training. The encoder part of the network contains 690k parameters and it is followed by a classification head realizing part of the supervised task. For SCL training our setup follows the structure proposed in T. Chen et al. (2020). The encoder is first trained with a contrastive loss on an augmented version of the dataset by using a projection head discarded at the end of training, then the classification head is finetuned with supervised training.

Q-Score. Q-Score (Kalibhat et al., 2022) is an unsupervised metric to quantify the quality of the latent representation of a sample produced by an encoder

network. It can be used as a regularizer during training to let the network produce a better latent representation. Learned representations of correctly classified samples (1) have few discriminative features, *i.e.*, higher feature sparsity, and (2) the feature values deviate significantly from the values of other features (Kalibhat et al., 2022). These two properties are encoded in Q-Score defined as follows:

$$Q_i = \frac{\text{Z-Score}(h_i)}{\|h_i\|_1}, \quad (2.2)$$

where h_i is a latent representation of a sample i , $\|h_i\|_1$ is its L1 norm and Z-Score is computed as $\frac{\max(h_i - \mu_i)}{\sigma_i}$. A sample with a high Q-Score has high probability to be classified correctly (Kalibhat et al., 2022).

Prediction depth (PD-Score). PD-Score (Baldock et al., 2021) is a supervised metric that measures sample difficulty based on the depth of a network required to correctly represent the sample. It uses an encoder network where, for each representation, a k -Nearest Neighbors (kNN) (Cover & Hart, 1967; Fix & Hodges Jr, 1952) classifier is trained using hidden representations of each sample in the training set. The prediction depth score of a sample is determined by the depth from which its hidden representation is correctly classified for all the subsequent kNNs. Baldock et al. (2021) show that difficult samples, *i.e.*, samples that are easily misclassified, have high prediction depth.

UMAP. We also illustrate the UMAP of the representation vectors after applying different pruning methods. UMAP (McInnes, Healy, & Melville, 2018) is a dimensionality reduction algorithm created over a mathematical framework which preserves pairwise and global distance structures of the data projected onto a lower dimensional space. Compared to other dimensionality reduction algorithms, UMAP has no restrictions on the size of the embedding vectors.

For each sparsity and learning method, we trained $30 \times$ WideResNet16-2 models with SGD on CIFAR-10. We prune the networks to achieve sparsity levels $t \in \{0, 30, 50, 70, 90\}\%$. GMP (Zhu & Gupta, 2017) and Δ GMP prune models during training, whereas One-Shot applies pruning post-training followed by a finetuning step. Below we compare the results obtained for Sup and SCL models with respect to the distribution of PIEs and compare the representation obtained by pruned models to the one learned by uncompressed models.

2.2.2 Impact of sparsity on distribution of PIEs

Table 2.1 presents the test-set accuracy of the models along with the total count of PIEs. Comparing GMP and One-Shot pruning for Sup models reveals that the latter results in a smaller number of PIEs across all pruning

Sparsity %	Sup / GMP (Ours)		Sup / One-Shot		SCL / GMP		SCL / ΔGMP		SCL / One-Shot	
	PIE*	Acc[%]	PIE	Acc[%]	PIE	Acc[%]	PIE	Acc[%]	PIE	Acc[%]
0	-	90.58±0.38	-	90.58±0.38	-	92.06 ±0.17	-	92.06 ±0.17	-	92.06±0.17
30	188	90.5±0.39	90	90.3±0.39	180	91.58±0.2	176	91.59±0.18	233	93.28±0.17
50	179	90.4±0.34	95	90.37±0.37	241	90.92±0.18	184	91.23±0.16	236	93.06±0.2
70	204	90.14±0.34	97	90.33±0.37	350	89.58±0.2	271	90.48±0.25	235	92.38±0.17
90	273	89.47±0.36	183	89.64±0.34	748	85.56±0.4	522°	87.76±0.33°	549	87.83±0.25

Table 2.1: **Comparison of the number of PIEs for different training and pruning methods (Corti et al., 2022).** SCL models show a more pronounced drop in performance (higher number of PIEs, lower accuracy) with higher sparsity compared to Sup models. Pruning later in training (One-Shot pruning and ΔGMP) is more friendly to representation learning than early sparsification in training (GMP). *Our results differ from those reported in (Hooker et al., 2019) due to a different model architecture. °Results are aggregated over 25 models instead of 30.

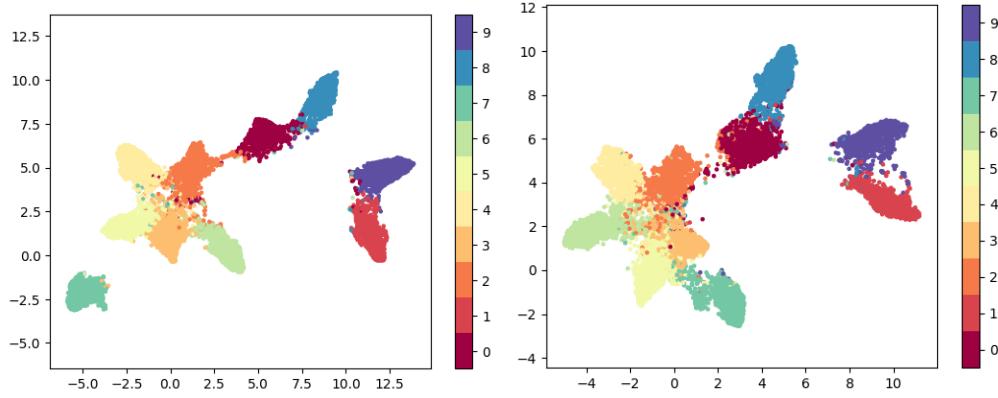
ratios. Among SCL models, the lowest quantity of PIEs is obtained by ΔGMP at all sparsity levels (excluding 70% sparsity). Additionally, we note that SCL models tend to forget more readily than Sup models at sparsity levels above 30%.

2.2.3 Impact of sparsity on representation quality

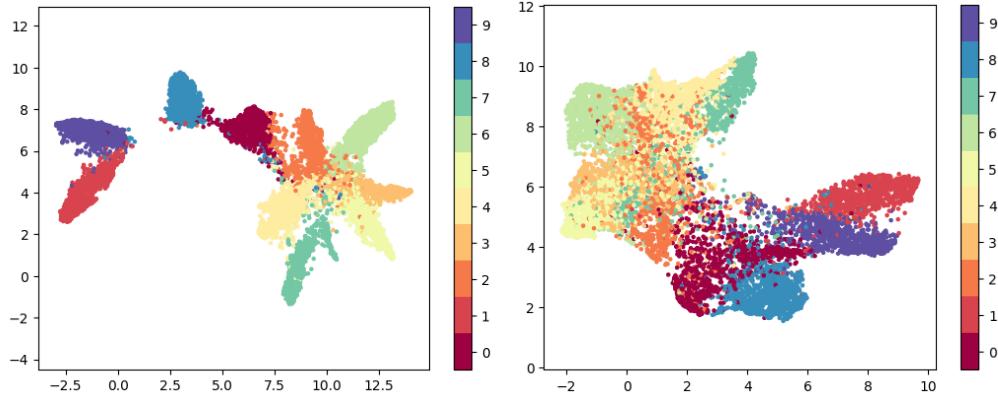
Figure 2.4 displays the UMAP projections of samples derived from both pruned and non-pruned SCL and Sup models. This visualization indicates that supervised contrastive learning is more vulnerable to reduced representation quality at elevated sparsity levels compared to supervised learning. Our results for the impact of sparsity evaluated by Q-Score also show that Q-Score decreases with higher sparsity and this trend is considerably more pronounced in SCL than in Sup models for both GMP and One-Shot pruning algorithms. See plots for the impact of sparsity evaluated by Q-Score, in Corti et al. (2022).

2.2.4 Impact of sparsity on sample difficulty

Figure 2.5 shows PD-Score of the impact of pruning on the quality of learned representations. The PD-Score for each sample is calculated as the average of 5 pruned or 5 non-pruned models. Figure 2.5 compares the PD-Score of samples processed using Sup and SCL models at 90% sparsity to the PD-Score obtained for non-pruned models. PD-Score is higher for PIEs and lower for the correctly classified samples. In the case of Sup models, a sample’s PD-Score in a non-pruned model effectively predicts its PD-Score in a pruned model. Conversely, for SCL, a sample’s PD-Score may undergo significant changes due to pruning, resulting in high variance observed in the two SCL plots on the left for GMP and pruning techniques. Similar



a) Representation by *supervised* training without pruning (**left**), pruned 95% pruned by One-Shot (**right**).



b) Representation by *supervised contrastive* training without pruning (**left**), 95% pruned by One-Shot (**right**).

Figure 2.4: Impact of pruning on the learned representation (Corti et al., 2022). UMAP diagrams of the models trained with supervised (**top row**) and supervised contrastive (**bottom row**) learning using WideResNet on CIFAR-10. Supervised contrastive learning is more susceptible to representation quality reduction at high sparsity than supervised learning. For comparison between supervised and contrastive when using GMP as pruning, refer to (Corti et al., 2022).

results are observed when using one-shot pruning. See Corti et al. (2022) for more plots.

2.2.5 Discussion

In this section, we investigate the effects of sparsity on supervised contrastive learning and compare the resulting representation quality with supervised learning, using various metrics proposed in the literature. Our research indicates that models utilizing a contrastive objective are more susceptible to the introduction of sparsity than those employing traditional supervised training with cross-entropy loss. Sparsity, in these optimization contexts,

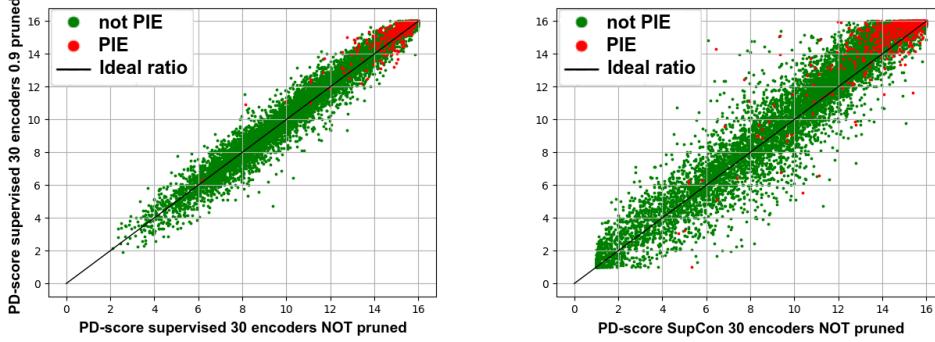


Figure 2.5: **PD-Score analysis of the impact of pruning on the quality of learned representations (Corti et al., 2022)**. In all plots PD-Score of a non-pruned network (x-axis) is compared to the PD-Score of a pruned network with 90% sparsity (y-axis). **Left:** Sup / GMP. **Right:** SCL / GMP. For Sup models, PD-Score of a sample in an uncompressed model is a good predictor of PD-Score of a pruned model (points lie close to the identity line). For SCL PD-Score of a sample may change drastically due to pruning, thus high variance in SCL plots. For comparison between supervised and contrastive one-shot pruning see (Corti et al., 2022).

considerably exacerbates class-wise error disparities at high sparsity levels. Notably, sparsifying models trained with contrastive objective results in three times the number of misclassified examples compared to supervised learning models sparsified through global post-training magnitude pruning. We hypothesized that this disparity arises from the semi-supervised objective's sensitivity to training regime, where imposing sparsity early on can be detrimental. Our modified approach, which delays pruning, enhances outcomes in a semi-supervised setting. Further explorations, such as compensating for the observed loss in accuracy by augmenting the capacity or size of the intermediate representation, are considerations we plan to investigate in future work. We hope this work sparks interest in developing more representation-friendly pruning methods for supervised contrastive learning.

2.3 Class-dependent pruning of deep neural networks

As discussed in Chapter 1, there is an unprecedented need for efficient deep learning models capable of addressing specific challenges in real-world applications, including classification or anomaly detection in the medical domain. However, these applications have to deal with both *data imbalance* and *class imbalance* when training a deep model. On the one hand, real-world data often follows a long-tailed data distribution in which a few

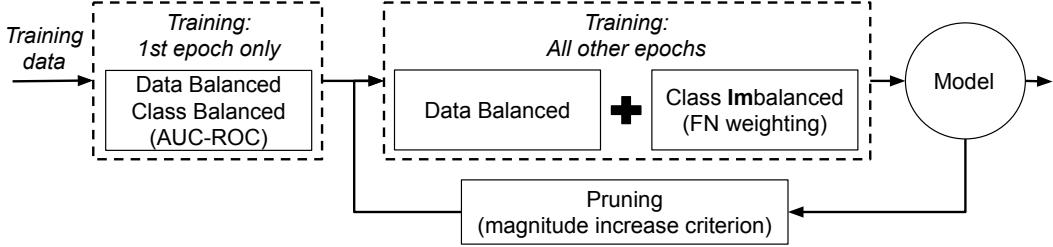


Figure 2.6: Iterative network pruning with FN optimization (Entezari & Saukh, 2019).

classes account for most of the data, while many classes have considerably fewer samples (Hasenfratz et al., 2014). Models trained on these datasets are biased toward dominant classes (Cui et al., 2019). Related literature treats data imbalance as a problem that leads to low model quality (Chawla et al., 2002). The proposed solutions typically adopt class re-balancing strategies such as re-sampling (Buda et al., 2018) and re-weighting (Y.-X. Wang et al., 2017) based on the number of observations in each class. On the other hand, there are many edge applications, which have unbalanced class importance: e.g., missing an event may have far more severe consequences than triggering a false alarm. This is especially the case in many detection scenarios and early warning systems in the IoT domain. In this section, we focus on keeping the number of FN low when pruning a deep network.

We provide an end-to-end network compression method based on the iterative *lottery ticket* (LT) algorithm (Frankle & Carbin, 2019b) for finding efficient smaller subnetworks in the original network. Since data imbalance is a common problem model designers have to deal with, *in the first step*, we extend the original LT algorithm with a parameterized loss function to fight data imbalance. Related literature suggests that a direct compensation for data imbalance into the loss function outperforms alternative methods (Cui et al., 2019). *In the second step*, we control the number of false negatives and false positives of the model by including a parameterized AUC-ROC measure into the model compression task. AUC-ROC is a comprehensive performance metric for binary classification models. It illustrates the model's capability to differentiate between positive and negative classes across all possible classification thresholds. Essentially, AUC can be interpreted as the probability that a randomly selected positive instance will be ranked higher by the model than a randomly selected negative instance, focusing on the model's discrimination capacity, not their absolute values (Fawcett, 2006).

The proposed method is comprised of the network compression pipeline outlined in Figure 2.6. Initially, the training data is utilized to train a class-balanced model, while subsequent epochs prioritize minimizing FN over FP. Our optimization function employs a combination of two loss functions to 1) address data imbalance, and 2) manage class imbalance. Specifically, we adopt a parameterized cross-entropy loss (Cui et al., 2019) to tackle the

Algorithm 1 Class-dependent network compression

- 1: Randomly initialize the network $f(x; m \odot W_0)$ with the initially trivial pruning mask $m = 1^{|W_0|}$;
 - 2: Train the network for n iterations with the class-dependent loss \mathcal{L} producing the network $f(x; m \odot W_k)$;
 - 3: Prune $p\%$ of the remaining weights with the magnitude increase strategy, *i.e.*, $m[i] := 0$ if $W_k[i]$ gets pruned;
 - 4: Replace remaining weights W_k with their initial values W_0 ;
 - 5: Go to step 2 if the next $(k + 1)$ -th round is required.
-

former, and incorporate the hinge ranking loss (Steck, 2007) to maximize AUC-ROC and regulate the trade-off between FN and FP to tackle the latter. The iterative pruning process is based on the LT algorithm.

2.3.1 Methodology

Lottery Ticket (LT) algorithm. Our class-dependent network compression method leverages the iterative pruning used to search for efficient sparse sub-networks called *lottery tickets* (LT) (Frankle & Carbin, 2019b) within an original deep network. LT postulates that within a large, randomly-initialized neural network, there exists a smaller subnetwork (referred to as the "winning ticket") that, when trained in isolation, can achieve similar or better performance compared to the original network, using fewer parameters and resources. LT starts with a randomly initialized neural network and trains the network using a standard training procedure, such as stochastic gradient descent (SGD), until it reaches a certain level of accuracy or converges. Once the network is trained, LT prunes the network by removing a fraction of the least important weights based on their magnitudes. After pruning, LT reset the remaining weights of the pruned network to their initial values (or earlier iterations in the training (Frankle et al., 2020)) and train the pruned network again using the same training procedure as before. This procedure is repeated iteratively, pruning and retraining the network multiple times to find an even smaller subnetwork with comparable performance.

Algorithm 2. provides a pseudo-code of the LT algorithm with the magnitude increase mask criterion and a class-dependent loss function \mathcal{L} explained below. The algorithm initializes the network with random weights W_0 and applies an initially trivial pruning mask $m = 1^{|W_0|}$. The operation \odot denotes an element-wise multiplication. After training the network for n iterations we prune p percent of the weights using the magnitude increase strategy by updating the mask m accordingly. The remaining weights W_k are then reset to their initial values W_0 before the next algorithm round starts.

In every round of the algorithm we minimize the loss function \mathcal{L} of the following form

$$\mathcal{L} = \mathcal{L}_{wCE} + \mathcal{L}_{SHR}, \quad (2.3)$$

where \mathcal{L}_{wCE} and \mathcal{L}_{SHR} are the *weighted binary cross-entropy loss* and the *squared hinge ranking loss* respectively detailed below.

Weighted binary cross-entropy loss. Inspired by Cui et al. (2019), we extend the notion of the classical binary cross-entropy to compensate for the data imbalance by introducing per-class weighting coefficients as follows

$$\mathcal{L}_{wCE} = - \sum_{c=1}^M \gamma_c \cdot y_{o,c} \log(p_{o,c}), \quad (2.4)$$

where γ_c are weighting coefficients for every class; M is the number of classes; $y_{o,c} \in \{0,1\}$ is a binary indicator if the class label c is a correct classification of the observation o ; $p_{o,c}$ is a predicted probability that the observation o is of class c , and n_c is the number of samples in c .

We leverage the results in Huang et al. (2016) and Y.-X. Wang et al. (2017) and handle the weighting coefficients γ_c for individual classes as $\gamma_c = \frac{1-\beta}{1-\beta^{n_c}}$, where $\beta \in [0;1)$ is a hyperparameter. In contrast to their work, we choose the value of the hyperparameter β to compensate the data imbalance in favor of a particular class. The setting $\beta = 0$ corresponds to no weighting and $\beta \rightarrow 1$ corresponds to weighting by inverse data frequency for a given class. Recent work by (Cui et al., 2019) shows that the weighting coefficients γ_c play an important role in data-balancing. In particular, when training a CNN on imbalanced data, data-balancing for each class, by means of γ_c , provides a significant boost to the performance of the commonly used loss functions, including cross-entropy.

Squared hinge ranking loss. The previous literature shows that 1) optimizing classification accuracy by minimizing cross-entropy cannot guarantee maximization of AUC-ROC (Cortes & Mohri, 2004), and 2) AUC-ROC maximization as an optimization task yields a discontinuous non-convex objective function and thus cannot be approached by the gradient based methods (Yan et al., 2003). Proposed solutions for AUC-ROC maximization (Gultekin et al., 2018) are based on approximations. In this paper, we use the squared hinge ranking loss suggested in (Steck, 2007), while adding the parameters λ_c to control the class imbalance.

$$\mathcal{L}_{SHR} = - \sum_{c=1}^M \lambda_c \max(1 - y_{o,c} r_{o,c}, 0)^2. \quad (2.5)$$

The squared hinge ranking loss is obtained from the hinge loss by replacing $p_{o,c}$ by a sorted in ascending order classifier output $r_{o,c}$.

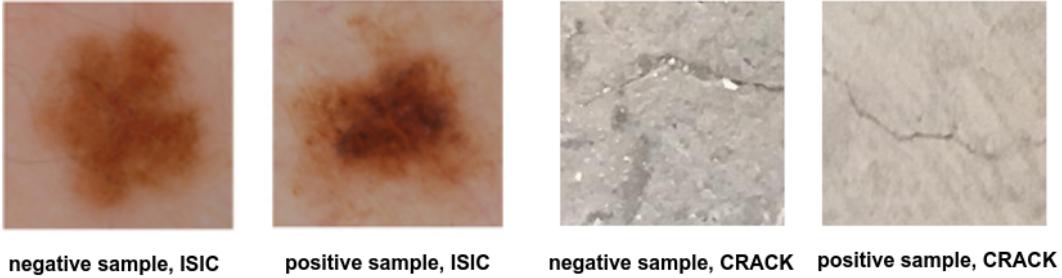


Figure 2.7: Images from the ISIC (Gutman et al., 2016) and CRACK datasets (Özgenel, 2017). Distinguishing between positive and negative samples in practical situations can be challenging and requires specialized expertise.

The authors of (Steck, 2007) show that AUC-ROC can be written in terms of the hinge rank loss as follows

$$\text{AUC-ROC} \geq 1 - \frac{\mathcal{L}_{SHR} - C}{n^+ n^-}, \quad (2.6)$$

where n^+, n^- are the number of positive and negatives samples and C is a constant independent of the rank order. Minimizing hinge ranking loss leads to AUC-ROC maximization (Steck, 2007). We use the *squared* hinge ranking loss \mathcal{L}_{SHR} to ensure our loss function \mathcal{L} is differentiable.

This section introduces the benchmark datasets, lists the metrics we use to evaluate the performance of our method, justifies parameter choices, and presents the results.

Datasets

ISIC-2016. lesion classification dataset (Gutman et al., 2016) comprises original medical images coupled with confirmed malignancy diagnosis labels obtained from expert consensus and pathology report information. Specifically, each image is assigned a label indicating whether it is benign or melanoma. The training dataset consists of 900 dermoscopic lesion images, with 173 positive and 727 negative examples, while the test set includes 379 images, with 76 positive and 303 negative samples, respectively. Examples of positive and negative samples from the ISIC dataset are displayed in Figure 2.7.

CRACK. dataset (Özgenel, 2017) comprises 40,K images, each of size 224×224 pixels, that are extracted from 500 full resolution images of 4032×3024 pixels taken from walls and floors of various concrete buildings. The camera is positioned approximately 1 m away from the surfaces, with the camera lens directly facing the target. The concrete surfaces vary in terms of surface finishes, including exposed, plastering, and paint. The label is positive if an image contains a crack and negative otherwise, and the labels are assigned

by material science experts. Positive and negative samples from this dataset are presented in Figure 2.7.

2.3.2 Experimental setup

In the case of the CRACK dataset, we enforce data imbalance by utilizing 20,K negative class images (no crack) and 4,K positive class images, with 70%, 15%, and 15% of samples allocated for training, validation, and testing, respectively.

Networks. For classification tasks involving the ISIC-2016 and CRACK datasets, we adopt AlexNet (Krizhevsky et al., 2012), which is pre-trained on ImageNet (Deng et al., 2009), and adjust the number of fully-connected layers to incorporate 256, 8, and 2 neurons. This network has a total of 2'471'842 parameters. Since our compression method employs iterative pruning based on the LT algorithm, we use these relatively shallow networks to manage the computational load on the available resources at edge devices.

Hyperparameters. To evaluate the performance of the proposed method, we follow our two-step design. First we focus on balancing the imbalanced data using the γ parameter, then we use λ multipliers for ranking loss to optimize AUC-ROC. Leveraging the results reported in (Cui et al., 2019; Huang et al., 2016) to achieve data balancing by setting the loss function parameters according to the inverse class frequencies, we set β close to 1 in our tests. For both datasets, we set $\gamma_c = 5$ *i.e.*, equal to the class frequency. We test $\lambda_c \in \{0, 1, 2, 10\}$, where $\lambda_c = 0$ stands for the standard LT algorithm. By following the magnitude increase pruning strategy we prune $p = 50\%$ of the remaining weights in every round.

To assess the effectiveness of the proposed method, we adopt a two-step approach. First, we prioritize balancing the imbalanced data by utilizing the γ parameter and subsequently employ λ multipliers for ranking loss to optimize AUC-ROC. Drawing on the results presented in (Cui et al., 2019; Huang et al., 2016) for achieving data balancing by setting the loss function parameters based on the inverse class frequencies, we set β to be close to 1 in our experiments. For both datasets, we set $\gamma_c = 5$, which is equivalent to the class frequency. We evaluate $\lambda_c \in \{0, 1, 2, 10\}$, where $\lambda_c = 0$ corresponds to the standard LT algorithm. To perform pruning, we adopt the magnitude increase pruning strategy and prune $p = 50\%$ of the remaining weights at every iteration.

Scenarios. We test our method in three different scenarios, each of which reflects a specific setting detailed below.

blue In this scenario, we test the effect of our ranking loss (class balancing) compared to the standard LT algorithm (red), without weighting cross entropy loss (data balancing). Therefore, we have $\gamma_c = 1$ and $\lambda_c = 5$.

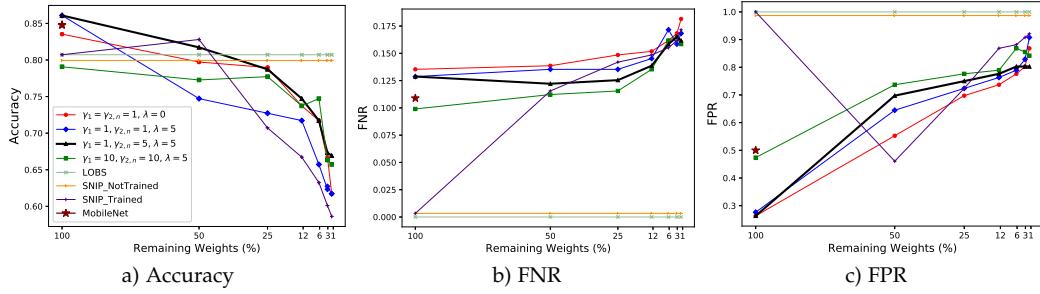


Figure 2.8: **Evaluation results on ISIC-2016** (Entezari & Saukh, 2019). Our method (black line) outperforms the LT algorithm (red line) in accuracy, FNR, and FPR for up to 1% of remaining weights in the pruned network. Our method also outperforms LOBS, SNIP, and MobileNet in AUC-ROC and FPR. Our results on CRACK dataset also show the superior performance of our method compared to the LT algorithm and other baselines. See Entezari and Saukh (2019) for CRACK evaluation and AUC-ROC plots.

black In this scenario, we test the effect of weighting cross-entropy loss (data balancing). Therefore, compared to the previous scenario (blue), we have $\gamma_c = 5$ along with $\lambda_c = 5$. We find that starting the first round with $\gamma_1 = 1$ helps the network to initially find the boundaries between the two classes without any specific focus

green In this scenario we test for weighting higher values for data balancing, so we set $\gamma_c = 10$ everywhere, while $\lambda_c = 5$.

We compare our method to the original **LT** original with the magnitude pruning criterion. The weights for both positive and negative classes are set to $\gamma_c = 1$ and we use no ranking loss with $\lambda_c = 0$. We also extend our comparison to three well-known baselines as **LOBS** (Dong, Chen, et al., 2017), **SNIP** (Lee et al., 2018), **SNIP** with finetuning, and **MobileNet** (A. G. Howard et al., 2018).

2.3.3 Evaluation of the proposed method

The evaluation results for the ISIC dataset are presented in Figure 2.8. We compare our outcomes to the best performance achieved with the conventional LT algorithm, utilizing the magnitude increase pruning criterion, along with three other recent benchmarks. Our objective is to compress the network while optimizing the classification accuracy and minimizing the number of false negatives for a specific class.

As illustrated in Figure 2.8(a), the "black" scenario, where $\lambda_c = 5$, $\gamma = 1$ for the first training epoch, and $\gamma = 5$ for subsequent epochs, yields the best accuracy for the ISIC dataset. Setting $\gamma_1 = 1$ highlights the significance of learning the class boundaries in the first iteration through balanced training. From the second iteration, we concentrate on the positive class by

employing $\gamma_{2,n} = 5$ for the desired positive class. This configuration not only outperforms the standard lottery ticket algorithm in terms of accuracy but also surpasses recent popular benchmarks, including LOBS, SNIP, and MobileNet.

Figure 2.8(b) indicates that incorporating ranking loss into the standard LT algorithm as a substitute for class balance leads to an improvement in the FNR (blue line). However, weighting the cross-entropy for data-balance results in an even stronger improvement of the FNR (black line). The optimal FNR is achieved by assigning a higher weight to the positive class by setting $\gamma_c = 10$ in all pruning iterations (green line). Comparing Figure 2.8(c) with Figure 2.8(b) reveals a trade-off between FPR and FNR. As demonstrated in the former, we attain the lowest FPR when applying neither class-balance nor data-balance. The FPR for LOBS and SNIP without training (after pruning) is also extremely high (close to 1), signifying that they wrongly classify all negative samples as positive. For more details on the results for the CRACK dataset and additional plots for the ISIC dataset, please refer to Entezari and Saukh (2019).

2.3.4 Discussion

A substantial number of edge applications must address both data imbalance and class imbalance, primarily due to the inherently dynamic environment of edge devices. On one hand, in many real-world scenarios, data follows a long-tailed distribution, where a small number of classes represent the majority of data points, and numerous other classes have significantly fewer samples. On the other hand, many edge applications exhibit unbalanced class importance, meaning that the consequences of missing an event can be far more severe than triggering a false alarm. Our network compression method combines network pruning with efficient network design to address both of these challenges. Unlike other compression methods, which try to minimize the overall error rate, our method additionally optimizes AUC-ROC while focusing on a desired class which appears to be useful in a number of real applications in the IoT domain.

2.4 Deep neural network pruning for nuclei instance segmentation

In certain real-world scenarios like telemedicine, quick diagnoses, and decision-making are crucial. Telemedicine involves delivering remote health-care services via communication technologies, enabling patients and health-care providers to interact and share information without being in the same physical location. Two primary factors present challenges to telemedicine

adoption: telemedicine can be particularly challenging in underdeveloped countries with poor Internet infrastructure. Privacy reservation is the second reason, as users often prefer to keep their data on their own devices. Moreover, the implementation of sparsity in telemedicine is highly advantageous, particularly when considering the growing need for accessible and efficient healthcare solutions. By leveraging neural network sparsity techniques, the computational demands of medical models are significantly reduced. Such compact models enable the deployment of advanced medical tools such as portable ultrasound on smartphones (Somauroo, 2019), facilitating immediate access to diagnostic information even in remote or resource-limited settings. This accessibility is essential for delivering high-quality healthcare services to a wider range of patients, irrespective of their geographical location while preserving their privacy.

In this section, we exploit sparsity for nuclei instance segmentation in Hematoxylin and Eosin (H&E)-stained histological images. This section showcases a collaboration with experts in the medical domain (Medical University of Vienna) to validate the advantages of sparsity in the medical domain. Nuclei instance segmentation refers to the process of identifying individual cell nuclei in microscopic images of tissue samples that have been stained using the H&E staining technique (Sirin et al., 2016). H&E staining is a widely used technique in histology and pathology to highlight different structures in tissue samples. Hematoxylin stains the cell nuclei blue, while Eosin stains the cytoplasm and extracellular matrix in varying shades of pink (see Figure 2.9). This contrast helps to distinguish different cellular structures and cell types. Nuclei instance segmentation plays an essential role in the analysis of histological whole slide images and can be considered as a fundamental step for further analysis (Skinner & Johnson, 2017). Parameters such as nuclei density or count can be extracted from instance segmentation masks. In the next step, this information is used for disease detection, diagnosis, and treatment planning (Jørgensen et al., 2017). Our findings indicate that by applying layer-wise pruning, the weights from the semantic segmentation and deep regression, models can be pruned with less than 2% drop in the evaluation indexes. Interestingly, our results also demonstrate that nuclei semantic segmentation is highly robust against pruning, with a barely reduced Dice score even at extreme compression ratios. Furthermore, we observe that both pruning methods exhibit high robustness against distribution shifts, highlighting their critical importance in real-world applications.

2.4.1 Methodology

In this section, we employ a recently developed model for nuclei instance segmentation (Mahbod et al., 2019), comprising two main branches: a seman-

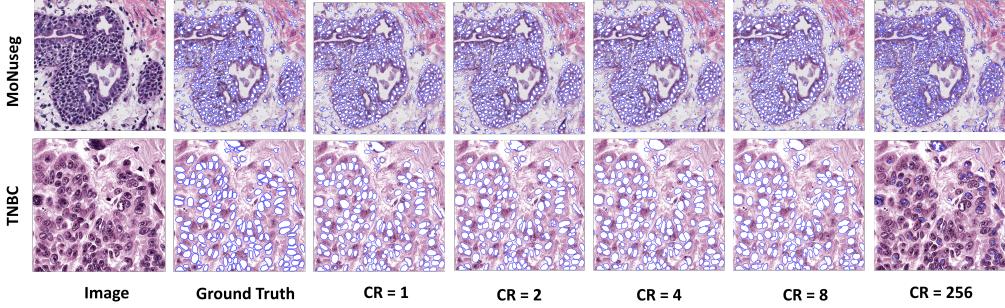


Figure 2.9: Visualizing the layer-wise pruning impact on the final instance segmentation performance (Mahbod et al., 2022). The first column shows the raw input images, the second column shows the ground truth instance segmentation masks, and the rest shows the predicted masks by the model with different compression ratios (CRs).

tic segmentation branch and a deep regression branch. We investigate the impact of DNN magnitude pruning on both branches, using two strategies, *i.e.*, networks-wide and layer-wise, and examine the resulting instance segmentation outcomes. Through this, we explore the pruning effect on three distinct image analysis tasks, namely semantic segmentation, regression, and instance segmentation.

The workflow of the DNN pruning approach for nuclei instance segmentation is illustrated in Figure 2.10, where the left branch performs semantic segmentation, while the right branch conducts deep regression. To compress the model, each branch undergoes iterative training and pruning. Finally, the final sparse networks for both branches are merged to produce the ultimate post-processed instance segmentation model.

Datasets

We evaluate the efficacy of our pruning strategies on two publicly available datasets, MoNuSeg (Kumar et al., 2020) and TNBC (Naylor et al., 2019). MoNuSeg contains 44 HE-stained histological images of fixed size 1000×1000 pixels. The image patches were extracted from the TCGA database and nine human organs (breast, kidney, liver, prostate, lung, bladder, colon, stomach, and brain). All images were acquired at $40 \times$ magnification, and the dataset includes over 28,000 manually segmented nuclei. The dataset is divided into a training set of 30 images and a test set of 14 images.

The TNBC dataset comprises 50 HE-stained histological images of fixed size 512×512 pixels, scanned at $40 \times$ magnification from one organ (breast). Over 4,000 nuclei were manually segmented in this dataset. Further details are available in (Naylor et al., 2019).

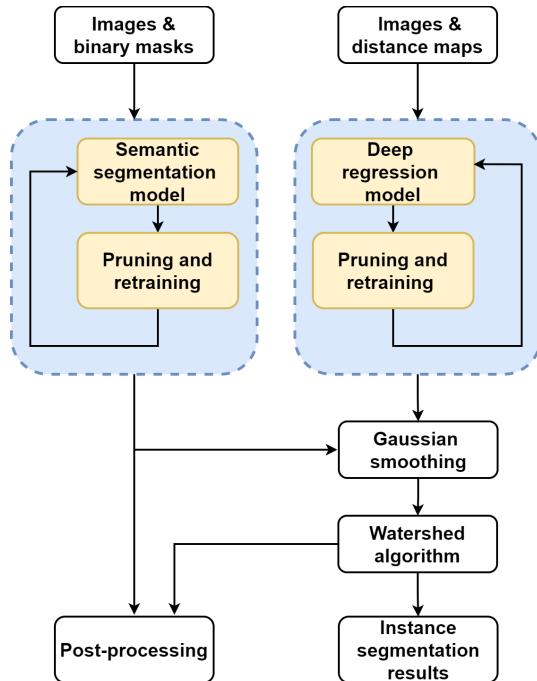


Figure 2.10: **The generic workflow of the proposed method (Mahbod et al., 2022).** Iterative pruning and training for semantic segmentation and regression models are shown in blue boxes.

2.4.2 Segmentation and regression models

Our instance segmentation model for nuclei segmentation is inspired by Mahbod et al. (2019). As shown in Figure 2.10, the proposed pruning pipeline processes two models separately, namely a semantic segmentation model and a deep regression model. The main task of the semantic segmentation model is to separate the background from the foreground, while the deep regression model predicts the nuclei distance maps. The architectures of the utilized models are the same, both being encoder-decoder-based models with skip connections between the encoder and decoder parts. Both models have the same number of trainable parameters, totaling around 24.4 million. As the semantic segmentation model deals with a binary segmentation task, we use the sigmoid activation function in the last layer with the Dice loss function. For the deep regression model, however, we use a linear activation function in the last layer with a mean square error loss function to handle the regression task. Aside from these two differences, the model architectures, training, and inference procedures are identical in both branches. We use Adam optimizer with a batch size of 2 for model training, and the initial learning rate was set to 0.001 with a cosine annealing learning rate scheduler (Loshchilov & Hutter, 2016). We train each model for 1000 epochs. As the DNN models were trained with limited training samples,

we made use of a number of augmentation techniques suggested in prior studies (Mahbod et al., 2021; Mahbod et al., 2019). We apply resizing (only for the MoNuSeg dataset to match them to 1024×1024 pixels), random horizontal flipping, random scaling and shifting, random Gaussian filtering, and random perspective transformation as the main augmentation techniques.

2.4.3 Network-wide and layer-wise pruning

In previous sections, we discussed different model sparsification strategies. Among all these methods, magnitude pruning has gained a lot of attention mainly for two reasons: it features superior performance than many other pruning strategies, and it is computationally inexpensive, *e.g.*, there is no need to compute Hessians. In this section, we employ two versions of magnitude pruning, namely network-wide (also known as global) and layer-wise magnitude pruning. Related works (Paul et al., 2022; Renda et al., 2020) show that iterative pruning achieves better performance than one-shot pruning. Therefore we also use iterative magnitude pruning for both network-wise (when we rank all weights and remove $\%X$ of all weights in each step) and layer-wise (remove $\%X$ from each layer).

In previous sections, we discussed different model sparsification strategies. Among all these methods, magnitude pruning is a popular method for DNN sparsification due to its simplicity and effectiveness. It involves setting a threshold value and removing weights below this threshold value. In network-wide magnitude pruning, a single threshold value is set for the entire network, and weights are pruned based on this threshold. On the other hand, in layer-wise magnitude pruning, a threshold value is set for each layer, and weights in each layer are pruned based on their respective threshold values.

Iterative pruning involves pruning the network multiple times, each time retraining the pruned network to recover accuracy. This allows for a more gradual sparsification process that can preserve the network’s accuracy better than one-shot pruning. In iterative magnitude pruning, a percentage of weights are pruned in each iteration until the desired sparsity level is achieved. Several recent works have compared the performance of different pruning methods and found that iterative magnitude pruning outperforms one-shot pruning (Paul et al., 2022; Renda et al., 2020).

The final instance segmentation masks were generated by merging the results from the semantic segmentation and deep regression branches, as presented in Figure 2.10. In order to prevent the detection of false-positive local maxima, we applied a Gaussian smoothing filter on the predicted nuclei distance maps that were generated by the deep regression model. The size of the kernel for the Gaussian filter was based on the estimated average nuclei size that was obtained from the segmentation model (Mahbod

et al., 2019). We then extracted local maxima from the smoothed distance maps and utilized them as seed points for the marker-controlled Watershed algorithm (Yang et al., 2006). The background for the instance segmentation masks was determined by the predicted binary segmentation masks from the semantic segmentation model. We also applied two post-processing steps that were suggested in previous studies (Kumar et al., 2020): we removed small objects from the segmentation masks (area <30 pixels) using morphological operations, and filled the holes in the predicted instances.

2.4.4 Evaluation of the sparse instance segmentation model

To assess the final instance segmentation performance, we utilized the Aggregate Jaccard Index (AJI) (Kumar et al., 2020) and Panoptic Quality (PQ) score (Graham et al., 2019) as the primary evaluation scores. The AJI is computed as the ratio of the sum of intersection areas between all matched predicted and ground-truth instances to the sum of union areas of all instances, both predicted and ground truth. On the other hand, PQ is a single score that combines segmentation quality and recognition quality. PQ is determined by multiplying the average IoU of matched predicted and ground-truth instances (segmentation quality) by the F1-score of matching pairs (recognition quality). PQ ranges from 0 to 1, where a higher score indicates better performance. Additionally, we employed Dice similarity score and Mean Square Error (MSE) as evaluation indices to assess the performance of each DNN model (i.e., semantic segmentation model and deep regression model). The dice score evaluates the similarity between the two sets and is calculated by taking twice the number of elements common to both sets and dividing by the sum of the sizes of the sets. A detailed explanation of these scores is available in (Graham et al., 2019).

In our experiments, we use the training set of the MoNuSeg dataset to train and iteratively prune the model. The performance results of the pruned models on the test set of the MoNuSeg dataset and the entire TNBC dataset are presented in Figure 2.11 and Figure 2.12, respectively.

Table 2.2: Theoretical Speedup. Layer-wise pruning yields higher speed-ups, as it prunes more weights in the early layers that have small kernels applied to a large input, requiring more FLOPS. $CR > 2^7$ is impossible to achieve with layer-wise pruning due to loss of connectivity inside the model.

Pruning Method	Compression Ratio (CR)								
	2	4	8	16	32	64	128	256	512
Network-wide	1.49	2.13	3.07	4.46	6.59	9.90	15.01	23.80	45.02
Layer-wise	1.99	3.99	7.97	15.89	31.57	62.27	121.18	-	-

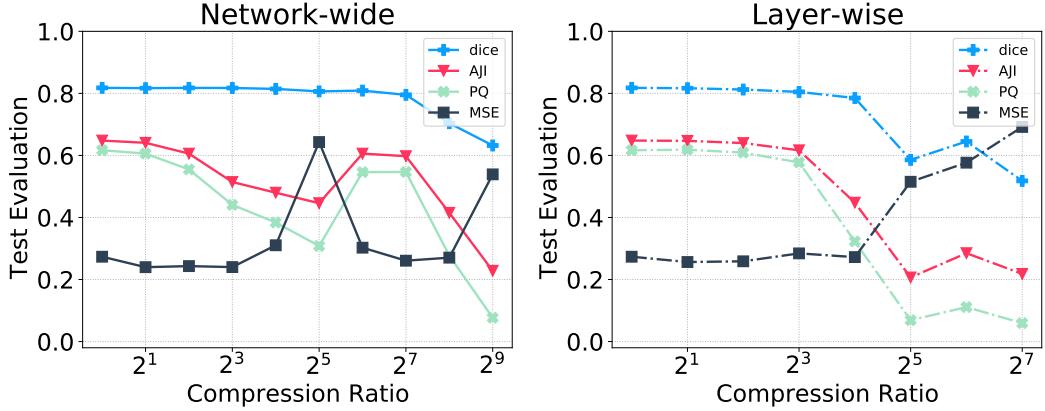


Figure 2.11: **Impact of two pruning methods on the semantic segmentation (Dice), regression (MSE), and instance segmentation (AJI, PQ) (Mahbod et al., 2022).** Performance evaluation on the MoNuSeg test set. Nuclei semantic segmentation is extremely robust against both pruning methods. Layer-wise pruning shows better performance for smaller compression ratios while enforcing larger pruning ratios to each layer harms the total performance.

Figure 2.11 shows that nuclei semantic segmentation network is extremely robust against pruning, where removing 0.992% of the parameters in network-wide ($CR = 2^7$) and 0.937% ($CR = 2^4$) in layer-wise fashion is possible with only 2% reduction in Dice. It also shows that layer-wise pruning is a better choice (for all measures and tasks) for smaller compression ratios, while enforcing extreme pruning ratios ($CR > 2^5$) to each layer harms the total performance. That is due to losing small yet important kernels in the early layers. Extreme pruning ratios ($CR > 2^8$) are also not possible to achieve with layer-wise pruning because this leads to removing the whole layer.

Another interesting observation in Figure 2.11 (network-wide) is that pruning deep regression model first increase MSE and then decrease MSE (and ultimately results in high MSE values due to extreme compression ratios). This observation is aligned to previous works (Renda et al., 2020), where the authors observe the performance improvement when pruning. Such a performance improvement results in extreme compression ratios ($CR = 2^7$) for network-wide instance segmentation pruning while only losing 2 % in AJI and PQ.

Distribution shift In the medical domain, it is quite common for data distributions to change over time due to factors such as evolving disease patterns, advancements in diagnostic techniques, and changing patient populations. As a result, it is crucial to continuously evaluate the performance of trained models to ensure their effectiveness and accuracy are maintained, even as the underlying data shifts. Figure 2.12 evaluates model performance under distribution shift, i.e. training and pruning on MoNuSeg and test on

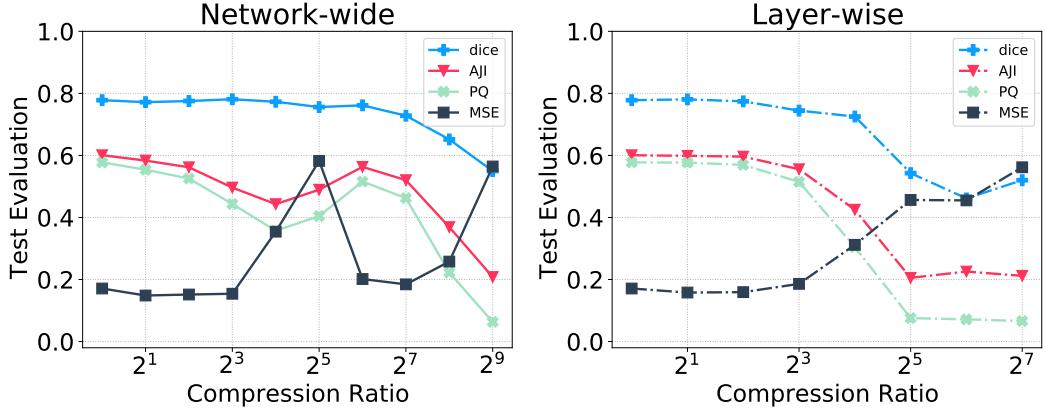


Figure 2.12: Impact of pruning on model performance under distribution shift (Mahbod et al., 2022), i.e. training and pruning on MoNuSeg and test on TNBC dataset. All evaluation metrics are identical to Figure 2.11. Layer-wise pruning shows less robustness to the distribution shift.

the TNBC dataset. All evaluation metrics are close to Figure 2.11, showing that our pruning schemes are robust to potential natural distribution shifts presented by a different dataset. Comparing network-wide and layer-wise pruning shows that the latter presents less robustness to the distribution shift. Similar evaluation patterns also can be seen in the TNBC dataset, i.e. for smaller CRs, layer-wise is a better choice for instance segmentation. We also observe the performance decrease and then increase for network-wide pruning, as better solutions are found using iterative pruning and training.

Table. 2.2 shows the theoretical speedup that can potentially be achieved by proper hardware supporting model sparsity for both pruning techniques along different CRs. As expected, layer-wise pruning gains much higher speed-ups, as it stronger prunes early layers.

Figure 2.9 provides qualitative evaluation of the impact of pruning on the final instance segmentation for different CRs. DNN pruning with small CRs ($CR \approx 2^3$ and smaller) has not drastically changed the predicted instance segmentation masks. However, the instance segmentation performance has been significantly degraded for very large CR ($CR = 2^8$).

2.4.5 Discussion

This section presents the application of two magnitude-based pruning techniques, namely network-wide and layer-wise pruning, for nuclei instance segmentation in H&E-stained histological images. Our findings indicate that by applying layer-wise pruning at a certain compression rate, the weights from the semantic segmentation and deep regression models can be pruned with less than 2% drop in the evaluation indexes. Interestingly, our results also demonstrate that nuclei semantic segmentation is highly robust against

pruning, with a barely reduced Dice score even at extreme compression ratios. Furthermore, we observe that both pruning methods exhibit high robustness against distribution shifts, highlighting their critical importance in real-world applications. Future research is required to explore alternative pruning techniques and investigate their impact on the nuclei instance segmentation performance for in-distribution and out-of-distribution regimes. Future directions include extending this work to investigate instances affected by different pruning strategies and focusing on avoiding accuracy reduction across all classes.

2.5 Conclusion

In this chapter, we investigated the interplay between sparsity and robustness in neural networks, particularly for edge devices operating in dynamic environments. We began by providing an overview of sparsity techniques and their potential impact on model robustness. Our findings demonstrated that contrary to common belief, sparsity indeed does not hurt network robustness when network capacity is properly managed. Moreover, we examined the impact of various learning techniques, such as supervised and contrastive learning, on the generalization capabilities of sparse neural networks. We observed that models trained with a contrastive objective are more sensitive to the introduction of sparsity relative to traditional supervised training with a cross-entropy loss.

Next, to tackle the challenges of data and class imbalance frequently encountered in real-world edge applications, we proposed an end-to-end sparsity method that incorporates a parameterized loss function. This method successfully addressed both data and class imbalance while optimizing the AUC-ROC and maintaining a focus on the desired class. The effectiveness of our approach was showcased in real-world edge applications.

Lastly, we highlighted the importance of domain expertise in medical imaging applications and demonstrated the potential of sparse neural networks in this area. By applying magnitude-based pruning techniques to nuclei instance segmentation in HE-stained histological images, we observed high robustness against pruning and distribution shifts, which is critical in real-world edge applications.

In conclusion, our sparsity-driven approach demonstrates significant promise for edge devices operating in diverse environments with varying conditions. By exploring the interplay between sparsity and robustness, and addressing the challenges of data and class imbalance, we have contributed to the development of more efficient, robust, and generalizable machine learning models for edge computing applications. One future direction on sparsity can be the development of adaptive pruning techniques for on-

2 Pruning and Generalization

device training and learning. This would enable efficient model optimization by dynamically adjusting the pruning process according to the device's resources and the specific application requirements.

3 Loss Landscape and Generalization

As discussed in Chapter 1, understanding the loss landscape of neural networks plays a crucial role in improving generalization. The shape of the loss landscape can be affected by several factors, including the model architecture, the amount and quality of training data, and the choice of an optimization algorithm, and can have a significant impact on the generalization performance of neural networks. In this chapter, we first introduce some background about the loss landscape of neural networks (Section 3.1). We then review the connectivity of neural networks and the geometric structure of the loss landscape (Section 3.2). We examine the loss landscape of neural networks and investigate how overparameterization influences its shape (Section 3.3).

We also look into the invariances of neural networks and their role in the shape of the loss landscape. We conjecture that the barriers between different solutions in the loss landscape are due to the symmetries of the function class (Section 3.5). Fundamentally, our conjecture suggests that accounting for permutation invariance eliminates loss barriers between various solutions, with all of them residing in the same loss landscape basin. Although this is a bold conjecture, we present theoretical support and multiple empirical findings to back our hypothesis. Furthermore, we assess different neuron alignment algorithms to identify the optimal permutation (Section 3.8).

We then investigate the internal behavior of neural networks by focusing on the statistics of hidden units and identified the "variance collapse" phenomenon, causing a rapid drop-off in the performance of interpolated networks (Section 3.9). We propose a method, called REPAIR to encounter this problem. We show that REPAIR significantly improves the performance of interpolated networks across a wide variety of architectures and datasets, supporting our permutation invariance conjecture (Section 3.10).

3.1 Background

In classical machine learning, we observe a U-shaped bias-variance trade-off, where increasing model complexity typically results in a decrease in training error and an increase in test error after reaching an optimal point.

However, for neural networks, the test error first decreases, then increases, and finally decreases again as the model size or training time increases. This phenomenon is referred to as deep double descent (Nakkiran et al., 2019). The deep double descent phenomenon suggests that overparameterized models can actually generalize better when trained for longer periods or with larger datasets (Nakkiran et al., 2019). Overparameterization in neural networks refers to the use of more parameters than necessary to solve a given task. An overparameterized model refers to a neural network with the capacity to memorize data, often having more parameters than the number of data points in the training set. The number of parameters serves as a simple proxy for measuring capacity (Neyshabur et al., 2014). Deep double descent has important implications for the design and training of deep learning models, as it challenges the conventional wisdom that simpler models with fewer parameters are always better for generalization.

Overparameterization. It has been shown that overparameterization is beneficial for improving the generalization performance of neural networks, despite additional computational and memory requirements. One reason for this is that overparameterization can enable the network to learn multiple solutions to a given task, allowing it to generalize better to new data. C. Zhang et al. (2021) show that overparameterization can improve the generalization performance of deep neural networks, particularly for image classification tasks. The authors show that when a network is overparameterized, it can learn multiple linear regions that correspond to different classes, enabling it to generalize better to new data. The authors also showed that overparameterization helps to avoid overfitting by effectively regularizing the network. This result has since been confirmed by other studies such as (Novak et al., 2018), which showed that overparameterization also helps to avoid sharp minima in the loss landscape, which can lead to poor generalization performance. In addition to generalization improvement, overparameterization can also enable the use of more efficient optimization methods. For instance, large-batch training, which involves training the network on a large batch of samples, has been shown to be more efficient for overparameterized networks (Goyal et al., 2017). This is because overparameterization can help to smooth out the loss landscape, making it easier to find a good solution using large batches.

Scaling laws. Kaplan et al. (2020) discover a set of scaling laws, specifically for large language models, but the results are applicable to most transformer models. They discover that language modeling performance progressively enhances as model size, dataset size, and compute resources for training are increased. They noted that empirical performance exhibits a power-law relationship with each individual element, when not constrained by the other two factors. This observation motivates the community for scaling neural networks to extremely large sizes both in parameter count and data size, *e.g.*,

Gopher (Rae et al., 2021), a DeepMind language model with 280B parameters (larger than GPT-3 with 175B parameters). However, Chinchilla (Hoffmann et al., 2022) (by DeepMind) could outperform Gopher despite being only 70B parameters and costing the exact same amount of computing. The findings of this study suggest that simply increasing the size of neural networks is not always the most effective approach. Rather, it is crucial to identify the appropriate balance between the number of parameters and the amount of training data for optimal neural network training. As a result, there is a growing emphasis on optimization rather than scale, in order to improve the efficiency and effectiveness of neural network training. Consequently, recent research has started to focus on model architecture and training optimization techniques to enhance the efficiency and effectiveness of neural network training without relying solely on scale (Tay et al., 2022; S. Wang et al., 2020).

A better understanding of the loss landscape allows researchers to design more efficient optimization algorithms and training strategies, ultimately leading to better generalization and faster convergence (Dinh et al., 2017). As the scale of neural networks and the complexity of tasks continue to increase, a deeper comprehension of the loss landscape becomes increasingly essential for developing models that are not only powerful but also efficient, making them more accessible and robust across various applications. Understanding the loss landscape could help researchers address overparameterization by guiding them toward models with fewer parameters. This is crucial for creating efficient models that are well-suited for resource-constrained edge devices. Furthermore, a comprehensive grasp of the loss landscape allows researchers to find solutions in the high dimensional space that demonstrate enhanced robustness to real-world variations. This quality is essential for edge devices, which often operate in dynamically changing environments.

Basin The definition of the basin is crucial for understanding the loss landscape and creating weight space ensembles. In the literature, the term "basin" is frequently used in a broad sense to denote regions in the parameter space where the loss function has relatively low values. Keskar et al. (2016) demonstrate that non-linear low-loss pathways can be identified to connect any two solutions. Given a loss function $\ell : \mathbb{R}^n \rightarrow \mathbb{R}^+$ and a closed convex set $S \subset \mathbb{R}^n$, Neyshabur et al. (2020a) defined that S is $a(\epsilon, \delta)$ -basin for ℓ if and only if S has all following properties:

- Let U_S be the uniform distribution over set S and $\mu_{S,\ell}$ be the expected value of the loss ℓ on samples generated from U_S . Then,

$$\mathbb{E}_{\mathbf{w} \sim U_S} [|\ell(\mathbf{w}) - \mu_{S,\ell}|] \leq \epsilon \quad (3.1)$$

- For any two points $w_1, w_2 \in S$, let $f(w_1, w_2) = w_1 + \tilde{\alpha}(w_2 - w_1)$, where $\tilde{\alpha} = \max \{ \alpha \mid w_1 + \alpha(w_2 - w_1) \in S \}$. Then,

$$\mathbb{E}_{\mathbf{w}_1, \mathbf{w}_2 \sim U_S, \nu \sim \mathcal{N}(0, (\delta^2/n) I_n)} [\ell(f(\mathbf{w}_1, \mathbf{w}_2) + \nu) - \mu_{S, \ell}] \geq 2\epsilon \quad (3.2)$$

- Let $\kappa(\mathbf{w}_1, \mathbf{w}_2, \nu) = f(\mathbf{w}_1, \mathbf{w}_2) + \frac{\nu}{\|f(\mathbf{w}_1, \mathbf{w}_2) - \mathbf{w}_1\|_2} (f(\mathbf{w}_1, \mathbf{w}_2) - \mathbf{w}_1)$. Then,

$$\mathbb{E}_{\mathbf{w}_1, \mathbf{w}_2 \sim U_S, \nu \sim \mathcal{N}(0, \delta^2)} [\ell(\kappa(\mathbf{w}_1, \mathbf{w}_2, |\nu|)) - \mu_{S, \ell}] \geq 2\epsilon \quad (3.3)$$

The first point requires that for most points on the basin, their loss should be close to the expected value of the loss in the basin. The other two requirements ensure that points in the vicinity of the basin exhibit a higher loss than the expected loss on the basin. Specifically, the second criterion achieves this by introducing Gaussian noise to the points in the basin, demanding that the loss be greater than the expected loss in the basin. Similarly, the third point requires that the loss should increase when moving away from the basin by extrapolating two points within it, along the subspaces spanned by this extrapolation (Neyshabur et al., 2020a).

3.2 Mode connectivity of neural networks

Understanding the loss landscape of deep neural networks has been the subject of many studies due to its close connections to optimization and generalization (Baldassi et al., 2020; Fort et al., 2019; Geiger et al., 2019; H. Li et al., 2017; Mei et al., 2018; Q. Nguyen et al., 2018). Empirical observations suggest that the loss landscape of deep networks has many minima (Draxler et al., 2018; Keskar et al., 2017; C. Zhang et al., 2017). One reason behind the abundance of minima is overparameterization. Overparameterized networks have enough capacity to present different functions that behave similarly on the training data but vastly different on other inputs (D. Li et al., 2018; C. Liu et al., 2020; Neyshabur et al., 2017; Q. Nguyen et al., 2018). Another contributing factor is the existence of scale and permutation invariances that allows the same function to be represented with many different parameter values of the same network and imposes a counter-intuitive geometry on the loss landscape (Brea et al., 2019a; Neyshabur et al., 2015).

Mode connectivity in neural networks refers to the phenomenon where different modes (*i.e.*, distinct minima) of the loss landscape are connected by low-loss pathways, implying that these modes are not isolated and can be connected through simple interpolation (Garipov et al., 2018). However, they discovered that these modes are *not* connected by a *linear* path and are connected by very simple curves, such as a polygonal chain with only one bend (Draxler et al., 2018; Freeman & Bruna, 2016; Garipov et al., 2018). The mode connectivity observation challenges the traditional belief that the

loss landscape of deep neural networks is highly non-convex, with isolated minima surrounded by high-loss regions. Mode connectivity has important implications for optimization algorithms, as it suggests that there may exist simpler ways to traverse the loss landscape and escape from local minima. **Linear Mode Connectivity.** Nagarajan and Kolter (2019) is probably the first example that considered *linear mode connectivity* (LMC) of trained MLPs from the same initialization. They note that the resulting networks are connected by linear paths of constant test error. Understanding LMC is highly motivated by several direct conceptual and practical implications from pruning and sparse training to distributed optimization and ensemble methods.

The relationship between LMC and pruning was established by Frankle et al. (2020) where they showed the correspondence between LMC and the well-known lottery ticket hypothesis (LTH) (Frankle & Carbin, 2019a). In short, LTH conjectures that neural networks contain sparse subnetworks that can be trained in isolation, from initialization, or early in training to achieve comparable test accuracy. Frankle et al. (2020) showed that solutions that are linearly connected with no barrier have the same lottery ticket. They further discuss how linear connectivity is associated with the stability of SGD. This view suggests that SGD solutions that are linearly connected with no barrier can be thought of as being in the same basin of the loss landscape and once SGD converges to a basin, it shows a stable behavior inside the basin¹. Because of the direct correspondence between LMC and LTH, any understanding of LMC has implications for LTH, the stability of SGD, and pruning techniques.

Linear mode connectivity has also direct implications for ensemble methods. Linear mode connectivity between solutions allows for making ensembles through weight averaging. As discussed in Chapter 1, in weight averaging the weights of multiple models are combined to create a single model with potentially improved performance. In the context of linear mode connectivity, it means that one can take the weights of two or more models located at different minimizers in the loss landscape and compute a linear combination of their weights, with the expectation that the resulting model will also have a low loss. The existence of linear mode connectivity implies that these low-loss pathways can be traversed by simply interpolating between the weights of different solutions. Consequently, it becomes possible to create ensembles by averaging the weights of multiple models, without the need for additional retraining or complex procedures.

¹This notion of the basin is consistent with the definition proposed by Neyshabur et al. (2020a) in Section 3.1

3.2.1 Loss barrier

Let $f_\theta(\cdot)$ be a function presented by a neural network with parameter vector θ that includes all parameters and $\mathcal{L}(\theta)$ be the any given loss (e.g., train or test error) of $f_\theta(\cdot)$. Let $\mathcal{E}_\alpha(\theta_1, \theta_2) = \mathcal{L}(\alpha\theta_1 + (1 - \alpha)\theta_2)$, for $\alpha \in [0, 1]$ be the loss of the network created by linearly interpolating between parameters of two networks $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$. The loss barrier $B(\theta_1, \theta_2)$ along the linear path between θ_1 and θ_2 is defined as the highest difference between the loss occurred when linearly connecting two points θ_1, θ_2 and linear interpolation of the loss values at each of them:

$$B(\theta_1, \theta_2) = \sup_{\alpha} [\mathcal{L}(\alpha\theta_1 + (1 - \alpha)\theta_2)] - [\alpha\mathcal{L}(\theta_1) + (1 - \alpha)\mathcal{L}(\theta_2)]. \quad (3.4)$$

We say that two networks θ_1 and θ_2 are **linear mode connected** if the barrier between them along a linear path is ≈ 0 (Frankle et al., 2020). It has been observed in the literature that any two minimizers of a deep network can be connected via a non-linear low-loss path (Draxler et al., 2018; Fort & Jastrzebski, 2019; Garipov et al., 2018). This work examines *linear* mode connectivity (LMC) between minima.

3.3 Empirical investigation of barriers

In this section, we investigate the impact of overparameterization on the shape of the loss landscape. We specifically look into barriers between different SGD solutions on all combinations of four architecture families (MLP (Rosenblatt, 1961), Shallow CNN (Neyshabur, 2020a), ResNet (K. He et al., 2015) and VGG (Simonyan & Zisserman, 2015)) and four datasets (MNIST (LeCun & Cortes, 2010), SVHN (Netzer et al., 2011), CIFAR-10 (Krizhevsky et al., 2009) and CIFAR-100 (Krizhevsky et al., 2009)). The main motivation to use Shallow CNN is to move from fully connected layers (MLP) to convolutions. The main difference between Shallow CNN and VGG16 is depth and the main difference between ResNet18 and VGG16 is the existence of residual connections.

We empirically investigate how different factors such as architecture family, width, depth, and task difficulty impact the barrier size². We refer to the training loss barrier as barrier.

Accuracy is more easily understood because it directly relates to the percentage of correct predictions. In addition, the loss value may vary

²In all plots the barrier is evaluated across 5 different random pairs (10 random SGD solutions). In our experiments, we observed that evaluating the barrier at $\alpha = \frac{1}{2}$ is a reasonable surrogate for taking the supremum over α (the difference is less than 10^{-4}). Therefore, to save computation, we report the barrier value at $\alpha = \frac{1}{2}$.

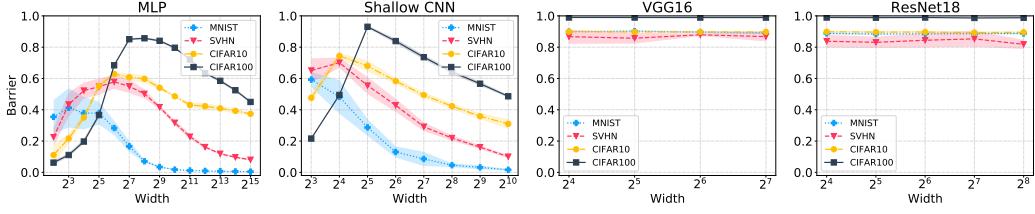


Figure 3.1: Effect of width on barrier size (Entezari et al., 2021). From left to right: one-layer MLP, two-layer Shallow CNN, VGG-16 and ResNet-18 architectures on MNIST, CIFAR-10, SVHN, CIFAR-100 datasets. For large width sizes, the barrier becomes small. This effect starts at lower width for simpler datasets such as MNIST and SVHN compared to CIFAR datasets. A closer look reveals a similar trend to that of double-descent phenomena. MLP architectures hit their peak at a lower width compared to CNNs and a decreasing trend starts earlier. For VGG and ResNet, the barrier size is saturated at a high value and does not change due to the effect of depth as discussed in Figure 3.3.

depending on the specific problem and the choice of the model. It may not be directly comparable between different models or even different problems. Consequently, in the following, we plot the accuracy barrier instead of the loss barrier similarly to Equation 3.4. using accuracy as a performance metric allows for better comparison of results across different models and problem domains. For loss barriers as well as train and test errors refer to Entezari et al. (2021).

3.3.1 Effect of width

We evaluate the impact of width on the barrier size in Figure 3.1. We note that for large values of width, the barrier becomes small. This effect starts at lower width for simpler datasets such as MNIST and SVHN compared to CIFAR datasets. A closer look reveals that the barrier increases with width up to a point and beyond that increasing width leads to lower barrier size. This effect is reminiscent of the double descent phenomena (Belkin et al., 2019; Nakkiran et al., 2019). Figure 3.2 depicts train and test error on the left while showing the barrier for MLP on the right. This figure indicates that in our experiments the barrier peak happens at the same size that is needed to fit the training data. Figure 3.1 shows that MLP architectures hit their barrier peak at a lower width compared to CNNs and a decreasing trend starts earlier. For ResNets, the barrier size is saturated at a high value and does not change. The barrier value for VGG architecture on different datasets is also saturated at a high value and does not change by increasing the width. Such similar behavior observed for both ResNets and VGG architectures is due to the effect of depth as discussed in the next paragraph.

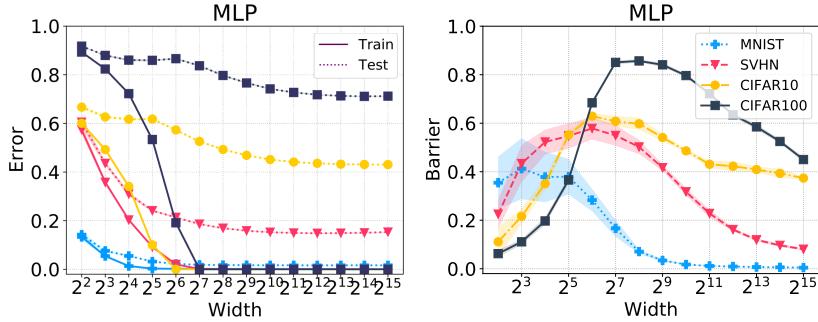


Figure 3.2: Deep double descent when looking at barrier (Entezari et al., 2021).

Left: Train and test error for MLP with one hidden layer across different widths and datasets. **Right:** Train barrier for the same network. The barrier increases with width up to a point and beyond that increasing width leads to lower barrier size. This effect is reminiscent of the double descent phenomena (Belkin et al., 2019; Nakkiran et al., 2019). Comparing both plots reveals that the barrier peak happens at the same size that is needed to fit the training data.

3.3.2 Effect of depth

We vary network depth in Figure 3.3 to evaluate its impact on the barrier between optimal solutions obtained from different initializations. For MLPs, we fix the layer width at 2^{10} while adding identical layers as shown along the x-axis. We observe a fast and significant barrier increase as more layers are added. For the VGG architecture family we observe significant barriers. This might be due to the effect of convolution or depth. In order to shed light on this observation, we use Shallow CNN (Neyshabur, 2020a) with only two convolutional layers. As can be seen in Figure 3.3 when Shallow CNN has two layers the barrier size is low while keeping the layer width fixed at 2^{10} and adding more layers increases the barrier size. For residual networks, we also consider three ResNet architectures with 18, 34, and 50 layers and observe the same barrier sizes as VGG for all these depth values. The main overall observation from depth experiments is that for both fully-connected and convolutional architectures, increasing depth increases the barrier size significantly so the effect of depth is not similar to width. This can also be attributed to the observations that deeper networks usually have less smooth landscapes (H. Li et al., 2017).

3.3.3 Effect of task difficulty and architecture choice

In Figure 3.4 we look into the impact of the task difficulty provided by the dataset choice (MNIST, SVHN, CIFAR-10, CIFAR-100, and ImageNet (Deng et al., 2009)) and the architecture type (one-layer MLP with 2^{10} neurons,

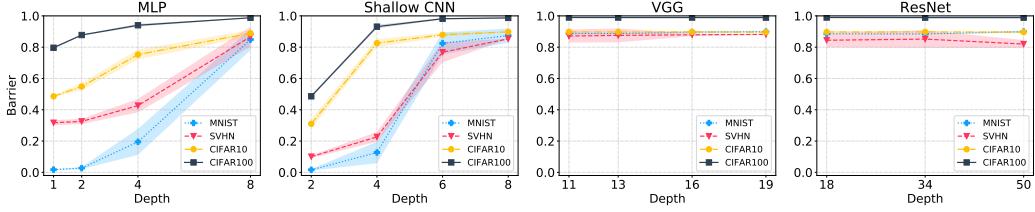


Figure 3.3: Effect of depth on barrier size (Entezari et al., 2021). From left to right MLP, Shallow CNN, VGG(11,13,16,19), and ResNet(18,34,50) architectures on MNIST, CIFAR-10, SVHN, CIFAR-100 datasets. For MLP and Shallow CNN, we fix the layer width at 2^{10} while adding identical layers as shown along the x-axis. Similar behavior is observed for fully connected and CNN families, *i.e.*, low barrier when the number of layers is low while we observe a fast and significant barrier increase as more layers are added. Increasing depth leads to higher barrier values until it saturates *e.g.*, VGG and ResNet

Shallow CNN with two convolutional layers and width of 2^{10} , VGG-16 with batch-normalization, ResNet18 and ResNet50). Each row in the left and middle figures in Figure 3.4 shows the effect of task difficulty, *e.g.*, fixing the task to SVHN and moving from MLP to Shallow CNN gives lower test error hence lower barrier size. Each column also represents the effect of architecture on a specific dataset, *e.g.*, fixing the architecture to Shallow CNN and moving from CIFAR10 to CIFAR100 presents an increase in test error, hence an increase in the barrier size. Although deep architectures like VGG16 and ResNet18 present low test error, the discussed effect of depth saturates their barrier at a high level. Figure 3.4 right, aggregates the correlation between test error and the size of the barrier. For MLP and Shallow CNN, we observe a high positive correlation between test error and barrier size across different datasets. Deeper networks (VGGs, ResNets) form a cluster in the top-left, with low test error and high barrier size.

3.4 Role of invariance in loss barriers

Understanding the loss landscape of deep networks has proven to be very challenging. One of the main challenges in studying the loss landscape without taking the optimization algorithm into account is that there exist many minima with different generalization properties. Most of such minima are not reachable by SGD and we only know about their existence through artificially-made optimization algorithms and training regimes (Neyshabur et al., 2017). To circumvent this issue, we focus on parts of the landscape that are reachable by SGD. Given a dataset and an architecture, one could define a probability distribution over all solutions reachable by SGD and focus on the subset in which SGD is more likely to converge.

3 Loss Landscape and Generalization

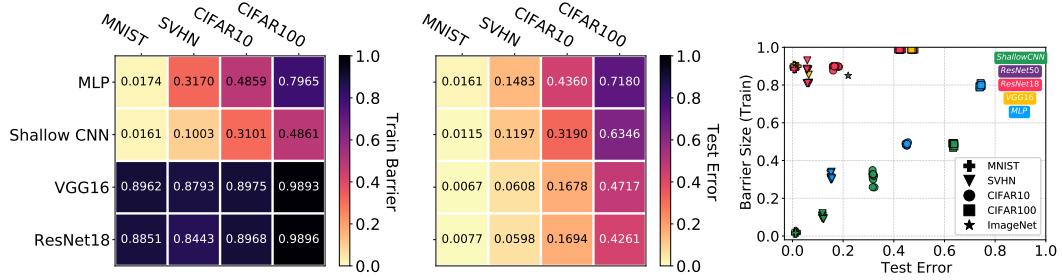


Figure 3.4: **Effect of architecture choice and task difficulty on barrier size (Entezari et al., 2021).** **Left:** Barrier for different architectures and task difficulty. **Middle:** Achieved test error. Each row in the left and middle figures shows the effect of task difficulty while each column represents the effect of architecture on a specific dataset. **Right:** Lower test error results in a lower barrier for shallow networks. A pair of (architecture, task) has a lower barrier if the test error is lower. Therefore, any changes in the architecture or the task that improves the test error also improve the loss barrier. The effect of depth is stronger than (architecture, task) which leads to high barrier values for ResNets on MNIST, SVHN, CIFAR10, CIFAR100, and ImageNet.

We say that a network is *invariant* with respect to a transformation if and only if the network resulting from the transformation represents the same function as the original network. There are two well-known invariances: one is the unit-rescaling due to positive homogeneity of ReLU activations (Neyshabur et al., 2015) and the other is the permutation of hidden units. Unit-rescaling has been well-studied and empirical evidence suggests that implicit bias of SGD would make the solution converge to a stage where the weights are more balanced (Neyshabur et al., 2015; X. Wu et al., 2019). Since we are interested in the loss landscape through the lens of SGD and SGD is much more likely to converge to a particular rescaling, consideration of this type of invariance does not seem useful. However, in the case of permutations, all permutations are equally likely for SGD and therefore, it is important to understand their role in the geometric properties of the landscape and its basins of attraction. Further investigation of other optimization algorithms such as ADAM is left for future works.

3.4.1 Permutation invariance

We consider invariances that are in form of permutations of hidden units in each layer of the network, *i.e.*, each layer i with parameters W_i is replaced with $P_i W_i P_{i-1}$ where P_i is a permutation matrix and $P_l = P_0$ is the identity matrix. We use \mathcal{P} to refer to the set of valid permutations for a neural network and use π to refer to a valid permutation.

Related work on permutation invariance. Permutation symmetry of neurons in every layer results in multiple equivalent minima connected via saddle points. Few studies investigate the role of these symmetries in the context of the connectivity of different basins.

Given a network with L layers of minimal widths r_1^*, \dots, r_{L-1}^* that reaches zero-loss minima at $r_1!, \dots, r_{L-1}!$ isolated points (permutations of one another), Şimşek et al. (2021) showed that adding one extra neuron to each layer is sufficient to connect all these previously discrete minima into a single manifold. Fukumizu and Amari (2000) prove that a point corresponding to the global minimum of a smaller model can be a local minimum or a saddle point of the larger model. Brea et al. (2019b) find smooth paths between equivalent global minima that lead through a permutation point, *i.e.*, where the input and output weight vectors of two neurons in the same hidden layer interchange. They describe a method to permute all neuron indices in the same layer at the same cost. Tatro et al. (2020) showed that aligning the neurons in two different neural networks makes it easier to find second-order curves between them in the loss landscape where barriers are absent.

3.5 Permutation invariance conjecture

As mentioned above, SGD’s implicit regularization balances weight norms and, therefore, scale invariance does not seem to play an important role in understanding the symmetries of solutions found by SGD. Consequently, here we focus on permutation invariance and conjecture that taking it into account allows us to have a much simpler view of SGD solutions. We first state our conjecture informally:

Most SGD solutions belong to a set \mathcal{S} whose elements can be permuted in such a way that there is no barrier on the linear interpolation between any two permuted elements in \mathcal{S} .

The above conjecture suggests that most SGD solutions end up in the same basin in the loss landscape after proper permutation. We acknowledge that the above conjecture is bold. Nonetheless, we argue that coming up with strong conjectures and attempting to disprove them is an effective method for scientific progress. Note, our conjecture also has great practical implications for model ensembling and parallelism since one can average models that are in the same basin in the loss landscape. The conjecture can be formalized as follows:

Conjecture 1. *Let $f(\theta)$ be the function representing a feedforward network with parameters $\theta \in \mathbb{R}^k$, \mathcal{P} be the set of all valid permutations for the network, $P : \mathbb{R}^k \times \mathcal{P} \rightarrow \mathbb{R}^k$ be the function that applies a given permutation to parameters and*

returns the permuted version, and $B(\cdot, \cdot)$ be the function that returns barrier value between two solutions as defined in Equation 3.4. Then, there exists a width $h > 0$ such that for any network $f(\theta)$ of width at least h the following holds: There exist a set of solutions $\mathcal{S} \subseteq \mathbb{R}^k$ and a function $Q : \mathcal{S} \rightarrow \mathcal{P}$ such for any $\theta_1, \theta_2 \in \mathcal{S}$, $B(P(\theta_1, Q(\theta_1)), \theta_2) \approx 0$ and with high probability over an SGD solution θ , we have $\theta \in \mathcal{S}$.

3.6 A theoretical result

In this section, we provide elementary theoretical results in support of our conjecture. Although the theoretical result is provided for a very limited setting, we believe it helps us understand the mechanism that could give rise to our conjecture. Below, we theoretically show that Conjecture 1 holds for a fully-connected network with a single hidden layer at initialization. For the proof refer to (Entezari et al., 2021).

Theorem 1. Let $f_{\mathbf{v}, \mathbf{U}}(\mathbf{x}) = \mathbf{v}^\top \sigma(\mathbf{U}\mathbf{x})$ be a fully-connected network with h hidden units where $\sigma(\cdot)$ is ReLU activation, $\mathbf{v} \in \mathbb{R}^h$ and $\mathbf{U} \in \mathbb{R}^{h \times d}$ are the parameters and $\mathbf{x} \in \mathbb{R}^d$ is the input. If each element of \mathbf{U} and \mathbf{U}' is sampled uniformly from $[-1/\sqrt{d}, 1/\sqrt{d}]$ and each element of \mathbf{v} and \mathbf{v}' is sampled uniformly from $[-1/\sqrt{h}, 1/\sqrt{h}]$, then for any $\mathbf{x} \in \mathbb{R}^d$ such that $\|\mathbf{x}\|_2 = \sqrt{d}$, with probability $1 - \delta$ over $\mathbf{U}, \mathbf{U}', \mathbf{v}, \mathbf{v}'$, there exist a permutation such that

$$\left| f_{\alpha\mathbf{v} + (1-\alpha)\mathbf{v}'', \alpha\mathbf{U} + (1-\alpha)\mathbf{U}''}(\mathbf{x}) - \alpha f_{\mathbf{v}, \mathbf{U}}(\mathbf{x}) - (1-\alpha) f_{\mathbf{v}', \mathbf{U}'}(\mathbf{x}) \right| = \tilde{O}(h^{-\frac{1}{2d+4}})$$

where \mathbf{v}'' and \mathbf{U}'' are permuted versions of \mathbf{v}' and \mathbf{U}' .

Theorem 1 states that for wide enough fully-connected networks with a single hidden layer, one can find a permutation that leads to having no barrier at random initialization. Although, our proof only covers random initialization, we believe with a more involved proof, it might be possible to extend it to NTK regime (Jacot et al., 2018). The NTK (Neural Tangent Kernel) regime refers to a phase in the training process of deep neural networks where the network behaves linearly, particularly in the context of infinitely wide networks.

3.7 Direct empirical evaluation

Another possible approach is to use brute-force (BF) search mechanism and find the function Q for elements of \mathcal{S} . The factorial growth of the number of permutations with the size of hidden units in each layer hinders exhaustive search for a winning permutation π to linear mode connect $P(\theta_1, \pi)$ and θ_2 .

Algorithm 2 Simulated Annealing (SA) for Permutation Search

```
1: procedure SA( $\{\theta_i\}, i = 1..n, n \geq 2$ )            $\triangleright$  Goal: minimize the barrier  
   between  $n$  solutions  
2:    $\pi_i = \pi_0, \forall i = 1..n$   
3:   for  $k = 0; k < k_{\max}; k++$  do  
4:      $T \leftarrow \text{temperature}\left(\frac{k+1}{k_{\max}}\right)$   
5:     Pick random candidate permutations  $\{\hat{\pi}_i\}, \forall i = 1..n$   
6:     if  $\Psi(P(\theta_i, \hat{\pi}_i)) < \Psi(P(\theta_i, \pi_i))$  then  $\triangleright \Psi$ : barrier objective function  
7:        $\pi_i \leftarrow \hat{\pi}_i$   
return  $\{\pi_i\}$ 
```

Even for MLPs with just one hidden layer brute-force works in reasonable time up to 2^4 neurons only, forcing the search to examine $2^4! \approx 2 \cdot 10^{13}$ permuted networks. This number grows as 10^{35160} for VGG-16 and 10^{55109} for ResNet-50. For comparison, the number of atoms in the observable universe is about 10^{82} (Ainsworth et al., 2022). For small networks, one can use BF to find permutations between different models. However, small size networks are not the focus of this paper, and Conjecture 1 specifically mentions that.

3.8 Search algorithms for finding a winning permutation

Given the size of the search space, using a more advanced search algorithm can be useful. There exist different neuron alignment algorithms in the literature. The issue with this approach is that since it relies on the strength of a search algorithm if the search algorithm fails in finding the permutation, one cannot be sure about the source of failure being the search algorithm or the nonexistence of a permutation that leads to no barrier. In the following, we first introduce three methods of neuron alignment in two networks and then compare their performance in our setting.

3.8.1 Simulated Annealing (SA)

This method is often used in the literature, *e.g.*, by Zhan et al. (2016), to find a solution for a combinatorial search problem. SA performance is, however, highly sensitive to the parameter choices, *e.g.*, the minimum and maximum temperatures, the cooling schedule, and the number of optimization steps. The pseudocode of SA is shown in Algorithm 2. SA takes a set of solutions $\{\theta_i\}, i = 1..n, n \geq 2$ as input (we use $n = 5$) and searches for a set of permutations $\{\pi_i\}$ that reduce the barriers between all

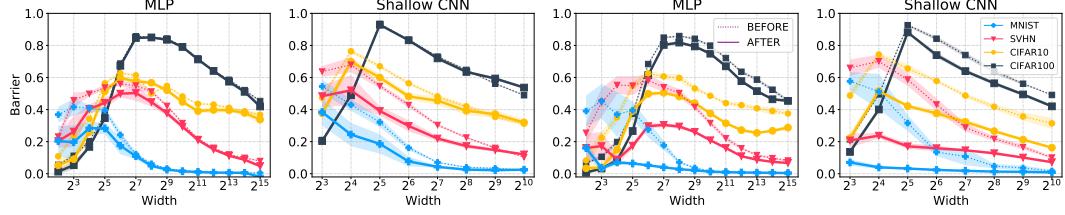


Figure 3.5: Performance of Simulated Annealing (SA) (Entezari et al., 2021). Two Left: SA₂ where we average the weights of permuted models first and ψ is defined as the train error of the resulting average model. Two Right: Search space is reduced *i.e.*, we take two SGD solutions θ_1 and θ_2 , permute θ_1 and report the barrier between permuted θ_1 and θ_2 as found by SA with $n = 2$. When search space is reduced, SA is able to find better permutations.

permuted $\binom{n}{2}$ solution pairs. To find the best $\{\pi_i\}$, in each step of SA the current candidate permutations $\{\hat{\pi}_i\}, i = 1..n$ are evaluated to minimize the objective function Ψ . We use two versions of simulated annealing that vary in their definition of Ψ to evaluate the conjecture.

Simulated Annealing 1 (SA₁) In the first version, Ψ is defined as the average pairwise barrier between candidate permutations $B(P(\theta_i, \pi_i), P(\theta_j, \pi_j))$, $i \neq j$.

Simulated Annealing 2 (SA₂) In the second version SA₂, we average the weights of permuted models $P(\theta_i, \pi_i)$ first and define Ψ as the train error of the resulting average model. The simplest form of SA₂ happens if $n = 2$. The rationale behind these two versions is that if the solutions reside in one basin, there is no barrier between them. Therefore averaging solutions in one basin yields another solution inside their convex hull. Our empirical results suggest that SA₁ and SA₂ yield very similar performance. However, SA₂ is significantly less computationally expensive, which makes it more suitable for exploring larger models. In the following, we present the results obtained with SA₂ only and refer to this version as SA. For more details on SA implementation see Entezari et al. (2021).

The left two plots in Figure 3.5 show that SA₂ is not able to find permutations that improve pair-wise barrier significantly. We know that SA does not guarantee to find a solution and is known to lose its effectiveness on Travelling Salesman Problem benchmarks beyond 1'000 cities (Zhan et al., 2016). The effectiveness of SA is also reduced here as we can only evaluate the cost of the full route (divide and conquer is not possible). One way to increase this effectiveness is to reduce the search space which we will discuss next.

Search space reduction. In order to reduce the search space, here we only take two SGD solutions θ_1 and θ_2 , permute θ_1 and report the barrier between permuted θ_1 and θ_2 as found by SA with $n = 2$. The right two plots in Figure 3.5 show this intervention helps SA to find better permutations. In particular, the barrier improves significantly for MNIST and SVHN datasets for both MLP and Shallow CNN across different widths. However, similar to Section 2, we did not observe significant improvements when increasing depth (see (Entezari et al., 2021) for evaluation of SA on deeper networks).

3.8.2 Functional Difference (FD)

X. He et al. (2018) proposed an algorithm that merges two models optimized for two different tasks into one network for cross-model compression. The compressed network does both tasks without losing too much accuracy on each task. Their algorithm is based on layer-wise neuron sharing, which uses a function of weights and post-activations to find which neurons could be zipped together. They define a similarity measure, referred to as functional difference (FD), of two neurons as follows:

$$\delta_{n_A, n_B} = \frac{1}{2} (w_{l,i}^A - w_{l,i}^B) \cdot ((H_{l,i}^A)^{-1} + (H_{l,i}^B)^{-1})^{-1} \cdot (w_{l,i}^A - w_{l,i}^B), \quad (3.5)$$

where $w_{l,i}^A$ and $w_{l,i}^B$ are the weights of neurons i in layer l of two networks A and B ; $H_{l,i}^A$ and $H_{l,i}^B$ are layer-wise Hessian matrices. We use FD to match neurons between two randomly initialized trained networks. Similarly to the original paper (X. He et al., 2018) we use post-activation to approximate Hessian matrices. Calculating the difference between each pair of neurons gives a $m \times m$ matrix (m width of the network). In the second step, neurons with a minimum distance are matched together in a greedy way, *i.e.*, if $n_{i,A}$ and $n_{j,B}$ have a minimum distance, row i and column j are removed from the distance matrix.

We change the second step of FD and use the Hungarian algorithm (Kuhn, 1955a) to match two networks. We use the Python Scipy library (Virtanen et al., 2020) and employ the cost matrix calculated in the first step of FD. Figure 3.6 shows that FD outperforms simulated annealing in reducing the original barrier. Hungarian implementation also improves FD performance.

3.8.3 Optimal Transport Fusion (OPT)

Singh and Jaggi (2020) suggest minimizing the transportation cost of neurons present in the layers of individual models and measured by the similarity of activations or incoming weights. The resulting layer-wise averaging scheme can be interpreted as computing the Wasserstein barycenter, *i.e.*, averaging

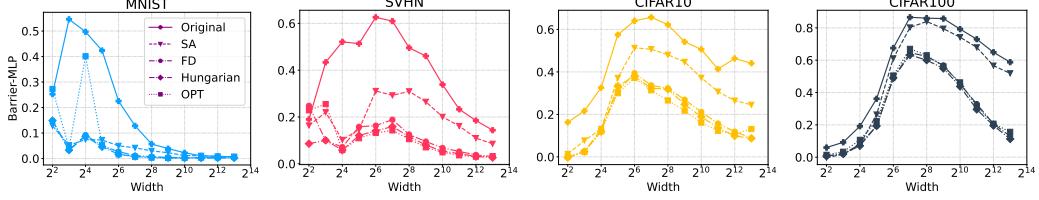


Figure 3.6: **Performance of neuron alignment algorithms: Simulated Annealing (SA), Functional Difference (FD) and Optimal Transport Fusion (OPT).** Comparison for MLPs on different datasets (columns). OPT, closely followed by FD, shows the best performance in finding a winning permutation to reduce the barrier. Hungarian implementation also shows slightly better performance than FD.

in the Wasserstein space of the probability measures defined at the corresponding layers of the parent models (Aguech & Carlier, 2011). Formally, let μ and ν be probability measures and let $\Pi(\mu, \nu)$ denote the set of joint probability measures with marginals μ and ν . The Wasserstein- p metric is defined as:

$$W_p(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu, \nu)} \mathbb{E}_{(x,y) \sim \pi} (\|x - y\|^p) \right)^{1/p}, \quad (3.6)$$

where the optimization searches for an optimal transport plan among all transport plans π between two distributions with the cost being the Euclidean p -norm.

3.8.4 Correlation of activations

Y. Li et al. (2015) propose to maximize the sum of correlations between the activations of paired neurons across a batch of training data. That is, if we let $X_{l,i}^{(0)}$ and $X_{l,i}^{(1)}$ be random variables corresponding to the activations of the i -th hidden units of the l -th layer (across a batch of training data), then Y. Li et al. (2015) proposes to optimize the permutation P_l to minimize following cost:

$$\sum_i \text{corr}(X_{l,i}^{(1)}, X_{l,P_l(i)}^{(2)}). \quad (3.7)$$

This amounts to a linear sum assignment problem corresponding to the matrix of correlations between pairs of hidden units in the two networks; which can be solved via the Hungarian algorithm (Kuhn, 1955b). Tatro et al. (2020) also perform alignment based on minimizing Equation 3.7, in order to reduce the barrier to non-linear interpolation. Furthermore, they show

3 Loss Landscape and Generalization

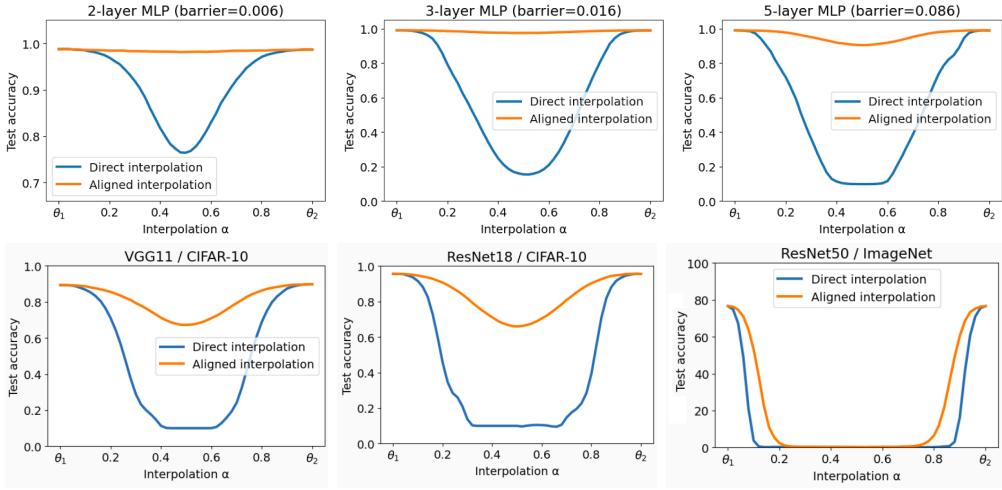


Figure 3.7: Performance of neuron alignment using correlation of activations (Jordan et al., 2022). Direct interpolation refers to naive interpolation between two trained models without finding the permutation. Aligned interpolation is the utilization of correlation for activations as the cost function (Y. Li et al., 2015), followed by the implementation of the Hungarian algorithm (Kuhn, 1955b). Aligned interpolation can significantly enhance the performance of search algorithms aimed at identifying a winning permutation. Nevertheless, as the network’s depth increases, the efficiency of this approach declines rapidly, with ResNet-50 for ImageNet (right bottom) showing scarcely any improvement.

using a proximal alternating minimization scheme that alignments found using this method are nearly optimal for their purposes.

We note that for networks with residual connections, care must be taken to restrict the set of permutations such that the function represented by the network does not change. In particular, the same permutation of hidden units must be applied to all layers which feed into a single residual stream.

Figure 3.7 depicts the utilization of correlation of activations (Y. Li et al., 2015), followed by the implementation of the Hungarian algorithm (Kuhn, 1955b). While other search algorithms show poor performance, even for one-layer MLP (Figure 3.6), this method makes zero barriers for 2 and 3-layer MLPs. However, as the network gets deeper, the efficiency of this approach declines rapidly, with ResNet-50 for ImageNet showing scarcely any improvement.

3.9 Identifying the problem: Variance Collapse

What causes a rapid drop-off in the performance of interpolated networks which are deeper than a few layers? To answer this question, we investigate

3 Loss Landscape and Generalization

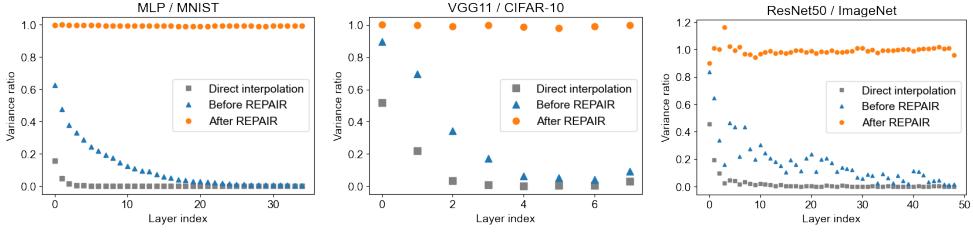


Figure 3.8: Variance collapse phenomenon in averaged networks (Jordan et al., 2022). We find that the hidden units of weight-space averaged neural networks suffer from variance collapse: as we progress through the network, the variance of neuron activations reduces, with neurons in deeper layers becoming nearly constant while varying the input data. *Before REPAIR* refers to networks that are interpolated from endpoint networks whose hidden units have been aligned (Ainsworth et al., 2022; Y. Li et al., 2015; Singh & Jaggi, 2020), before our correction method REPAIR is applied on top of this baseline in order to restore the internal statistics of averaged networks back to the level of the endpoint networks.

the internal behavior of such networks, focusing on the statistics of hidden units (Figure 3.8). We find that for deep MLPs, interpolated from a pair of aligned endpoint networks which both have high accuracy on the MNIST test-set, hidden units undergo a *variance collapse*. That is, the variance of their activations progressively decays as we move deeper into the network, with the activations of later layers becoming nearly constant. For each layer, we quantify this decay as follows. First, we measure the variance of the activations of each neuron across a batch of training data. We then take the sum of this variance across each neuron in the layer. Finally, if we let this sum be denoted v_α, v_1, v_2 for the interpolated and two endpoint networks, respectively, then we report the ratio $\frac{v_\alpha}{(v_1+v_2)/2}$. We compute this ratio for each layer in the network, giving a sequence of values which we report in Figure 3.8 (left). For the set of variances of each neuron in a single layer of an interpolated ResNet18, see Figure 3.9 (left).

We observe that variance decays to nearly zero by the final layer of an interpolated 35-layer MLP, indicating that the activations in these last layers have become nearly constant. This effect seems to be further exacerbated when directly interpolating between unaligned networks. We repeat this experiment for VGG (Simonyan & Zisserman, 2015) and ResNet-50 architectures, trained on CIFAR-10 and ImageNet respectively, and find that variance by the final layers decays by more than $10\times$ (Figure 3.8 (middle) and Figure 3.8 (right)). This is a problem: if these networks have nearly constant activations in their final layers, then they will no longer even be able to differentiate between inputs.

3.9.1 Why does the variance collapse phenomenon occur?

We argue that this phenomenon can be understood through the following statistical calculation. Consider a hidden unit or channel in the first layer of the interpolated network. Such a unit will be functionally equivalent to the linear interpolation between the respective units in the endpoint networks. That is, if we represent the unit's pre-activation by X_α in the interpolated network, and X_1, X_2 in the two endpoint networks (as random variables over the input data distribution), then the equality $X_\alpha = (1 - \alpha)X_1 + \alpha X_2$ holds. We will argue that the variance of X_α is typically reduced as compared to that of X_1 or X_2 .

If the two endpoint networks are perfectly aligned and have learned the same features, then we should have $\text{corr}(X_1, X_2) = 1$. But in practice, it is more typical for pairs of aligned units (whose alignment minimizes the cost function given by Equation 3.7) to have a correlation of $\text{corr}(X_1, X_2) \approx 0.4$. When considering the midpoint interpolated network ($\alpha = 0.5$), the variance of X_α is given by

$$\begin{aligned}\text{Var}(X_\alpha) &= \text{Var}\left(\frac{X_1 + X_2}{2}\right) \\ &= \frac{\text{Var}(X_1) + \text{Var}(X_2) + 2\text{Cov}(X_1, X_2)}{4} \\ &= \frac{\text{std}^2(X_1) + \text{std}^2(X_2) + 2 \cdot \text{corr}(X_1, X_2) \cdot \text{std}(X_1)\text{std}(X_2)}{4} \\ &= \left(\frac{\text{std}(X_1) + \text{std}(X_2)}{2}\right)^2 - \frac{(1 - \text{corr}(X_1, X_2))}{2} \text{std}(X_1)\text{std}(X_2).\end{aligned}$$

We typically have $\text{std}(X_1) \approx \text{std}(X_2)$, so that this simplifies to $\text{Var}(X_\alpha) = (0.5 + 0.5 \cdot \text{corr}(X_1, X_2)) \cdot \text{Var}(X_1)$. With our typical value of $\text{corr}(X_1, X_2) \approx 0.4$ for aligned networks, this yields $\text{Var}(X_\alpha) = 0.7 \cdot \text{Var}(X_1)$: a 30% reduction compared to the endpoint networks. This analysis cannot be rigorously extended to deeper layers of the interpolated network, but intuitively we expect this decay to compound with depth. This intuition matches our experiments, where we see that variance collapse becomes worse as we progress through the layers of MLP, VGG, and ResNet-50 networks (Figure 3.8).

3.10 REPAIR

We propose two methods for addressing variance collapse. Both aim to correct the statistics of hidden units in the interpolated network. We call these methods REPAIR (REnormalizing Permuted Activations for Interpolation Repair).

Given an interpolated network $\theta_\alpha = (1 - \alpha) \cdot \theta_1 + \alpha \cdot \theta_2$ for some $0 < \alpha < 1$ (with aligned endpoint networks θ_1, θ_2), we select the set of hidden units or channels whose statistics we aim to correct. For example, for VGG networks we correct the pre-activations of every convolutional layer. For ResNets, we correct both these convolutional pre-activations and the outputs of each residual block.

Our goal will be to compute a set of affine (rescale-and-shift) coefficients for every selected channel, such that the statistics of all selected channels are corrected. Let us consider a particular channel, *e.g.*, the 45th convolutional channel of the 8th layer in an interpolated ResNet18. Similar to the analysis in the last section, let X_1 and X_2 be the values of the channel in the two endpoint networks, viewed as random variables over the input training data, and let X_α be the same channel in the interpolated network. Then we want the following two conditions to hold:

$$\mathbb{E}[X_\alpha] = (1 - \alpha) \cdot \mathbb{E}[X_1] + \alpha \cdot \mathbb{E}[X_2], \quad (3.8)$$

$$\text{std}(X_\alpha) = (1 - \alpha) \cdot \text{std}(X_1) + \alpha \cdot \text{std}(X_2). \quad (3.9)$$

Before any correction, we typically have $\text{std}(X_\alpha) \ll \min(\text{std}(X_1), \text{std}(X_2))$ due to variance collapse. In the following, we present two algorithms to compute the appropriate sets of affine coefficients for each selected channel, in order to induce these conditions. Both algorithms depend upon the computation of the statistics $\mathbb{E}[X_1]$, $\mathbb{E}[X_2]$, $\text{std}(X_1)$, $\text{std}(X_2)$ for each selected channel in the endpoint networks.

3.10.1 Closed-form approximate variant

We first present an efficient, approximate algorithm that computes the desired affine coefficients without using forward-passes in the interpolated network. Consider a hidden unit in the first layer of the interpolated network. As before, let X_α represent the unit in the interpolated network, and X_1, X_2 the same unit in the two endpoint networks, respectively. Condition (3.8) will already be satisfied for this unit by virtue of the equation $X_\alpha = (1 - \alpha) \cdot X_1 + \alpha \cdot X_2$. Given the values $\text{Var}(X_1)$, $\text{Var}(X_2)$, and $\text{Cov}(X_1, X_2)$, it is possible to compute the variance of X_α exactly according to the formula

$$\text{Var}(X_\alpha) = (1 - \alpha)^2 \text{Var}(X_1) + \alpha^2 \text{Var}(X_2) + 2\alpha(1 - \alpha)\text{Cov}(X_1, X_2). \quad (3.10)$$

Therefore, to satisfy condition (3.9) for this unit, the rescaling coefficient β must be

$$\beta = \frac{(1 - \alpha) \cdot \text{std}(X_1) + \alpha \cdot \text{std}(X_2)}{\sqrt{(1 - \alpha)^2 \text{Var}(X_1) + \alpha^2 \text{Var}(X_2) + 2\alpha(1 - \alpha)\text{Cov}(X_1, X_2)}},$$

which is simply the desired standard deviation divided by the standard deviation of X_α . For each unit in the first layer, this factor is exactly correct

3 Loss Landscape and Generalization

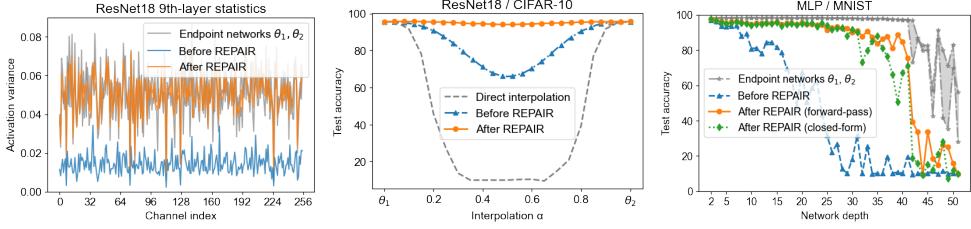


Figure 3.9: REPAIR restores the internal statistics of averaged neural networks (Jordan et al., 2022). **Left:** We visualize the statistics of different channels in 9th layer of an interpolated ResNet18 on CIFAR-10. The uncorrected network undergoes variance collapse, whereas REPAIR restores the internal statistics of the network to be similar to the endpoint networks. **Middle:** After applying REPAIR, internal statistics are restored, and the barrier is reduced to 1.5% for CIFAR-10. **Right:** Permuted interpolation without a statistical correction (before REPAIR) only performs well when limited to MLPs of a few layers (Entezari et al., 2021). REPAIR enables high-performance weight-space averaging between much deeper aligned MLPs.

in order to obtain the desired statistics. For deeper layers, this factor is an approximation that we empirically test.

In Figure 3.9 (right), we apply this rescaling to every hidden unit of MLPs of depth between 2 and 50 hidden layers, which are linearly interpolated ($\alpha = 0.5$) between aligned endpoint networks trained on MNIST. We find that this rescaling significantly improves the performance of such interpolated networks. In particular, we obtain interpolated checkpoints of up to 27 layers that achieve over 90% accuracy, whereas, without a correction, we hit this limit after only 6 layers.

We note that this correction requires forward passes in the endpoint networks in order to compute the values of $\text{Var}(X_1)$, $\text{Var}(X_2)$, and $\text{Cov}(X_1, X_2)$ for each hidden unit. Once these values have been computed, correction coefficients to interpolated networks across the arbitrary choice of interpolation coefficient α can be generated without requiring any further forward passes.

3.10.2 Forward-pass exact variant

The rescaling coefficients generated by the above algorithm are approximate, only being guaranteed to induce the desired conditions (3.8), (3.9) for hidden units or channels in the first layer of an interpolated network. We find that it is effective for the case of deep MLPs, but in our experiments, it was insufficient to significantly reduce the interpolation barrier for more challenging cases like ResNet-50 trained on ImageNet. We now propose an exact algorithm, which uses forward passes in the interpolated networks in order

to generate affine (rescale-and-shift) parameters for every channel in the network that we aim to correct. This method outperforms the approximate variant, especially for challenging cases.

The exact method proceeds as follows. For each module in the interpolated network whose outputs we have identified as targets for statistical correction, we apply a wrapper (for the PyTorch pseudocode see Jordan et al. (2022)). This wrapper adds a Batch Normalization layer after the wrapped module which is initially set to “train” mode. As a recall, batch normalization consists of two steps: normalization and affine transformation. The normalization step normalizes the input data to a layer by adjusting and scaling it based on the mean and variance of the data within a mini-batch. This helps to mitigate the effects of internal covariate shift and makes it easier for the subsequent layers to learn the representations of the data. The affine transformation step involves applying a linear transformation to the normalized data. This step allows the network to learn the appropriate scaling and shifting of the normalized data for each layer (Davis & Frank, 2022).

Each added BatchNorm layer also contains affine (*i.e.*, per-channel rescale-and-shift) parameters. For a given channel incoming to the BatchNorm layer, we set the respective affine weight to $(1 - \alpha) \text{std}(X_1) + \alpha \text{std}(X_2)$ and the bias to $(1 - \alpha) \mathbb{E}[X_1] + \alpha \mathbb{E}[X_2]$, where X_1 and X_2 are the respective channels in the endpoint networks as in conditions (3.8), (3.9).

With the added BatchNorm layers in training mode, these affine parameters will exactly induce our statistical conditions with respect to batches of input data. The reason for this is that during execution, the added BatchNorm layers first renormalize their inputs to have zero mean and unit variance per channel, and then apply our given affine transformation which sets the statistics of the output to be that of conditions (3.8), (3.9). Next, we pass a set of training data through the network (~5,000 examples suffices) so that the running mean and variance parameters of our added BatchNorm layers will be accurately estimated. During this pass, any BatchNorm layers which already existed in the original network are kept frozen. Finally, we set the added BatchNorm layers to evaluation mode, so that they behave as affine layers which do not recompute statistics. At this point, the resulting network is functionally equivalent to one in which the weights of our selected set of channels have been rescaled and biases shifted. If we wish to generate a new parameter vector θ'_α which is compatible with the original network architecture (*i.e.*, lacks these added BatchNorm layers), then we can perform BatchNorm layer fusion (Markuš, 2018) in which appropriate rescaling and bias-shifting values are computed from each added BatchNorm layer and then applied to the preceding convolutional filters.

The networks resulting from this process have their internal statistics corrected so that all selected channels satisfy conditions (3.8), (3.9). In Figure 3.9 (left) we observe that REPAIR has resolved variance collapse.

In Figure 3.9 (middle) we apply REPAIR to networks where the weights are linearly interpolated between a pair of ResNet18s whose hidden units have been brought into alignment. Before the correction, networks near the midpoint have a reduced accuracy of 66.0% on the CIFAR-10 test set, while the endpoints accuracy is 95.5%. After correction, all checkpoints along the linear path have significantly boosted accuracy, with the midpoint performing at 94.1%. In comparison, (Singh & Jaggi, 2020) report a linear midpoint accuracy of 77.0% using strong optimal-transport based alignment methods to improve over the baseline.

3.11 Effectiveness of REPAIR

In this section, we extend the results on the effectiveness of REPAIR across a wider range of scenarios. We apply REPAIR to (interpolations between aligned) ResNets trained on ImageNet. (Section 3.11.1) We also perform a replication using REPAIR of an experiment from (Ainsworth et al., 2022), in which a pair of models trained on disjoint subsets of data is constructively merged. (Section 3.11.2). For more experiments on studying the relationship between width and depth with barrier size, as well as a comparison between different normalization techniques *e.g.*, BatchNorm (Ioffe & Szegedy, 2015), LayerNorm (Ba et al., 2016), and Fixup (H. Zhang et al., 2019) refer to Jordan et al. (2022).

3.11.1 REPAIR for ImageNet

Next, we explore the impact of REPAIR on the barriers to interpolation between standard ResNet models trained from scratch on ImageNet (Deng et al., 2009). We test ResNet18, ResNet-50, and a double-width variant of ResNet-50 in Figure 3.10 (right). Without REPAIR, the interpolated midpoints between each aligned pair of networks perform at below 1% accuracy on the ImageNet validation set. After REPAIR, the midpoint ResNet18 improves to 41.1%, ResNet-50 to 56.5%, and double-width ResNet-50 to 64.2%.

We find in Figure 3.10 (left) that it is important to apply REPAIR not just to all convolutional layer outputs, but also to the outputs of every residual block. For the case of ResNet-50, using REPAIR with these extra channels boosted the performance of the midpoint from 53.2% to 56.5%, which is a 14% reduction in the size of the barrier (from 23.4% to 20.1%). We note that for architectures that contain BatchNorm after every convolutional layer, applying REPAIR only to convolutional layer outputs is mathematically equivalent to resetting the BatchNorm statistics of the network. Performing such a reset on averaged networks goes back to (Izmailov et al., 2018), but as far as we are aware, has not been applied to interpolation between

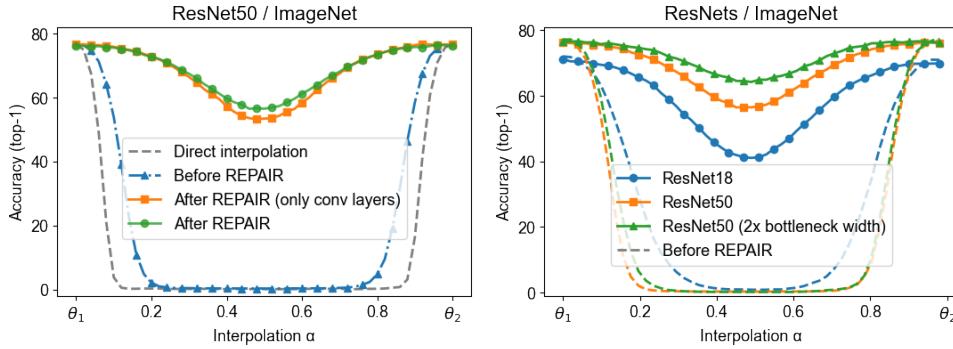


Figure 3.10: **REPAIR significantly reduces barrier between ResNet-50s trained on ImageNet (Jordan et al., 2022).** **Left:** Without REPAIR, interpolations between two aligned, independently trained ResNet-50s attain less than 1% test accuracy on ImageNet. After applying REPAIR to convolutional layer outputs, the midpoint is boosted to 53.2%. Using full REPAIR, which is also applied to the outputs of residual blocks, this is further boosted to 56.5%. **Right:** Larger and wider ResNet architectures have a smaller barrier. Dashed lines indicate the baseline of interpolation between aligned networks (Ainsworth et al., 2022; Singh & Jaggi, 2020), and solid lines refer to REPAIR on top of the baseline.

independently trained networks until now.

In general, the barriers for these architectures on ImageNet are still relatively high. The standard ResNet-50 architecture has a final-block bottleneck width of 1024, and we measure the barrier after REPAIR to be 20.1% in terms of test error. The double-width ResNet-50 variant has a final-block bottleneck width of 2048, reducing the barrier to 12.9%. In comparison, the widest ResNet20 we studied on CIFAR-10 had final-layer width of 1024 and a barrier of nearly zero. Therefore, it may be the case that for more difficult datasets, larger widths are required in order to reach low barriers.

3.11.2 Split data training

In this section, we study the setting where two endpoint networks are trained on disjoint splits of the training dataset. We aim to replicate the corresponding experiment of (Ainsworth et al., 2022), but using REPAIR applied to standard BatchNorm networks.

We first split the CIFAR-100 training set, consisting of 50,000 images distributed across 100 classes, into two disjoint sets of 25,000 images. The first split consists of a random 80% of the images in the first 50 classes, and 20% of the images in the second 50 classes, with the second split having the proportions reversed. We then train two networks, one for each split. The result is that one network is more accurate in the first 50 classes, and

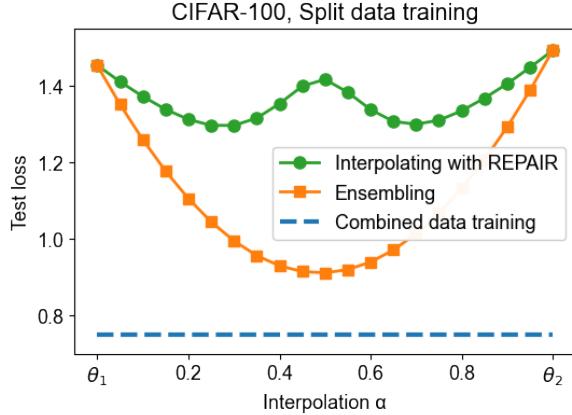


Figure 3.11: **Split data training (Jordan et al., 2022).** When two networks are trained on disjoint, biased subsets of CIFAR-100, their REPAIRRed interpolations outperform either endpoint with respect to the combined test set.

the other is more accurate in the second, with both performing worse than either their ensemble or a network trained on the full training set.

We next align the hidden units of these two networks and generate a series of linearly interpolated networks between the two, applying REPAIR to each (Figure 3.11). We find that many of these interpolated networks significantly outperform either of the two endpoints in terms of loss on the full CIFAR-100 test set. In this sense, the endpoint networks can be said to have been constructively merged. The best interpolated network of (Ainsworth et al., 2022) was reported to obtain a loss of 1.73 at $\alpha \approx 0.3$. Using REPAIR, our best interpolated network achieves a test loss of 1.30, also with mixing coefficient $\alpha = 0.3$. We attribute this improvement partially to REPAIR, and partially to the increased performance of standard ResNets compared to the LayerNorm-based variants used in (Ainsworth et al., 2022).

3.12 Conclusion

In this chapter, we tried to understand the loss landscape of neural networks and its implications on model generalization. We explored the geometric structure of the loss landscape based on the connectivity of neural networks, the role of overparameterization on the loss landscape shape, and the invariances of neural networks. We conjectured that by accounting for permutation invariance, barriers between different solutions in the loss landscape can be eliminated, with all solutions residing in the same basin. We approached the permutation invariance conjecture from both theoretical and empirical views, supporting our conjecture. We also explored various neuron alignment algorithms and compared them in finding permutations

to match two different trained neural networks. We later used a feature alignment method and observed the variance collapse phenomenon as the key behind the drop performance between aligned interpolated networks. To address the variance collapse problem we proposed REPAIR, a method that significantly improves the performance of these networks across various architectures and datasets. Our results provide support for our conjecture and contribute to understanding the underlying causes of the variance collapse phenomenon. Moreover, our REPAIR algorithm removed the barrier for many deep neural networks trained on various datasets. To the best of our knowledge, our reported barriers for ResNet-50s trained on ImageNet are the lowest reported in the community.

In conclusion, our exploration of the loss landscape of neural networks and the role of permutation invariance has provided valuable insights into improving generalization and the design of more efficient algorithms for initialization, ensembling, and distributed training. Although there are limitations to our study, the results obtained thus far offer a strong foundation for future research. By extending our findings to other tasks and addressing the computational challenges in searching the loss landscape, we can further enhance the capabilities of neural networks and their applications in various domains.

4 Pre-training and Generalization

Neural networks are heavily dependent on data, which provides the essential input for training and refining these powerful models. Data holds immense significance in the context of neural networks, as it lays the foundation for improving their performance.

Edge devices operate in diverse environments with varying conditions, making distribution shift a significant concern for their performance. In other words, the data distribution may change over time, leading to inaccuracies and discrepancies in models trained on such data. In Chapter 1, we introduced out-of-distribution generalization and briefly examined the roles of data augmentation and pre-training in enhancing performance. In this chapter, we begin by investigating the root cause behind the out-of-distribution performance drop and discuss advances made to improve out-of-distribution generalization. Motivated by the success of multimodal learning, we explore various multimodal architectures in Section 4.1.1 as well as multimodal datasets and the disparities concerning their sample sizes, sources, and caption cleanliness in Section 4.1.2.

We proceed to examine the influence of pre-training data distribution on transfer performance in Section 4.2. Transfer learning has gained popularity as a solution for tackling the challenges posed by limited labeled data in edge applications. In Section 4.5, we introduce various research questions concerning the role of data in transfer learning and address these questions through large-scale experiments, employing diverse combinations of pre-training and transfer datasets.

4.1 Background

In practice, poor performance on out-of-distribution inputs happens mostly because the data collection includes noise/error, practical test conditions change drastically/dynamically, or deliberate perturbations are applied to the input data. Such changes may include image corruptions *e.g.*, blurring, contrast/brightness, JPEG, adversarial examples, geometric transformations *e.g.*, rotations, etc.. Given a trained model, we can plot its in-distribution performance on the X-axis and its out-of-distribution performance on the

Y-axis. We consider a model robust to distribution shifts if it achieves a consistent in-distribution (ID) and out-of-distribution (OOD) accuracy. Figure 4.1 shows such a plot, where $y = x$ line illustrates the expected robust performance. However, in practice, we observe a gap between ID performance and OOD performance. Recht et al. (2019) investigated the cause for this accuracy drop and decomposed such a gap as the sum of overfitting (in case of test set re-use), distribution shift, and generalization error. Equation 4.1 illustrates this decomposition:

$$\widehat{Acc_S(f)} - \widehat{Acc_{S'}(f)} = \underbrace{\widehat{Acc_S(f)} - Acc_D(f)}_{\text{ID}_{\text{acc}} - \text{OOD}_{\text{acc}}} + \underbrace{Acc_D(f) - Acc_{D'}(f)}_{\text{distribution shift}} + \underbrace{Acc_{D'}(f) - \widehat{Acc_{S'}(f)}}_{\text{generalization error}} \quad (4.1)$$

Samples S and S' are draws from distributions D and D' , respectively. The expression on the left-hand side of the Equation 4.1 presents the performance gap between ID and OOD. $\widehat{Acc_S(f)}$ represents the mean-accuracy over samples S , as shown in Equation 4.2.

$$\widehat{Acc_S(f)} = \frac{1}{|S|} \sum_{(x,y) \in S} 1[f(x) = y] \quad \text{and} \quad Acc_D(f) = \mathbb{E}_{(x,y) \sim D} 1[f(x) = y] \quad (4.2)$$

The first term on the right-hand side of the Equation 4.1 attributes the gap between ID accuracy and OOD accuracy to overfitting due to the test set reuse. Overfitting due to test set reuse happens when the trained models have been adapted to the specific samples in the original test sets, for example via extensive hyperparameter tuning. Figure 4.1 (Left) illustrates that when test set reuse occurs, overfitting is likely to happen, leading to a larger discrepancy between ID and OOD performance. However, Figure 4.1 (right) (Recht et al., 2019) shows that the gap between ID and OOD performance shrinks for higher accuracy levels on CIFAR-10, showing that the root cause for such performance drop is not overfitting to the test set reuse. In practice, the generalization error term ($Acc_{D'}(f) - \widehat{Acc_{S'}(f)}$) is also small. Therefore, the main cause for the gap between ID accuracy and OOD accuracy should come from the remaining **distribution shift** term in Equation 4.1.

Various techniques can be employed to compensate for performance drop when distribution shifts happen. Data augmentation increases the diversity of the training data by applying different transformations to the input data and therefore improves OOD accuracy. Domain adaptation is another approach, which involves training the network on data similar, but not identical, to the target domain, enabling the network to adapt to new data while preserving its performance on the training data.

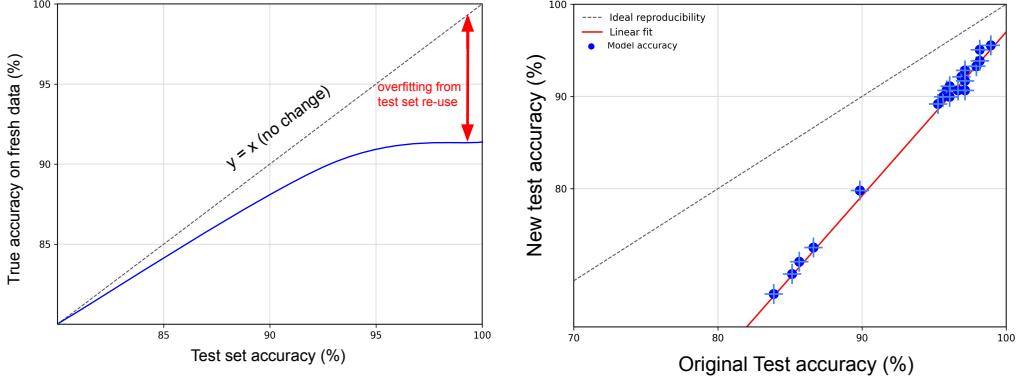


Figure 4.1: **Overfit is not causing the out-of-distribution performance drop.** **Left:** A sketch presenting the effect of overfit. X-axis shows the ID and Y-axis presents the OOD. The gap between ID and OOD should increase as the accuracy increase. **Right:** What actually happens when ID is original CIFAR-10 and OOD is new created CIFAR-10 (Recht et al., 2019). The gap between ID and OOD shrinks, showing that the root cause for such performance drop is not overfitting to test set reuse.

Taori et al. (2020) investigate various interventions to improve OOD generalization. Figure 4.2 summarizes these interventions on ImageNet (Deng et al., 2009) as ID dataset and ImageNet-v2 (Recht et al., 2019) as OOD dataset. Standard training refers to models trained on the ImageNet-ILSVRC 2012 training set without a specific robustness focus, from AlexNet (Krizhevsky et al., 2017) to VGGs (Simonyan & Zisserman, 2015), ResNets (K. He et al., 2016), and EfficientNet (Szegedy et al., 2015; Tan & Le, 2019).

Robustness interventions refer to models that are trained with explicit robustness interventions such as adversarially robust models (Cohen et al., 2019; Salman et al., 2019; Shafahi et al., 2019; C. Xie et al., 2019), models with special data augmentation such as AugMix (Hendrycks et al., 2019), CutMix (Yun et al., 2019), MixUp (H. Zhang et al., 2017), and models with architecture modifications (R. Zhang, 2019).

Figure 4.2 shows that there exist some models that deviate from the linear fit trend. The small outliers to the linear fit trend are those models trained with orders of magnitude larger data than the standard ImageNet training set e.g., models trained on the full ImageNet dataset of 21,841 classes (W. Wu, n.d.), YFCC 100 million dataset (Yalniz et al., 2019), Google’s JFT 300 million dataset (Sun et al., 2017; Q. Xie et al., 2020), or 1000 \times more data with 1 billion Instagram images (Mahajan et al., 2018; Q. Xie et al., 2020). These models show only small gains, in the best case improving the accuracy drop from 8.6% to 7.5% on ImageNetV2 for EfficientNet-L2 NoisyStudent (Q. Xie et al., 2020).

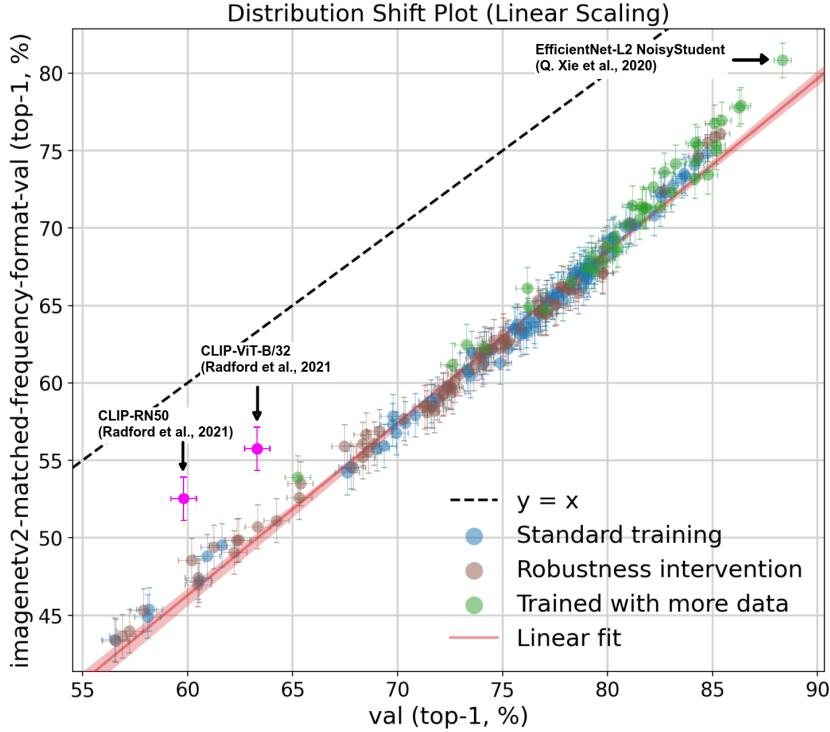


Figure 4.2: Overview of different methods to improve out-of-distribution generalization (Taori et al., 2020). Standard training refers to models trained on the ILSVRC 2012 training set without a specific robustness focus, from AlexNet (Krizhevsky et al., 2017) to VGGs (Simonyan & Zisserman, 2015), ResNets (K. He et al., 2016), and EfficientNet (Szegedy et al., 2015; Tan & Le, 2019). Robustness interventions refer to models that are trained with explicit robustness interventions such as adversarially robust models (Cohen et al., 2019; Salman et al., 2019; Shafahi et al., 2019; C. Xie et al., 2019), and models with special data augmentations (DeVries & Taylor, 2017; Engstrom et al., 2019; Geirhos et al., 2018; C. Xie et al., 2020). Models trained on orders of magnitude larger than the standard ImageNet training set show a small improvement above the linear fit and closer to $y = x$. Two trained CLIP models (Radford et al., 2021) with ResNet-50 (K. He et al., 2016) and ViT-B/32 (Dosovitskiy et al., 2020) backbones show the best performance under distribution shift.

However, there are two points in Figure 4.2 that are largely separated from the linear fit and are closer to $y = x$. These points are CLIP (Radford et al., 2021) models that have shown impressive performance under distribution shifts. Radford et al. (2021) leveraged the power of both image and text modalities to jointly learn image and text representations in a shared latent space. During the pretraining phase, CLIP learns to associate images and their corresponding textual descriptions by optimizing a contrastive loss function. The model is trained to maximize the dot product similar-

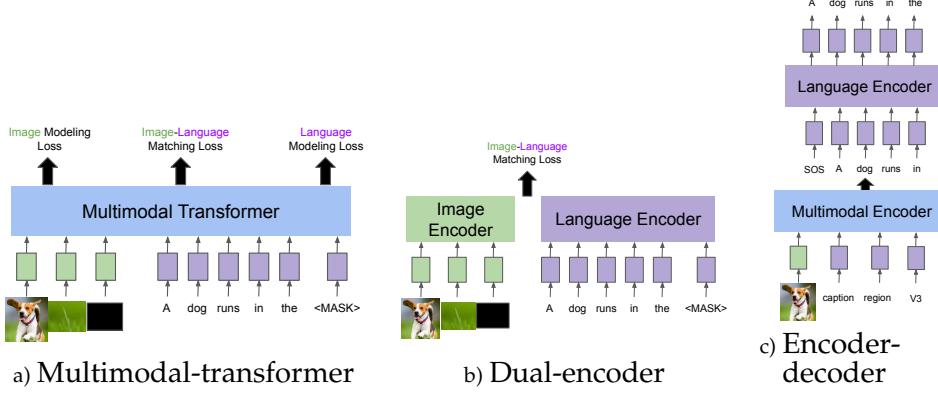


Figure 4.3: **Overview of multimodal pre-training architectures** (Nematzadeh, 2021) **Left:** Multimodal transformers use three types of loss and usually, each image is processed by an object detector **Middle:** In Dual Encoders, image and language modalities are processed separately, without any multimodal attention, *i.e.*, we can process image and language inputs, cache the features, and reuse them across pairs. Therefore dual Encoders are usually easier to scale, but they do not have the notion of multimodal cross attention, where the model learns to attend over both modalities. **Right:** Encoder Decoders models, also known as captioning models, process image and language and predict language.

ity between the embeddings of the corresponding image-text pairs while minimizing it for non-matching pairs. Later Fang et al. (2022) investigated the key success behind CLIP OOD robustness to understand if contractive loss, language supervision, or the new dataset is the reason for the improved OOD performance. They found that data distribution plays the most important role in CLIP’s robust performance.

4.1.1 Multimodal architectures

Motivated by CLIP’s success, there has been a surge in interest in using multi-modal architectures. Multimodal architectures integrate multiple data types to enhance AI versatility and adaptability (Bai et al., 2018; Baltrušaitis et al., 2018). These models have demonstrated improvements in tasks like zero-shot learning, image classification, and natural language understanding (Radford et al., 2021). Multimodal datasets can be gathered from internet on a large scale, offering abundant and varied data that can be utilized to enhance the performance of models (Radford et al., 2021).

Figure 4.3 depicts different architectures used for multimodal pre-training. **Dual encoders** (Srivastava & Salakhutdinov, 2012) consist of separate encoders for each modality, which are then combined through a shared latent space *e.g.*, CLIP. **Multimodal transformers** (J. Lu et al., 2019) are extensions of the standard transformer architecture, designed to handle multiple modalities simultaneously. They are characterized by a single encoder that

processes different modalities jointly, allowing the model to learn cross-modal interactions and representations more effectively. **Encoder-decoder** (Cho et al., 2014) architectures consist of separate encoders and decoders for each modality. The encoder processes the input modality, and the decoder generates the output modality. This architecture is particularly useful for tasks that require generating output in one modality given input from another, such as image captioning. Hendricks et al. (2021) and Miech et al. (2021) explored different architectures used for multimodal pretraining in the context of zero-shot domain transfer. They observed that given the same amount of pre-training data and evaluated on zero-shot retrieval, dual encoders (such as CLIP) perform better than multimodal transformers (joint encoders) and encoder decoders.

4.1.2 Multimodal datasets

Multimodal image and text datasets are valuable resources as they provide a rich source of information for the training and evaluation of machine learning algorithms. Figure 4.1 shows a list of more common multimodal datasets, where they differ in the number of samples, what is their source, and how clean their captions are.

SBU captions (Ordonez et al., 2011) is sourced from Flickr and features human-generated annotations. MS-COCO (Lin et al., 2014) is composed of images, each accompanied by five human-generated captions that have undergone specific filtering procedures to ensure high-quality images and annotations. Conceptual Captions (Sharma et al., 2018) is created by extracting images and their alt-text HTML attributes from the internet and annotating these images with refined descriptions. In addition to these datasets, visual question-answering tasks also serve as a data source for many image and text datasets, which are often structured in nature. A prominent large-scale dataset in this context is Visual Genome (Krishna et al., 2017), which contains a wealth of information in its structured data format.

Datasets like MS-COCO and WIT (Srinivasan et al., 2021) are considered to have *low noise* because they have been manually annotated or have captions that come from relatively reliable sources, such as Wikipedia articles. As a result, they generally contain cleaner and more informative captions. Datasets like Visual Genome and Conceptual Captions are considered to have *medium noise*. These datasets might have some inaccuracies or inconsistencies in the annotations or captions, either due to the automated extraction process (*e.g.*, Conceptual Captions) or the complexity of the annotations (*e.g.*, Visual Genome). Datasets like SBU Captions and LAION (Schuhmann et al., 2022) are considered to have *high noise* because they rely on user-generated captions or tags, which can be noisy, inconsistent, or even irrelevant to the

image content.

The availability of larger sample sizes in noisier datasets offers both challenges and opportunities for the development of multimodal models. As the volume of data increases, the potential for discovering meaningful patterns and relationships between images and text also grows. This increased dataset size can help improve the generalization capabilities of models by exposing them to a broader range of examples, leading to better performance on unseen data. Moreover, the diversity in data arising from various sources, including user-generated content, provides a more realistic representation of real-world scenarios that AI systems are likely to encounter. However, the presence of noise in these larger datasets necessitates the development of more sophisticated techniques to handle the inherent inconsistencies and inaccuracies (F.-L. Chen et al., 2023).

Several efforts have been made to understand the disparities between multimodal datasets concerning their performance on image recognition tasks. These studies aim to analyze the impact of different dataset characteristics, such as size, noise level, and annotation quality, on the effectiveness of models trained on these datasets. Hendricks and Nematzadeh (2021) examined MS-COCO, MS-COCO-narrative, and Visual Genomes, containing a fairly equal number of images. They observed that these datasets exhibit differing levels of performance when tested on Flickr30K (Young et al., 2014). In particular, although MS-COCO and MS-COCO-narrative share identical images, their distinct captions lead to varied outcomes. The authors observe that the language similarity between MS-COCO and Flickr is greater. They utilized perplexity (Chiusano, 2022) to measure language similarity. Perplexity, at a high level, quantifies the uncertainty of a language model when predicting the next word in a sequence. Hendricks and Nematzadeh (2021) pre-trained a multimodal transformer on Conceptual Captions and MS-COCO and test the performance of these trained models on two datasets, namely SVO (subject-verb-object) and FOIL (noun understanding) (Hendricks et al., 2021; Hendricks & Nematzadeh, 2021). They observed that MS-COCO with less noisy samples outperforms Conceptual Captions on both SVO and FOIL. The SVO dataset consists of image-caption pairs with simple subject-verb-object structures, focusing on the model's ability to understand the relationships between objects and actions in an image. The FOIL dataset is designed to evaluate models' understanding of the relationships between nouns and verbs in image-caption pairs. In this dataset, the authors create captions by replacing a single noun with an incorrect one while keeping the rest of the caption intact. The task is to identify the incorrect noun in the FOIL caption.

4 Pre-training and Generalization

Dataset Name	Sample image	Sample caption	Size	Caption Noise Level	Source
MS-COCO (Lin et al., 2014)		a dog on the beach with some people who also have a surfboard	~ 330K	low	Flickr
MS-COCO-narratives		In this image we can see a bridge and sea. In the background, we can see trees and the sky. We can see so many people on the bridge. At the bottom of the image, we can see two people. We can see stairs in the right bottom of the image ...	~ 2K	low	Flickr
Visual Genome (Krishna et al., 2017)		small round yellow frisbee, man has cast on his arm, concrete trail path in the park, man searing black sunglasses	~ 108K	medium	Flickr, Google Images
Conceptual Captions (Sharma et al., 2018)		The scenic route through mountain ranges includes these unbelievably colored mountains.	~ 12M	medium	alt-text from different webpages
SBU captions (Ordonez et al., 2011)		King Arthur's beheading rock - right on the sidewalk in the middle of town	~ 1K	high	Flickr
Redcaps (Desai et al., 2021)		the kids got t-shirts	~ 11M	medium	Reddit
YFCC (Thomee et al., 2016)		CHELSEA FOOTBALL CLUB Chelsea Magazine - Issue 63, November 2009	~ 400M	high	Flickr
LAION (Schuhmann et al., 2022)		I'm a Rugby Referee - Men's Premium Hoodie	~ 5B	high	Common Crawl
WIT		Hugo van der Goes, Saint Luke Drawing the virgin, c. 1470-80. National Museum of Ancient Art, Lisbon	~ 5M	low	Wikipedia
Shutterstock		Happy rich kenelbulle mascot design carries money bags	~ 11M	low	Shutterstock website

Table 4.1: **Multimodal image and text datasets.** Different datasets vary in the number of samples, what is their source, and how clean their captions are

4.2 The role of pre-training data in transfer learning

In the preceding section, we discussed that large-scale multi-modal pre-training methods such as CLIP have demonstrated impressive generalization capabilities, especially in their ability to handle out-of-distribution data (Figure 4.2). We noted that data plays the most important role in this success (Fang et al., 2022). Additionally, we reviewed recent research on the differences between various multimodal datasets and how these distinctions can impact *transfer learning* performance.

Transfer learning has become a popular approach for transferring representations learned by the pre-trained models to a new task and addressing the challenges of limited labeled data in various domains. Upstream and downstream refer to the different stages of the transfer learning pipeline. The upstream stage involves pre-training a model on a large dataset. This pre-trained model is then finetuned on a smaller dataset with labeled examples, referred to as the target or downstream dataset. Few-shot learning aims to learn from a small amount of labeled data for the downstream task, typically less than 20 samples per class. On the other hand, full-shot finetuning involves finetuning a pre-trained model on a large amount of labeled data to achieve high performance on a specific task. Zero-shot learning also refers to the ability of a model to generalize to unseen classes without any labeled examples.

In this section, we will extend our previous discussion on multimodal pre-training by assessing the impact of data distribution in the context of transfer learning. In contrast to prior works (Abnar et al., 2021; Bolya et al., 2021; Deshpande et al., 2021; Hendricks & Nematzadeh, 2021; C. Nguyen et al., 2020; You et al., 2021), our main focus is on the role of the pre-training data distribution in downstream performance.

We conduct an extensive empirical investigation in the context of transfer learning for computer vision tasks (see (Entezari et al., 2023) for details on 4000 experiments). Our study covers seven pre-training datasets including YFCC (Thomee et al., 2016), LAION (Schuhmann et al., 2022), Redcaps (Desai et al., 2021), Conceptual Captions-3m (Sharma et al., 2018), Conceptual Captions-12m (Changpinyo et al., 2021), WiT (Srinivasan et al., 2021), Shutterstock, and ImageNet (Deng et al., 2009) nine finetuning datasets including (CIFAR100 (Krizhevsky et al., 2009), DTD (Cimpoi et al., 2014), Caltech-101 (Fei-Fei et al., 2004), PETS (Parkhi et al., 2012), REAL and CLIPART from DomainNet (Peng et al., 2019), EuroSAT (Helber et al., 2019), Cassava Leaf Disease Classification (Divyanth L G, 2021), and Caltech Camera Traps-20 (Beery et al., 2018)) and three pre-training methods: supervised, CLIP (Radford et al., 2021) and SimCLR (T. Chen et al., 2020). To evaluate downstream performance, we examine both few-shot finetuning and full

finetuning.

In the following, we first review closely related works in Section 4.3, followed by our experimental setup presented in Section 4.4. Section 4.5 details our observations related to our research questions by measuring the downstream transfer accuracy of models pre-trained on various data sources, dataset sizes, and with different pre-training losses.

4.3 Related works

Our study of the role of pre-training data is inspired by and closely related to D. Kim et al. (2022) and Abnar et al. (2021). D. Kim et al. (2022) conducted an in-depth study of the effect of network architecture, pre-training dataset, supervised vs self-supervised learning objectives, and different domain transfer methods on the transferability of representations to new domains. They found that the transferability of the pre-trained representations depends on factors such as the target benchmark, adaptation method, and network depth. However, they do not study few-shot transfer, where we see the most impact of the pre-training distribution. They also did not provide a set of controlled experiments for some of the considered impacting factors because they are limited to available pre-trained models. For example, when comparing the role of data distribution (their Figure 2, ImageNet-21K (Deng et al., 2009) vs. JFT-300M (Sun et al., 2017)), they change the dataset size and also architecture, and the reader is left wondering if JFT has a better distribution for the transfer or if the observed effects come from more data or a better architecture.

Abnar et al. (2021) also explored how different upstream training settings affect transfer accuracy for two upstream datasets and more than 20 downstream tasks. They showed that as the upstream accuracy increases, the transfer learning performance on downstream datasets saturates. However, the authors study only upstream models that are pre-trained with a supervised loss function on ImageNet-21K or JFT-300M (different size and distributions). In this work, we extend these results to more pre-training datasets and methods, with a special focus on data distribution and curation. Abnar et al. (2021) also lack controlled comparison between different distributions in the pre-training datasets *e.g.*, they compare JFT and ImageNet with different distribution and sample sizes. We consider full finetuning in addition to few-shot transfer.

Deshpande et al. (2021), C. Nguyen et al. (2020), and You et al. (2021) develop metrics for predicting transferability of a model. Their main focus is to develop a measure to predict the full finetune accuracy without actually finetuning on a downstream task. While we also cover full finetuning accuracy, our main research question lies in studying the extent to which

pre-training data affect transfer accuracy. Looking at few-shot and full-shot also gives us the ability to study the effect of transfer learning as more target data become available. Moreover, predictability of the transfer performance is mostly limited to supervised ImageNet-1K pretraining, while we scale both pre-training distributions, size, and pre-training loss functions. The transferability line of research mainly focuses on the Internet-crawled datasets, while we extended our results to domain-specific datasets (Camera Traps, Cassava Leaf Diseases, and EuroSAT),

4.4 Methodology

Model Our main focus is CLIP (Radford et al., 2021). This model has demonstrated unprecedented robustness to natural distribution shifts (Miller et al., 2021; Taori et al., 2020), and transfers well to many downstream tasks (Radford et al., 2021; Wortsman et al., 2021). Given an image-text pair, CLIP learns a joint embedding space for both images and their captions and tries to maximize the cosine similarity between the text and image relative to the cosine similarity of unaligned pairs. We use the CLIP implementation from the OpenCLIP GitHub repository (Ilharco et al., 2021).

Pre-training We mainly use ResNet-50 (K. He et al., 2016) as the image encoder unless stated otherwise. We vary the pre-training data distribution in Section 4.5.1, curation method in Section 4.5.4, and pre-training dataset size in Section 4.5.6 to obtain different pre-trained models. We also change the contrastive loss function to SimCLR in Section 4.5.7 to test the effect of the pre-training method on downstream transfer accuracy. (For further training details refer to (Entezari et al., 2023)).

finetuning For most of the experiments we finetune the pre-trained model end-to-end on a target transfer dataset unless stated otherwise. For each pre-trained model and downstream transfer dataset, we used a large grid search over various finetuning hyperparameters including learning rate, batch size, and the number of epochs. We report the best-performing accuracy in the plots. For further finetuning details refer to (Entezari et al., 2023).

Datasets Our large-scale experiments yield more than 4000 trained networks. Our pre-training datasets consist of the million-size image and language pairs from multiple recent multi-modal datasets including YFCC (Thomee et al., 2016), LAION (Schuhmann et al., 2022), RedCaps (Desai et al., 2021), Shutterstock, Conceptual Captions (Changpinyo et al., 2021; Sharma et al., 2018), WiT (Srinivasan et al., 2021). Our pre-training datasets are

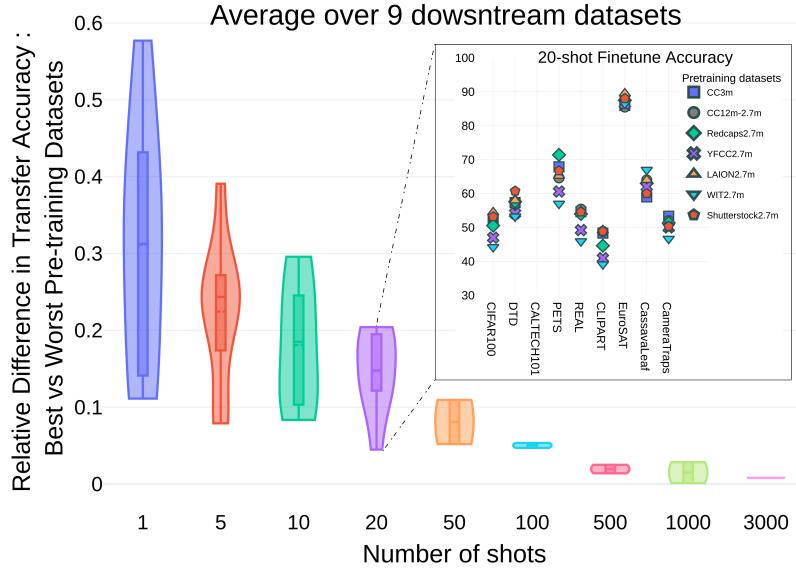


Figure 4.4: **Differences between various pre-training sources diminish as more data is available for the downstream tasks (Entezari et al., 2023).** In the few-shot setting, different pre-training datasets lead to noticeable differences in downstream performance. However, if many samples are available for finetuning, the difference in absolute accuracy between models pre-trained on different sources largely evaporates. See Figure 4.10 for a detailed view of this plot.

crawled from different sources covering different data distributions (See (Entezari et al., 2023) for details of pre-training sources and their samples). For downstream tasks, we use nine different datasets CIFAR100, DTD, Caltech-101, Oxford-PETS, REAL, and CLIPART from DomainNet, EuroSAT, Cassava Leaf Disease, and Caltech Camera Traps (Beery et al., 2018; Cimpoi et al., 2014; Divyanth L G, 2021; Fei-Fei et al., 2004; Helber et al., 2019; Krizhevsky et al., 2009; Parkhi et al., 2012; Peng et al., 2019). While the first six datasets are Internet-crawled datasets (similarly to the pre-training datasets) and are more common in transfer learning in computer vision benchmarks, we include three new downstream datasets that are domain-specific, i.e. the dataset is created to solve a well-defined challenge in a specific domain.

4.5 Research questions

In the subsequent sections, we pose several research questions to examine the significance of pre-training data, the selection of pre-training techniques (supervised, CLIP, or SimCLR), and the effect of ImageNet distribution on downstream performance. To address these questions, we carefully devise experiments for each section, while ablating other impacting factors.

4.5.1 What is the impact of different pre-training data sources on transfer learning?

Do we expect different distributions to perform differently in the transfer setting? Figure 4.4 aggregates transfer performance from different pre-training datasets across all downstream datasets. To obtain each point, we (1) pre-train CLIP models using a set of seven large data sources, (2) finetune each pre-trained model on all downstream datasets using a different number of shots (a sweep over multiple hyperparameters), and (3) for each downstream dataset, calculate the difference between the *best* and *worst* finetune performance among used pre-training sources, normalized by the maximum finetune performance. Figure 4.4 aggregates over all downstream datasets for each number of shots, highlighting as an example different pre-training models finetuned using 20 samples/class on all downstream datasets. We observe that changing the pre-training dataset leads to noticeable differences in the downstream performance in a few-shot setting. However, as more images are available for finetuning, the difference in absolute accuracy between different pre-training models is largely diminished. Figure 4.10 shows this diminishing effect in detail for different downstream datasets. The fully finetuned models have very similar downstream performance despite different pre-training datasets (see the top-right point of CIFAR100 and REAL in Figure 4.10, and also the top-right point for CameraTraps, Cassava Leaf, and EuroSAT in Figure 4.10). However, this is not true for DTD, CALTECH101, PETS, and CLIPART, where they have far fewer images per class for finetuning on the full dataset.

Table 4.2 compares finetune accuracy for different pre-training choices along all downstream datasets. Transfer learning from even the worst pre-training dataset outperforms training from scratch largely. The gap between best and worst-performing pre-training datasets is small. For results on the Vision Transformers (Dosovitskiy et al., 2021) instead of ResNet-50 refer to (Entezari et al., 2023).

4.5.2 Which data distribution is better for transfer learning?

The detailed results in Figure 4.10 demonstrate that pre-training on the Shutterstock and LAION datasets results in superior transfer performance across a range of downstream tasks. A closer look shows a superior performance of Redcaps for PETS. We investigate this further and inspect many pets by looking at random samples from Redcaps (see (Entezari et al., 2023) for these samples). We also look into the most common words in the captions of these pre-training datasets, summarized in Table 4.3. We observe that "cats" and "dogs" are among the most common words in the Redcaps dataset.

	CIFAR100	DTD	CALTECH	PETS	DomainNet REAL	DomainNet CLIPART	EuroSAT	Cassava Leaf Disease	Camera Traps	average
Train from scratch (no pre-training)	72.82	44.62	55.32	67.96	77.76	55.20	58.50	70.30	89.10	65.73
Worst performing pre-training	81.62	61.06	82.38	83.15	81.87	68.98	70.72	87.05	98.85	79.52
Best performing pre-training	83.71	68.56	86.88	87.84	82.92	72.38	72.79	87.57	98.91	82.39

Table 4.2: Full finetune accuracy from six different pre-training datasets across all downstream datasets (Entezari et al., 2023). Transfer learning from even the worst pre-training dataset outperforms training from scratch largely.

Table 4.3 also shows that "background", "design", "pattern", and "texture" are among the most common words in the captions of Shutterstock, supporting a high correlation between DTD (Describable Textures Dataset) and Shutterstock. These observations support the intuition between the closeness of the pre-training data and the target tasks, both in visual and caption (label) domains.

4.5.3 How much pre-training contributes to downstream performance as opposed to training from scratch?

While transfer learning from a large pre-training dataset outperforms training from scratch for all downstream tasks, the magnitude of the improvement varies for different datasets in Figure 4.10. We observe a large improvement for PETS, CALTECH-101, and CLIPART. PETS, for example, has a small number of samples per class for training (30), which makes it hard to train from scratch. PETS is also scraped from the web (Fei-Fei et al., 2004) *i.e.*, Google image search, similar to our web-scraped pre-training sources. Therefore, we hypothesize that the benefits of pre-training over training from scratch become more apparent when the pre-training and target tasks are closely related, both in terms of semantic content and sample similarity.

4.5.4 Do well-curated pre-training datasets lead to better transfer?

There has been a significant effort to create computer vision datasets with high-quality labels. On the other hand, many recent datasets are large but noisy *e.g.*, LAION. This challenge is especially pronounced in domain-specific machine learning applications where obtaining accurate labels can be a difficult and resource-intensive process. Expert opinions are often

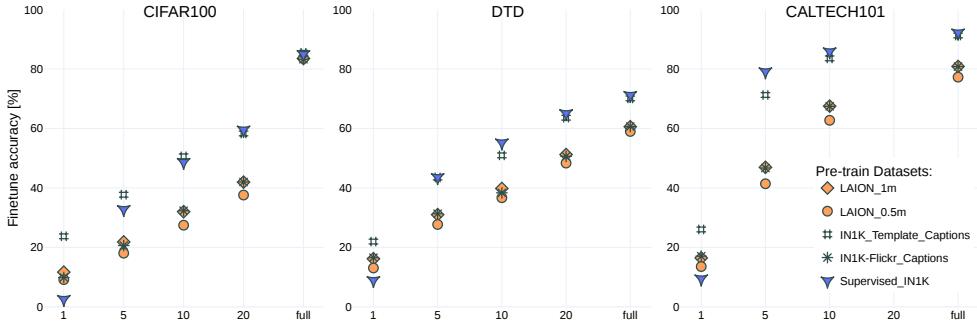


Figure 4.5: **Effect of data curation and labeling (Entezari et al., 2023)**. We compare supervised pre-training on ImageNet-1K to (1) contrastive pre-training on original captions from Flickr with 0.5m samples, and (2) contrastive pre-training on templated (clean) captions using ImageNet labels with 1.2m samples. Supervised pre-training on ImageNet leads to a better transfer accuracy than contrastive pre-training. Improving captions quality from Flickr to Template leads to huge improvements in downstream transfer accuracy, highlighting the importance of caption quality. On a different comparison between ImageNet and LAION distributions, pre-training CLIP on Flickr captions outperforms pre-training on LAION distribution with the same size (0.5m). For more results on other target datasets see (Entezari et al., 2023).

relied upon to generate these labels, but even these can introduce noise and inconsistencies due to human biases and errors. As machine learning models are increasingly deployed on edge devices, the need for high-quality training data becomes even more crucial (Cao et al., 2020). Edge devices, with their inherent limitations in computational power and storage, require models that are both efficient and robust to noise in order to deliver accurate and reliable performance. This underscores the importance of addressing the issue of noisy labels, particularly in domain-specific tasks and their implementation on edge devices. In this section, we are going to investigate:

How much is laborious ImageNet labeling worth?

To answer this question, we first start by pre-training ResNet-50 ImageNet-1K (ILSVRC 2012), using supervised cross-entropy loss and finetune on our downstream datasets in Figure 4.5. To investigate the role of supervision, we then discard ImageNet labels and use CLIP to pre-train on ImageNet. Because the ImageNet dataset has no captions, we include original Flickr captions, which reduces the size of the image and captions to 0.5M samples, see (Entezari et al., 2023; Fang et al., 2022) describing the required steps to create ImageNet-Flickr. Figure 4.5 shows that supervised pre-training on ImageNet outperforms CLIP pre-training on ImageNet with Flickr captions by a large margin in all downstream tasks.

However, such a gap could be attributed to two differences between mentioned pre-trainings: (1) supervised vs. contrastive image-language loss,

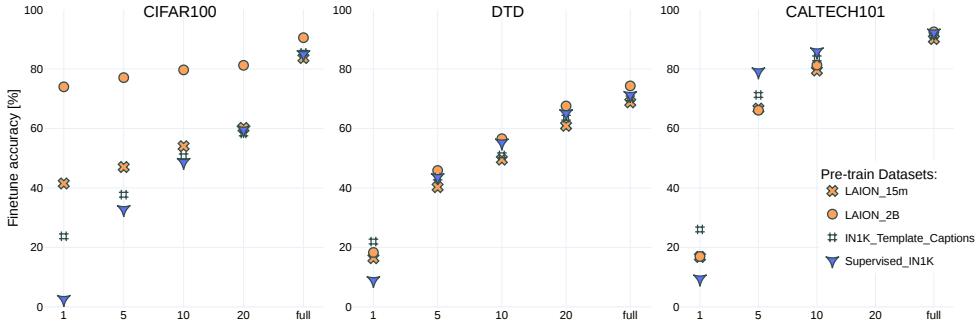


Figure 4.6: How much LAION data is worth of ImageNet pre-training? (Entezari et al., 2023) While Figure 4.5 shows the superiority of supervised pre-training over contrastive pre-training with the same size, here we increase the size of contrastive pre-training size to see if contrastive pre-training could perform better than supervised pre-training at scale. Including 15x more data from LAION outperform supervised ImageNet pre-training (and template captions) on CIFAR100. However, DTD needs 2000x more data from LAION to match or outperform ImageNet pre-training. Even including 2000x more data did not help CALTECH-101, where supervised ImageNet pre-training is still the best choice. For extended results on other target datasets see (Entezari et al., 2023).

and (2) the size of training samples for supervised-ImageNet (1.2m) is two times larger than CLIP with ImageNet-Flickr captions (0.5m). To remove the second effect we then use all the images from ImageNet, paired with templated clean captions, e.g., “a photo of a *class name*”. This allows us to have a fair comparison between supervised and CLIP pre-training on ImageNet, given the same size. Figure 4.5 shows that pre-training with clean captions improves the performance of CLIP pre-training by a large margin and outperforms supervised pre-training on CIFAR100. However, supervised pre-training on ImageNet still performs best for the rest of the datasets.

4.5.5 How does the effectiveness of ImageNet pre-training compare to that of LAION pre-training?

Figure 4.5 shows that pre-training CLIP on ImageNet (with template captions) outperforms LAION with the same size (1m) by a large margin. However, this gap shrinks as more data for the downstream task are available. Interestingly, pre-training CLIP on LAION-1m is only as good as ImageNet with Flickr captions with half of the data (0.5m).

What if we scale LAION pre-training size? Figure 4.6 shows that including 15× more data from LAION outperforms ImageNet pre-training with template captions only on CIFAR100 transfer. DTD, REAL, and CLIPART need 2000× more data from LAION to match or outperform ImageNet pre-training. However, even including 2000× more data did not help CAL-

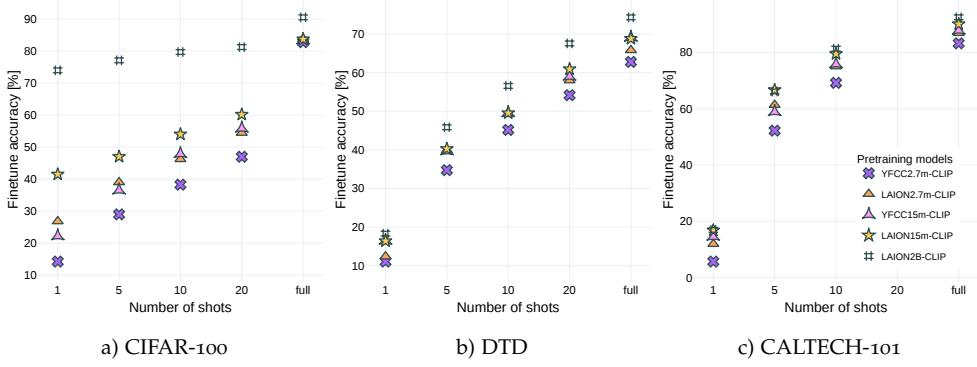


Figure 4.7: **Effect of the pre-training dataset size** (Entezari et al., 2023). Increasing the size of the dataset used for pre-training results in better transfer accuracy on downstream tasks. However, the absolute accuracy difference is smaller in the high-shot regime, even when pre-training consists of $100\times$ more data. The benefit of pre-training on LAION-2B is different on target tasks. While there is a major gap between LAION-2B and LAION-15m for CIFAR100, the performance gain from scaling up the pre-training dataset on CALTECH101 gets saturated. For extended results on other target datasets see (Entezari et al., 2023).

TECH101. ImageNet pre-training also outperforms LAION-2B on PETS by a large margin. Our hypothesis is that the overlapping samples of pets, such as different dog breeds, in both PETS and ImageNet datasets contribute to this similarity.

4.5.6 How does the downstream performance improve as more data is available for pre-training?

Should we expect that more pre-training data implies a better performance? Does the pre-training effectiveness saturate at some point? We fix the pre-training distribution to YFCC and LAION and compare pre-training on 2.7m samples with 15m samples. We also extend our experiments to see the effect of extreme sample sizes and include ViT-B/32 CLIP model trained on 2b samples from LAION. Figure 4.7 shows that increasing the size of the dataset used for pre-training results in higher downstream transfer accuracy. However, the magnitude of the improvement varies across different downstream datasets. While increasing the pre-training size of YFCC and LAION improves the CIFAR100 performance by a large margin, this improvement is modest for the rest of the downstream datasets. Specifically including 2 billion samples from LAION does not help CALTECH101 and PETS. Similarly to the findings in Figure 4.4, in larger sizes, we observe more noticeable differences in downstream performance in the few-shot regime. The difference in the absolute accuracy when more data is available for finetuning is usually smaller. However, in contrast to the findings by Abnar et al. (2021),

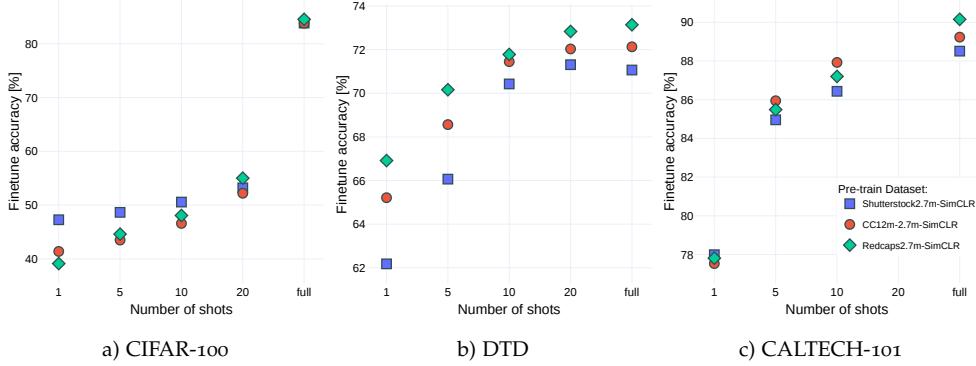


Figure 4.8: **Effect of the pre-training data distribution when using SimCLR as the pre-training method** (Entezari et al., 2023). Using different datasets for pre-training leads to a noticeable difference in downstream transfer accuracy. Similarly to the previous results for CLIP pre-training, the absolute difference in downstream transfer accuracy between different pre-training datasets is smaller when many images are available for finetuning. For more results on other datasets see (Entezari et al., 2023).

pre-training on the extremely large LAION-2B still manages to boost the downstream performance in the full finetuning mode. Figure 4.7 shows that LAION pre-training outperforms YFCC pre-training on downstream tasks for both 2.7m and 15m subsets of the datasets. Interestingly, pre-training on LAION-2.7m performs similarly to a much larger in size of YFCC-15m pre-training, highlighting the efficiency of the LAION distributions. Using an extremely large dataset of LAION-2B improves the performance by a significant margin in the few-shot regime for CIFAR100. While differences in absolute accuracy are smaller as more data is available for finetuning, LAION-2B pre-training still performs consistently better.

4.5.7 Effect of pre-training loss

In this section, we replace the pre-training loss from language-image contrastive in CLIP with image-only contrastive loss in SimCLR (T. Chen et al., 2020). Figure 4.8 highlights that our observations from Figure 4.4 are now extended to image-only pre-training, *i.e.*, changing the pre-train dataset leads to differences only in the few-shot downstream performance. Next, we evaluate the distinction between pre-training using CLIP and SimCLR. The results are presented in Figure 4.9. Overall we find that the models pre-trained with SimCLR have better downstream transfer accuracy than the models pre-trained with CLIP in the few-shot regime.

Similarly to our observations regarding the effect of the pre-training data distribution (Figures 4.4, 4.5, and 4.8), the absolute accuracy difference is smaller when more data is used for finetuning. We note that this is different

4 Pre-training and Generalization

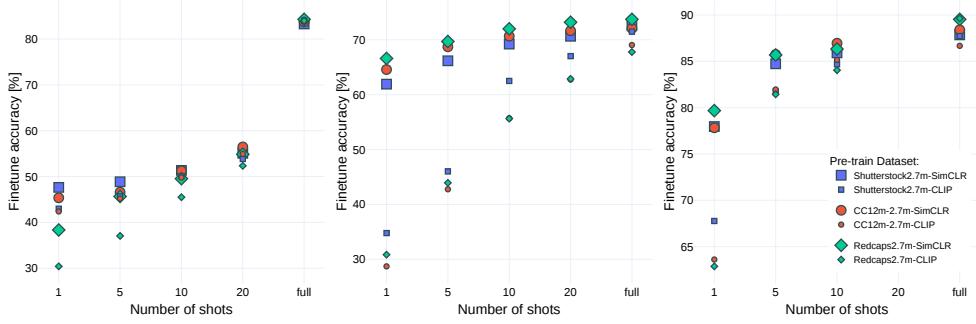


Figure 4.9: **Pre-training with CLIP vs. SimCLR (Entezari et al., 2023)**. Overall we observe that SimCLR pre-training results in a better downstream transfer accuracy than using CLIP pre-training on the same dataset. These differences are more pronounced in the few-shot setting.

from what (Santurkar et al., 2022) observed. However, we suspect this difference is because we are finetuning all model parameters while they only consider a linear classifier.

The difference in the downstream transfer accuracy for CLIP and SimCLR pre-training varies across different datasets. While SimCLR is only marginally better than CLIP for CIFAR100, the difference is significantly larger for DTD and CALTECH101, especially in the few-shot setting.

4 Pre-training and Generalization

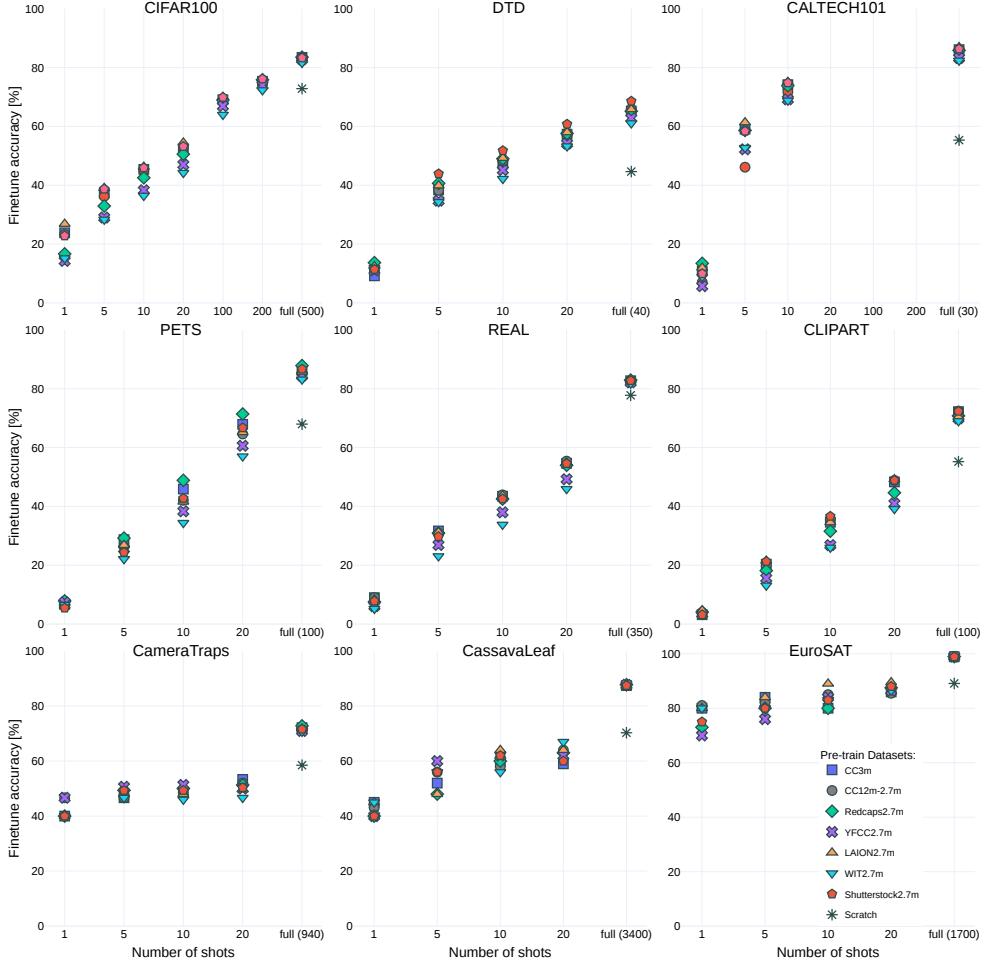


Figure 4.10: Effect of the pre-training data distribution (Entezari et al., 2023). While Figure 4.4 shows the aggregated results all downstream datasets, here we include the performance for each pair of (pretraining, downstream) datasets in detail. We provide a detailed analysis of the performance for each specific combination of pretraining and downstream datasets (as opposed to Figure 4.4 presenting the aggregated results for all downstream datasets). In the low-shot setting, different pre-training datasets lead to noticeable differences in downstream performance. If many samples are available for finetuning, the difference in absolute accuracy between the models pre-trained on different sources largely evaporates.

Table 4.3: Most common words in captions of pre-training distributions (Entezari et al., 2023).

Pre-training dataset	Top 20 words in 1M sample of captions
Shutterstock	background , vector, illustration, design , icon, pattern , texture , style, woman, concept, hand, color, flower, view, template, line, business, logo, card, symbol
Redcaps	day, today, year, time, cat, plant, friend, anyone, picture, baby, guy, week, dog, home, morning, night, month, way, boy, work
YFCC-15m	photo, day, park, street, city, picture, view, time, world, year, house, state, center, part, garden, shot, image, building, road, museum
LAION-15m	photo, stock, image, black, woman, design, set, vector, white, print, home, men, blue, dress, art, card, sale, gold, bag, cover
CC-12m	illustration, stock, art, design, photo, image, background, room, vector, house, home, woman, wedding, style, photography, royalty, car, fashion, girl, world
CC-3m	background, actor, artist, player, illustration, view, woman, man, football, team, tree, premiere, city, vector, day, girl, beach, game, hand, people
WIT	view, church, station, map, house, building, hall, museum, city, location, street, park, river, state, john, county, town, center, bridge, world

4.6 Conclusion

In this chapter, we have emphasized the critical role of data in training and generalizing neural networks, specifically focusing on the impact of pre-training data distribution on transfer performance. We explored the root cause behind out-of-distribution performance drops and discussed advances targeting improvement in generalization under such circumstances. Moreover, we examined the transfer learning paradigm and highlighted its growing significance as more pre-trained models become accessible.

Our findings indicate that variations in pre-training distributions and methods can result in differences in downstream transfer accuracy, particularly in the few-shot transfer regime. However, these disparities tend to decrease when a larger number of images are utilized for finetuning. We also discovered that the pre-training method influences performance on downstream transfer, and incorporating more pre-training data may help bridge the performance gap between training methods, such as supervised and contrastive approaches.

Nonetheless, our study has its limitations. Considering the resource constraints on edge devices, one could explore lightweight finetuning methods,

like linear probing, instead of full finetuning. Furthermore, while we performed an exhaustive hyperparameter sweep for finetuning, we did not do the same for pre-training, primarily due to the cost disparity between the two processes.

Our findings offer valuable insights into transfer learning and provide a foundation for future research. By exploring the influence of different pre-training methods and data distributions on downstream transfer accuracy, we encourage further investigation into optimal pre-training techniques and the interplay between data size and model performance. Our work also highlights current limitations, opening up opportunities for future research to develop lightweight finetuning methods and optimize the pre-training process. Given our results on the disparities between datasets in transfer performance, one promising future direction is the development of improved pre-training datasets. By carefully curating and refining these datasets, researchers can potentially enhance the effectiveness of pre-training methods and consequently improve the transfer learning process. This advancement would contribute to the overall performance of neural networks, particularly in edge environments, where limited labeled data is available.

5 Conclusion and Outlook

The recent development of deep neural networks promised to bring machine intelligence to our daily life. However, the resource demands of neural networks limit their scalability, accessibility, and deployment on edge devices like mobile phones or embedded systems. Furthermore, Edge devices are often deployed in diverse environments with different conditions, such as varying temperatures. Therefore data collected by edge devices can be shifted over time. In addition, this data is often noisy, incomplete, or corrupted due to factors like sensor inaccuracies, communication errors, or environmental influences. These reasons motivate the implementation of machine learning models that can handle different variations and generalize well to unseen data and conditions.

In this dissertation, we have explored the generalization of neural networks, investigating the challenges and limitations these models face when deployed on edge devices. This concluding chapter summarizes the key findings from each chapter and discusses how our research advances the understanding of neural networks at the edge while offering potential future research directions.

In Chapter 2, we investigated the interplay between sparsity and robustness in neural networks, particularly for edge devices operating in dynamic environments. We discovered that sparsity can improve network robustness without negatively impacting overall accuracy. Our findings also revealed that models trained with a contrastive objective are more sensitive to the introduction of sparsity relative to traditional supervised training with a cross-entropy loss. Moreover, we tackled data and class imbalance challenges by proposing an end-to-end sparsity method that incorporates a parameterized loss function, showcasing its effectiveness in real-world edge applications. Our sparsity-driven approach demonstrates significant promise for edge devices operating in diverse environments with varying conditions. We also highlighted the importance of domain expertise in medical imaging applications and demonstrated the potential of sparse neural networks in this area. By applying sparsity techniques to nuclei instance segmentation in medical images, we observed that sparse models maintain high accuracy and robustness against distribution shifts, which is critical in real-world edge applications. Our sparsity-driven approach demonstrates significant promise for edge devices operating in diverse environments with varying conditions.

In Chapter 3, we delved into the loss landscape of neural networks and its implications on model generalization. We explored the geometric structure of the loss landscape, the role of overparameterization on the loss landscape shape, and the invariances of neural networks. We conjectured that by accounting for permutation invariance, barriers between different solutions in the loss landscape can be eliminated, with all solutions residing in the same basin. We provided theoretical and empirical support for our conjecture and proposed the REPAIR method, which improves the performance of aligned interpolated networks. Our exploration of the loss landscape contributes to improving generalization and designing more efficient algorithms for initialization, ensembling, and distributed training.

In Chapter 4, we emphasized the critical role of data in training and generalizing neural networks, specifically focusing on the impact of pre-training data distribution on transfer performance. Our findings indicate that variations in pre-training distributions and methods can result in differences in downstream transfer accuracy, particularly in the few-shot transfer regime. We also discovered that the pre-training method influences performance on downstream transfer, and incorporating more pre-training data may help bridge the performance gap between training methods, such as supervised and contrastive approaches.

This thesis advances the understanding of neural networks at the edge by examining sparsity, loss landscape, and the role of data in training and generalization. The insights provided in this research can be used to guide future work on neural network optimization and deployment in edge environments. Some examples may include but are not limited to:

1. Developing pruning methods that address fairness concerns with respect to underrepresented groups.
2. Investigating training dynamics to improve model adaptation under resource constraints, *e.g.*, effectively training sparse networks from scratch.
3. Extending our findings on sparsity and loss landscape to understand why Lottery Ticket Hypothesis works and propose methods to find sparse sub-networks without expensive iterative training and pruning procedure.
4. Investigating the impact of REPAIR in practice *e.g.*, improving test accuracy by learning ensembles or improving federated learning techniques.
5. Investigating the role of language in contrastive pre-training *e.g.*, quality of captions in zero-shot and finetune performance.
6. Extending our results on the quality of the pre-training datasets by designing new filtering techniques or curating new data sources *e.g.*, DataComp (Gadre et al., 2023).

Bibliography

- Abnar, S., Dehghani, M., Neyshabur, B., & Sedghi, H. (2021). Exploring the limits of large scale pre-training. *arXiv preprint arXiv:2110.02095* (cit. on pp. 82, 83, 90).
- Aguech, M., & Carlier, G. (2011). Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2), 904–924 (cit. on p. 63).
- Ainsworth, S. K., Hayase, J., & Srinivasa, S. (2022). Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836* (cit. on pp. 60, 65, 70–72).
- Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. (2022). Flamingo: A visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198* (cit. on p. 16).
- Arora, S., Cohen, N., Golowich, N., & Hu, W. (2018). A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281* (cit. on p. 2).
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450* (cit. on p. 70).
- Bai, X., Yang, M., Lyu, P., Xu, Y., & Luo, J. (2018). Integrating scene text and visual appearance for fine-grained image classification. *IEEE Access*, 6, 66322–66335 (cit. on p. 78).
- Baldassi, C., Pittorino, F., & Zecchina, R. (2020). Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1), 161–170 (cit. on p. 51).
- Baldock, R., Maennel, H., & Neyshabur, B. (2021). Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34 (cit. on pp. 26, 28).
- Baltrušaitis, T., Ahuja, C., & Morency, L.-P. (2018). Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2), 423–443 (cit. on p. 78).
- Beery, S., Van Horn, G., & Perona, P. (2018). Recognition in terra incognita. *Proceedings of the European conference on computer vision (ECCV)*, 456–473 (cit. on pp. 82, 85).
- Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32), 15849–15854 (cit. on pp. 54, 55).

Bibliography

- Bellegarda, J. R., de Souza, P. V., Nádas, A. J., Nahamoo, D., Picheny, M. A., & Bahl, L. R. (1992). Robust speaker adaptation using a piecewise linear acoustic mapping. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 1, 445–448 (cit. on p. 13).
- Berend, D., & Kontorovich, A. (2015). A finite sample analysis of the naive bayes classifier. *J. Mach. Learn. Res.*, 16(1), 1519–1545 (cit. on p. 9).
- Biggio, B., & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84, 317–331. <https://doi.org/10.1016/j.patcog.2018.07.023> (cit. on p. 20)
- Bolya, D., Mittapalli, R., & Hoffman, J. (2021). Scalable diverse model selection for accessible transfer learning. *Advances in Neural Information Processing Systems*, 34, 19301–19312 (cit. on p. 82).
- Brea, J., Simsek, B., Illing, B., & Gerstner, W. (2019a). Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911* (cit. on p. 51).
- Brea, J., Simsek, B., Illing, B., & Gerstner, W. (2019b). Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. (Cit. on p. 58).
- Breiman, L. (1996). Bagging predictors. *Machine learning*, 24, 123–140 (cit. on pp. 8, 9).
- Breiman, L. (2001). Random forests. *Machine learning*, 45, 5–32 (cit. on pp. 8, 9).
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877–1901 (cit. on p. 4).
- Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106, 249–259 (cit. on p. 32).
- Cao, N., Meyer, M., Thiele, L., & Saukh, O. (2020). Automated pollen detection with an affordable technology. *EWSN*, 108–119 (cit. on p. 88).
- Changpinyo, S., Sharma, P., Ding, N., & Soricut, R. (2021). Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3558–3568 (cit. on pp. 82, 84).
- Chawla, N. V., Bowyer, K. W., & et. al. (2002). SMOTE: Synthetic minority over-sampling technique. *AI research*, 16, 321–357 (cit. on p. 32).
- Chen, F.-L., Zhang, D.-Z., Han, M.-L., Chen, X.-Y., Shi, J., Xu, S., & Xu, B. (2023). Vlp: A survey on vision-language pre-training. *Machine Intelligence Research*, 20(1), 38–56 (cit. on p. 80).

Bibliography

- Chen, J., Chen, S., & Pan, S. J. (2020). Storage efficient and dynamic flexible runtime channel pruning via deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 14747–14758 (cit. on p. 6).
- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations, 1597–1607 (cit. on pp. 27, 82, 91).
- Chiusano, F. (2022). *Two minutes nlp — perplexity explained with simple probabilities*. <https://medium.com/nlplanet/two-minutes-nlp-perplexity-explained-with-simple-probabilities-6cdc46884584>. (Cit. on p. 80)
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (cit. on p. 79).
- Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., & Vedaldi, A. (2014). Describing textures in the wild. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3606–3613 (cit. on pp. 82, 85).
- Cohen, J., Rosenfeld, E., & Kolter, Z. (2019). Certified adversarial robustness via randomized smoothing. *international conference on machine learning*, 1310–1320 (cit. on pp. 76, 77).
- Cortes, C., & Mohri, M. (2004). AUC optimization vs. error rate minimization. *NIPS*, 313–320 (cit. on p. 34).
- Corti, F., Entezari, R., Hooker, S., Bacciu, D., & Saukh, O. (2022). Studying the impact of magnitude pruning on contrastive learning methods. *arXiv preprint arXiv:2207.00200* (cit. on pp. 8, 29–31).
- Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1), 21–27 (cit. on p. 28).
- Croce, F., & Hein, M. (2020). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *International Conference on Machine Learning*, 2206–2216 (cit. on pp. 21, 24).
- Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* (cit. on p. 13).
- Cui, Y., Jia, M., Lin, T.-Y., & et. al. (2019). Class-balanced loss based on effective number of samples. *CVPR*, 9268–9277 (cit. on pp. 32, 34, 36).
- Davis, J., & Frank, L. (2022). Revisiting batch norm initialization. *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI*, 212–228 (cit. on p. 69).
- Deng, J., Dong, W., Socher, R., & et. al. (2009). ImageNet: A large-scale hierarchical image database. *CVPR*, 248–255 (cit. on pp. 15, 20, 36, 55, 70, 76, 82, 83).
- Desai, K., Kaul, G., Aysola, Z., & Johnson, J. (2021). Redcaps: Web-curated image-text data created by the people, for the people. *arXiv preprint arXiv:2111.11431* (cit. on pp. 81, 82, 84).

Bibliography

- Deshpande, A., Achille, A., Ravichandran, A., Li, H., Zancato, L., Fowlkes, C., Bhotika, R., Soatto, S., & Perona, P. (2021). A linearized framework and a new benchmark for model selection for fine-tuning. *arXiv preprint arXiv:2102.00084* (cit. on pp. 82, 83).
- DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (cit. on pp. 13, 77).
- Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, 2000 Proceedings* 1, 1–15 (cit. on p. 8).
- Dinh, L., Pascanu, R., Bengio, S., & Bengio, Y. (2017). Sharp minima can generalize for deep nets. (Cit. on p. 50).
- Divyanth L G. (2021). Cassava leaf disease dataset. <https://doi.org/10.17632/3832TX2CB2.1>. (Cit. on pp. 82, 85)
- Dong, X., Chen, S., & Pan, S. (2017). Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in Neural Information Processing Systems*, 4857–4867 (cit. on p. 37).
- Dong, X., Huang, J., Yang, Y., & Yan, S. (2017). More is less: A more complicated network with less inference complexity. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5840–5848 (cit. on p. 6).
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (cit. on p. 77).
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*. <https://openreview.net/forum?id=YicbFdNTTy> (cit. on p. 86)
- Draxler, F., Veschgini, K., Salmhofer, M., & Hamprecht, F. (2018). Essentially no barriers in neural network energy landscape. *International conference on machine learning*, 1309–1318 (cit. on pp. 51, 53).
- Duchi, J., Hashimoto, T., & Namkoong, H. (2020). Distributionally robust losses for latent covariate mixtures. (Cit. on p. 20).
- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7) (cit. on p. 1).
- Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., & Madry, A. (2019). Exploring the landscape of spatial robustness. *International conference on machine learning*, 1802–1811 (cit. on p. 77).

Bibliography

- Entezari, R., & Saukh, O. (2019). Class-dependent compression of deep neural networks. *to appear in Sensys-ML 2020, arXiv preprint arXiv:1909.10364* (cit. on pp. 4, 32, 37, 38).
- Entezari, R., Sedghi, H., Saukh, O., & Neyshabur, B. (2021). The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296* (cit. on pp. 11, 54–57, 59, 61, 62, 68).
- Entezari, R., Wortsman, M., Saukh, O., Shariatnia, M. M., Sedghi, H., & Schmidt, L. (2023). The role of pre-training data in transfer learning. *arXiv preprint arXiv:2302.13602* (cit. on pp. 82, 84–94).
- Evci, U., Gale, T., Menick, J., Castro, P. S., & Elsen, E. (2020). Rigging the lottery: Making all tickets winners. *International Conference on Machine Learning*, 2943–2952 (cit. on p. 8).
- Fang, A., Ilharco, G., Wortsman, M., Wan, Y., Shankar, V., Dave, A., & Schmidt, L. (2022). Data determines distributional robustness in contrastive language image pre-training (clip). *arXiv preprint arXiv:2205.01397* (cit. on pp. 78, 82, 88).
- Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8), 861–874 (cit. on p. 32).
- Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *2004 conference on computer vision and pattern recognition workshop*, 178–178 (cit. on pp. 82, 85, 87).
- Fix, E., & Hodges Jr, J. L. (1952). *Discriminatory analysis-nonparametric discrimination: Small sample performance* (tech. rep.). California Univ Berkeley. (Cit. on p. 28).
- Ford, N., Gilmer, J., Carlini, N., & Cubuk, D. (2019). Adversarial examples are a natural consequence of test error in noise. *arXiv preprint arXiv:1901.10513* (cit. on p. 13).
- Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412* (cit. on p. 9).
- Fort, S., Hu, H., & Lakshminarayanan, B. (2019). Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757* (cit. on p. 51).
- Fort, S., & Jastrzebski, S. (2019). Large scale structure of neural network loss landscapes. (Cit. on p. 53).
- Frankle, J., & Carbin, M. (2019a). The lottery ticket hypothesis: Finding sparse, trainable neural networks. (Cit. on p. 52).
- Frankle, J., & Carbin, M. (2019b). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)* (cit. on pp. 5, 6, 21, 32, 33).
- Frankle, J., Dziugaite, G. K., Roy, D., & Carbin, M. (2020). Linear mode connectivity and the lottery ticket hypothesis. *ICML*, 119, 3259–3269 (cit. on pp. 11, 12, 33, 52, 53).

Bibliography

- Frantar, E., & Alistarh, D. (2023). Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774* (cit. on p. 5).
- Freeman, C. D., & Bruna, J. (2016). Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540* (cit. on p. 51).
- Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. *icml*, 96, 148–156 (cit. on p. 9).
- Fukumizu, K., & Amari, S. (2000). Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13, 317–327 (cit. on p. 58).
- Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., Marten, R., Wortsman, M., Ghosh, D., Zhang, J., Orgad, E., Entezari, R., Daras, G., Pratt, S., Ramanujan, V., Bitton, Y., Marathe, K., Mussmann, S., Vencu, R., ... Schmidt, L. (2023). Datacomp: In search of the next generation of multimodal datasets. (Cit. on p. 97).
- Gale, T., Elsen, E., & Hooker, S. (2019). The state of sparsity in deep neural networks. (Cit. on p. 5).
- Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D., & Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of DNNs. (Cit. on pp. 10, 51, 53).
- Geiger, M., Spigler, S., d'Ascoli, S., Sagun, L., Baity-Jesi, M., Biroli, G., & Wyart, M. (2019). Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1), 012115 (cit. on p. 51).
- Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11), 665–673 (cit. on pp. 15, 16).
- Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2018). Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231* (cit. on p. 77).
- Ghosh, J., & Turner, K. (1994). Structural adaptation and generalization in supervised feedforward networks. *Journal of Artificial Neural Networks*, 1(4), 431–458 (cit. on p. 8).
- Golubeva, A., Neyshabur, B., & Gur-Ari, G. (2021). Are wider nets better given the same number of parameters? (Cit. on pp. 20–22).
- Gontijo-Lopes, R., Smullin, S. J., Cubuk, E. D., & Dyer, E. (2020). Affinity and diversity: Quantifying mechanisms of data augmentation. *arXiv preprint arXiv:2002.08973* (cit. on p. 14).
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. (Cit. on pp. 3, 12).
- Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (cit. on pp. 2, 21, 24).

Bibliography

- Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., & He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (cit. on p. 49).
- Graham, S., Vu, Q. D., Raza, S. E. A., Azam, A., Tsang, Y. W., Kwak, J. T., & Rajpoot, N. (2019). Hover-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical Image Analysis*, 58, 101563. <https://doi.org/10.1016/j.media.2019.101563> (cit. on p. 43)
- Gultekin, S., Saha, A., Ratnaparkhi, A., & Paisley, J. (2018). Mba: Mini-batch auc optimization. *arXiv preprint arXiv:1805.11221* (cit. on p. 34).
- Guo, Y., Yao, A., & Chen, Y. (2016). Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29 (cit. on p. 5).
- Gutman, D., Codella, N. C., Celebi, E., & et. al. (2016). Skin lesion analysis toward melanoma detection. *arXiv preprint arXiv:1605.01397* (cit. on p. 35).
- Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (cit. on p. 5).
- Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 1135–1143 (cit. on p. 5).
- Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10), 993–1001 (cit. on p. 8).
- Hasenfratz, D., Saukh, O., Walser, C., Hueglin, C., Fierz, M., & Thiele, L. (2014). Pushing the spatio-temporal resolution limit of urban air pollution maps. *Proceedings of the IEEE Conference on Pervasive Computing and Communications (PerCom)*, 69–77 (cit. on p. 32).
- Hassibi, B., & Stork, D. (1992). Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5 (cit. on p. 5).
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. (Cit. on pp. 22, 53).
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *CVPR*, 770–778 (cit. on pp. 76, 77, 84).
- He, X., Zhou, Z., & Thiele, L. (2018). Multi-task zipping via layer-wise neuron sharing. *Advances in Neural Information Processing Systems*, 6016–6026 (cit. on p. 62).
- He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4340–4349 (cit. on p. 6).

Bibliography

- Helber, P., Bischke, B., Dengel, A., & Borth, D. (2019). Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (cit. on pp. 82, 85).
- Hendricks, L. A., Mellor, J., Schneider, R., Alayrac, J.-B., & Nematzadeh, A. (2021). Decoupling the role of data, attention, and losses in multi-modal transformers. *Transactions of the Association for Computational Linguistics*, 9, 570–585 (cit. on pp. 79, 80).
- Hendricks, L. A., & Nematzadeh, A. (2021). Probing image-language transformers for verb understanding. *arXiv preprint arXiv:2106.09141* (cit. on pp. 80, 82).
- Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations* (cit. on pp. 21, 24).
- Hendrycks, D., & Dietterich, T. G. (2018). Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697* (cit. on p. 24).
- Hendrycks, D., & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (cit. on p. 2).
- Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., & Lakshminarayanan, B. (2019). Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781* (cit. on p. 76).
- Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., & Song, D. (2021). Natural adversarial examples. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15262–15271 (cit. on p. 21).
- Hochreiter, S., & Schmidhuber, J. (1997). Flat minima. *Neural Comput.*, 9(1), 1–42. <https://doi.org/10.1162/neco.1997.9.1.1> (cit. on p. 2)
- Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241), 1–124 (cit. on pp. 4–8).
- Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556* (cit. on p. 50).
- Hooker, S., Courville, A., Clark, G., Dauphin, Y., & Frome, A. (2019). What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248* (cit. on pp. 25–27, 29).
- Hooker, S., Courville, A., Clark, G., Dauphin, Y., & Frome, A. (2020). What do compressed deep neural networks forget? (Cit. on pp. 20–22).

Bibliography

- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., & et. al. (2018). MobileNets: Efficient convolutional neural networks for mobile vision applications. *CoRR, abs/1704.04861* (cit. on p. 37).
- Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (cit. on p. 15).
- Huang, C., Li, Y., Change Loy, C., & Tang, X. (2016). Learning deep representation for imbalanced classification. *CVPR*, 5375–5384 (cit. on pp. 34, 36).
- Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., & Schmidt, L. (2021). Openclip [If you use this software, please cite it as below.]. <https://doi.org/10.5281/zenodo.5143773> (cit. on p. 84)
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 448–456 (cit. on p. 70).
- Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407* (cit. on p. 70).
- Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2019). Averaging weights leads to wider optima and better generalization. (Cit. on p. 12).
- Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572* (cit. on p. 59).
- Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., & Bengio, S. (2019). Fantastic generalization measures and where to find them. (Cit. on p. 23).
- Jordan, K., Sedghi, H., Saukh, O., Entezari, R., & Neyshabur, B. (2022). Repair: Renormalizing permuted activations for interpolation repair. *arXiv preprint arXiv:2211.08403* (cit. on pp. 64, 65, 68–72).
- Jørgensen, A. S., Rasmussen, A. M., Andersen, N. K. M., Andersen, S. K., Emborg, J., Røge, R., & Østergaard, L. R. (2017). Using cell nuclei features to detect colon cancer tissue in hematoxylin and eosin stained slides. *Cytometry Part A*, 91(8), 785–793. <https://doi.org/10.1002/cyto.a.23175> (cit. on p. 39)
- Kalibhat, N. M., Narang, K., Tan, L., Firooz, H., Sanjabi, M., & Feizi, S. (2022). Understanding failure modes of self-supervised learning. *arXiv preprint arXiv:2203.01881* (cit. on pp. 26–28).
- Kang, M., & Han, B. (2020). Operation-aware soft channel pruning using differentiable masks. *International Conference on Machine Learning*, 5122–5131 (cit. on p. 6).

Bibliography

- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (cit. on p. 49).
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836* (cit. on pp. 3, 50).
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. (Cit. on p. 51).
- Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., & Krishnan, D. (2020). Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33, 18661–18673. <https://arxiv.org/abs/2004.11362> (cit. on p. 26)
- Kim, D., Wang, K., Sclaroff, S., & Saenko, K. (2022). A broad study of pre-training for domain generalization and adaptation. *arXiv preprint arXiv:2203.11819* (cit. on p. 83).
- Kim, H. (2020). Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950* (cit. on p. 21).
- Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (cit. on p. 1).
- Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., et al. (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123, 32–73 (cit. on pp. 79, 81).
- Krizhevsky, A., Nair, V., & Hinton, G. (2009). Cifar-100 and cifar-10 (canadian institute for advanced research) [MIT License]. <http://www.cs.toronto.edu/~kriz/cifar.html>. (Cit. on pp. 22, 27, 53, 82, 85)
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *NIPS*, 1097–1105 (cit. on pp. 1, 36).
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84–90 (cit. on pp. 15, 76, 77).
- Kuhn, H. W. (1955a). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2), 83–97 (cit. on p. 62).
- Kuhn, H. W. (1955b). The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52 (cit. on pp. 63, 64).
- Kumar, N., Verma, R., Anand, D., Zhou, Y., Onder, O. F., Tsougenis, E., Chen, H., Heng, P.-A., Li, J., Hu, Z., Wang, Y., Koohbanani, N. A., Jahanifar, M., Tajeddin, N. Z., Gooya, A., Rajpoot, N., Ren, X., Zhou, S., Wang, Q., ... Sethi, A. (2020). A multi-organ nucleus segmentation

Bibliography

- challenge. *IEEE Transactions on Medical Imaging*, 39(5), 1380–1391. <https://doi.org/10.1109/TMI.2019.2947628> (cit. on pp. 40, 43)
- Kurakin, A., Goodfellow, I., Bengio, S., et al. (2016). Adversarial examples in the physical world. (Cit. on pp. 21, 24).
- Kusupati, A., Ramanujan, V., Somani, R., Wortsman, M., Jain, P., Kakade, S., & Farhadi, A. (2020). Soft threshold weight reparameterization for learnable sparsity. *International Conference on Machine Learning*, 5544–5555 (cit. on p. 6).
- Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30 (cit. on p. 9).
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436–444 (cit. on pp. 1–3, 12).
- LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database [Creative Commons Attribution-Share Alike 3.0]. <http://yann.lecun.com/exdb/mnist/>. (Cit. on pp. 22, 53)
- LeCun, Y., Denker, J., & Solla, S. (1989). Optimal brain damage. *Advances in neural information processing systems*, 2 (cit. on p. 5).
- LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. *Advances in neural information processing systems*, 598–605 (cit. on p. 5).
- Lee, N., Ajanthan, T., & Torr, P. H. (2018). Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR), 2019* (cit. on p. 37).
- Li, D., Ding, T., & Sun, R. (2018). Over-parameterized deep neural networks have no strict local minima for any continuous activations. *arXiv preprint arXiv:1812.11039* (cit. on p. 51).
- Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2017). Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913* (cit. on pp. 51, 55).
- Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31 (cit. on p. 15).
- Li, Y., Yosinski, J., Clune, J., Lipson, H., & Hopcroft, J. (2015). Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543* (cit. on pp. 63–65).
- Liang, S., Sun, R., Li, Y., & Srikant, R. (2018). Understanding the loss surface of neural networks for binary classification. *International Conference on Machine Learning*, 2835–2843 (cit. on p. 10).
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, 740–755 (cit. on pp. 79, 81).

Bibliography

- Liu, C., Zhu, L., & Belkin, M. (2020). Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *arXiv preprint arXiv:2003.00307* (cit. on p. 51).
- Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2018). Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270* (cit. on p. 6).
- Lopes, R. G., Yin, D., Poole, B., Gilmer, J., & Cubuk, E. D. (2019). Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611* (cit. on p. 13).
- Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (cit. on p. 41).
- Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32 (cit. on p. 78).
- Lu, Z., Wu, X., Zhu, X., & Bongard, J. (2010). Ensemble pruning via individual contribution ordering. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 871–880 (cit. on p. 8).
- Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., & Wilson, A. G. (2019). A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32 (cit. on p. 9).
- Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2019). Towards deep learning models resistant to adversarial attacks. (Cit. on pp. 20, 21, 24).
- Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., & Van Der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. *Proceedings of the European conference on computer vision (ECCV)*, 181–196 (cit. on p. 76).
- Mahbod, A., Entezari, R., Ellinger, I., & Saukh, O. (2022). Deep neural network pruning for nuclei instance segmentation in hematoxylin and eosin-stained histological images. *Applications of Medical Artificial Intelligence: First International Workshop, AMAI 2022, Held in Conjunction with MICCAI 2022, Singapore, September 18, 2022, Proceedings*, 108–117 (cit. on pp. 40, 41, 44, 45).
- Mahbod, A., Schaefer, G., Bancher, B., Löw, C., Dorffner, G., Ecker, R., & Ellinger, I. (2021). CryoNuSeg: A dataset for nuclei instance segmentation of cryosectioned H&E-stained histological images. *Computers in Biology and Medicine*, 132, 104349. <https://doi.org/10.1016/j.combiomed.2021.104349> (cit. on p. 42)
- Mahbod, A., Schaefer, G., Ellinger, I., Ecker, R., Smedby, Ö., & Wang, C. (2019). A two-stage U-Net algorithm for segmentation of nuclei in H&E-stained tissues. *Digital Pathology*, 75–82. https://doi.org/10.1007/978-3-030-23937-4_9 (cit. on pp. 39, 41, 42)

Bibliography

- Mallya, A., & Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 7765–7773 (cit. on p. 6).
- Markuš, N. (2018). Fusing batch normalization and convolution in runtime [Accessed: 2022-09-28]. (Cit. on p. 69).
- McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (cit. on p. 28).
- McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29), 861. <https://doi.org/10.21105/joss.00861> (cit. on p. 26)
- Mei, S., Montanari, A., & Nguyen, P.-M. (2018). A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33), E7665–E7671 (cit. on p. 51).
- Miech, A., Alayrac, J.-B., Laptev, I., Sivic, J., & Zisserman, A. (2021). Thinking fast and slow: Efficient text-to-visual retrieval with transformers. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9826–9836 (cit. on p. 79).
- Miller, J. P., Taori, R., Raghunathan, A., Sagawa, S., Koh, P. W., Shankar, V., Liang, P., Carmon, Y., & Schmidt, L. (2021). Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization. *International Conference on Machine Learning*, 7721–7735 (cit. on p. 84).
- Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., Yu, C., & Micikevicius, P. (2021). Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378* (cit. on p. 6).
- Mu, N., & Gilmer, J. (2019). Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337* (cit. on pp. 21, 24).
- Mustafa, B., Riquelme, C., Puigcerver, J., Jenatton, R., & Houlsby, N. (2022). Multimodal contrastive learning with limoe: The language-image mixture of experts. *arXiv preprint arXiv:2206.02770* (cit. on p. 7).
- Nagarajan, V., & Kolter, J. Z. (2019). Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32 (cit. on p. 52).
- Nakkiran, P., Neyshabur, B., & Sedghi, H. (2020). The deep bootstrap framework: Good online learners are good offline generalizers. <https://doi.org/10.48550/ARXIV.2010.08127>. (Cit. on p. 25)
- Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2019). Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292* (cit. on pp. 49, 54, 55).
- Naylor, P., Laé, M., Reyal, F., & Walter, T. (2019). Segmentation of nuclei in histopathology images by deep regression of the distance map. *IEEE*

Bibliography

- Transactions on Medical Imaging*, 38(2), 448–459. <https://doi.org/10.1109/TMI.2018.2865709> (cit. on p. 40)
- Nematzadeh, A. (2021). *Aida nematzadeh - invited talk at the vqa workshop 2021. CVPR*. https://www.youtube.com/watch?v=o6fSUQw_RCQ&ab_channel=MLPLab. (Cit. on p. 78)
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning [License: CCo: Public Domain]. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. <http://ufldl.stanford.edu/housenumbers> (cit. on p. 53)
- Neyshabur, B. (2020a). Towards learning convolutions from scratch. *Advances in Neural Information Processing Systems* (cit. on pp. 53, 55).
- Neyshabur, B. (2020b). Towards learning convolutions from scratch. *arXiv preprint arXiv:2007.13657* (cit. on p. 21).
- Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). Exploring generalization in deep learning. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 5949–5958 (cit. on pp. 1, 51, 56).
- Neyshabur, B., Salakhutdinov, R., & Srebro, N. (2015). Path-SGD: Path-normalized optimization in deep neural networks. *arXiv preprint arXiv:1506.02617* (cit. on pp. 51, 57).
- Neyshabur, B., Sedghi, H., & Zhang, C. (2020a). What is being transferred in transfer learning? *NeurIPS* (cit. on pp. 10, 50–52).
- Neyshabur, B., Sedghi, H., & Zhang, C. (2020b). What is being transferred in transfer learning? *Advances in neural information processing systems*, 33, 512–523 (cit. on pp. 10–12, 15).
- Neyshabur, B., Tomioka, R., & Srebro, N. (2014). In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614* (cit. on pp. 1, 49).
- Ngiam, J., Chen, Z., Chia, D., Koh, P., Le, Q., & Ng, A. (2010). Tiled convolutional neural networks. *Advances in neural information processing systems*, 23 (cit. on p. 4).
- Nguyen, C., Hassner, T., Seeger, M., & Archambeau, C. (2020). Leep: A new measure to evaluate transferability of learned representations. *International Conference on Machine Learning*, 7294–7305 (cit. on pp. 82, 83).
- Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., & Poor, H. V. (2021). Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3), 1622–1658 (cit. on p. 13).
- Nguyen, Q., Mukkamala, M. C., & Hein, M. (2018). On the loss landscape of a class of deep neural networks with no bad local valleys. *arXiv preprint arXiv:1809.10749* (cit. on p. 51).

Bibliography

- Nikdan, M., Pegolotti, T., Iofinova, E., Kurtic, E., & Alistarh, D. (2023). Sparseprop: Efficient sparse backpropagation for faster training of neural networks. *arXiv preprint arXiv:2302.04852* (cit. on p. 5).
- Novac, P.-E., Hacene, G. B., Pegatoquet, A., Miramond, B., & Gripon, V. (2021). Quantization and deployment of deep neural networks on microcontrollers. *Sensors*, 21(9), 2984 (cit. on p. 21).
- Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., & Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: An empirical study. *arXiv preprint arXiv:1802.08760* (cit. on p. 49).
- Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23), 3311–3325 (cit. on p. 5).
- Ordonez, V., Kulkarni, G., & Berg, T. (2011). Im2text: Describing images using 1 million captioned photographs. *Advances in neural information processing systems*, 24 (cit. on pp. 79, 81).
- Ortega, L. A., Cabañas, R., & Masegosa, A. (2022). Diversity and generalization in neural network ensembles. *International Conference on Artificial Intelligence and Statistics*, 11720–11743 (cit. on p. 9).
- Özgenel, Ç. (2017). Concrete crack images for classification. *Mendeley Data*, v1 (cit. on p. 35).
- Paganini, M., & Forde, J. (2020). Streamlining tensor and network pruning in pytorch. *arXiv preprint arXiv:2004.13770* (cit. on p. 26).
- Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779* (cit. on p. 13).
- Parkhi, O. M., Vedaldi, A., Zisserman, A., & Jawahar, C. (2012). Cats and dogs. *2012 IEEE conference on computer vision and pattern recognition*, 3498–3505 (cit. on pp. 82, 85).
- Paul, M., Chen, F., Larsen, B. W., Frankle, J., Ganguli, S., & Dziugaite, G. K. (2022). Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask? *arXiv preprint arXiv:2210.03044* (cit. on p. 42).
- Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., & Wang, B. (2019). Moment matching for multi-source domain adaptation. *Proceedings of the IEEE/CVF international conference on computer vision*, 1406–1415 (cit. on pp. 82, 85).
- Pittorino, F., Lucibello, C., Feinauer, C., Malatesta, E. M., Perugini, G., Baldassi, C., Negri, M., Demyanenko, E., & Zecchina, R. (2020). Entropic gradient descent algorithms and wide flat minima. *arXiv preprint arXiv:2006.07897* (cit. on p. 23).
- Qu, Z. (2022). Enabling deep learning on edge devices. *arXiv preprint arXiv:2210.03204* (cit. on p. 6).

- Qu, Z., Zhou, Z., Tong, Y., & Thiele, L. (2022). P-meta: Towards on-device deep model adaptation. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1441–1451 (cit. on p. 6).
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *International Conference on Machine Learning*, 8748–8763 (cit. on pp. 16, 77, 78, 82, 84).
- Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training (cit. on p. 1).
- Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. (2021). Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446* (cit. on p. 50).
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1), 5485–5551 (cit. on p. 4).
- Raskutti, G., Wainwright, M. J., & Yu, B. (2014). Early stopping and non-parametric regression: An optimal data-dependent stopping rule. *The Journal of Machine Learning Research*, 15(1), 335–366 (cit. on p. 2).
- Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2019). Do imagenet classifiers generalize to imagenet? *International Conference on Machine Learning*, 5389–5400 (cit. on pp. 20, 75, 76).
- Renda, A., Frankle, J., & Carbin, M. (2020). Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389* (cit. on pp. 21, 42, 44).
- Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407 (cit. on p. 1).
- Rosenblatt, F. (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms* (tech. rep.). Cornell Aeronautical Lab Inc Buffalo NY. (Cit. on p. 53).
- Rosset, C. (2020). Turing-nlg: A 17-billion-parameter language model by microsoft. (Cit. on p. 4).
- Sagawa, S., Koh, P. W., Hashimoto, T. B., & Liang, P. (2020). Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. (Cit. on p. 20).
- Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., & Yang, G. (2019). Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*, 32 (cit. on pp. 76, 77).

Bibliography

- Santurkar, S., Dubois, Y., Taori, R., Liang, P., & Hashimoto, T. (2022). Is a caption worth a thousand images? a controlled study for representation learning. *arXiv preprint arXiv:2207.07635* (cit. on p. 92).
- Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. (2022). Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402* (cit. on pp. 15, 79, 81, 82, 84).
- Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., & Komatsuzaki, A. (2021). Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114* (cit. on p. 15).
- Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., & Goldstein, T. (2019). Adversarial training for free! *Advances in Neural Information Processing Systems*, 32 (cit. on pp. 76, 77).
- Shafeezadeh-Abadeh, S., Esfahani, P. M., & Kuhn, D. (2015). Distributionally robust logistic regression. (Cit. on p. 20).
- Shankar, V., Roelofs, R., Mania, H., Fang, A., Recht, B., & Schmidt, L. (2020). Evaluating machine accuracy on imagenet. *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 119, 8634–8644. <http://proceedings.mlr.press/v119/shankar20c.html> (cit. on p. 20)
- Sharma, P., Ding, N., Goodman, S., & Soriceut, R. (2018). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2556–2565 (cit. on pp. 79, 81, 82, 84).
- Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (cit. on p. 4).
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1), 1–48 (cit. on p. 2).
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015 (cit. on pp. 22, 53, 65, 76, 77).
- Şimşek, B., Ged, F., Jacot, A., Spadaro, F., Hongler, C., Gerstner, W., & Brea, J. (2021). Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. *arXiv preprint arXiv:2105.12221* (cit. on p. 58).

Bibliography

- Singh, S. P., & Jaggi, M. (2020). Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33, 22045–22055 (cit. on pp. 62, 65, 70, 71).
- Sirin, K., Raza, S. E. A., Tsang, Y.-W., Snead, D. R., Cree, I. A., & Rajpoot, N. M. (2016). Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Transaction on Medical Imaging*, 35(5), 1196–1206. <https://doi.org/10.1109/TMI.2016.2525803> (cit. on p. 39)
- Skinner, B. M., & Johnson, E. E. (2017). Nuclear morphologies: Their diversity and functional relevance. *Chromosoma*, 126(2), 195–212. <https://doi.org/10.1007/s00412-016-0614-5> (cit. on p. 39)
- Somauroo, D. J. (2019). Butterfly, a 1.25 billion healthtech company, launches new ultrasound technology in the u.k. <https://www.forbes.com/sites/jamessomauroo/2019/09/27/butterfly-a-125-billion-healthtech-company-launches-new-ultrasound-technology-in-the-uk/>. (Cit. on p. 39)
- Srinivasan, K., Raman, K., Chen, J., Bendersky, M., & Najork, M. (2021). Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2443–2449 (cit. on pp. 79, 82, 84).
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929–1958 (cit. on pp. 2, 9).
- Srivastava, N., & Salakhutdinov, R. R. (2012). Multimodal learning with deep boltzmann machines. *Advances in neural information processing systems*, 25 (cit. on p. 78).
- Steck, H. (2007). Hinge rank loss and the area under the roc curve. *ECML*, 347–358 (cit. on pp. 33–35).
- Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. *Proceedings of the IEEE international conference on computer vision*, 843–852 (cit. on pp. 12, 76, 83).
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9 (cit. on pp. 76, 77).
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Conference on Computer Vision and Pattern Recognition*, 2818–2826 (cit. on p. 12).

Bibliography

- Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946* (cit. on pp. 76, 77).
- Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., & Schmidt, L. (2020). Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33, 18583–18599 (cit. on pp. 76, 77, 84).
- Taturo, N. J., Chen, P.-Y., Das, P., Melnyk, I., Sattigeri, P., & Lai, R. (2020). Optimizing mode connectivity via neuron alignment. *arXiv preprint arXiv:2009.02439* (cit. on pp. 58, 63).
- Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2022). Efficient transformers: A survey. *ACM Computing Surveys*, 55(6), 1–28 (cit. on p. 50).
- Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., & Li, L.-J. (2016). Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2), 64–73 (cit. on pp. 81, 82, 84).
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267–288 (cit. on p. 5).
- Tieleman, T., & Hinton, G. (2012). Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *COURSERA Neural Networks Mach. Learn*, 17 (cit. on p. 1).
- Timpl, L., Entezari, R., Sedghi, H., Neyshabur, B., & Saukh, O. (2021). Understanding the effect of sparsity on neural networks robustness. *ICML 2021 Workshop Overparameterization: Pitfalls & Opportunities* (cit. on pp. 23–25).
- Torralba, A., & Efros, A. A. (2011). Unbiased look at dataset bias. *CVPR 2011*, 1521–1528 (cit. on p. 13).
- Tuli, S., Basumatary, N., & Buyya, R. (2019). Edgelens: Deep learning based object detection in integrated iot, fog and cloud computing environments. *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, 496–502 (cit. on p. 13).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30 (cit. on p. 1).
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2> (cit. on p. 62)

Bibliography

- von Oswald, J., Kobayashi, S., Sacramento, J., Meulemans, A., Henning, C., & Grewe, B. F. (2021). Neural networks with late-phase weights. (Cit. on p. 21).
- Wang, D., Khosla, A., Gargeya, R., Irshad, H., & Beck, A. H. (2016). Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718* (cit. on p. 13).
- Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (cit. on p. 50).
- Wang, Y.-X., Ramanan, D., & Hebert, M. (2017). Learning to model the tail. *NIPS*, 7029–7039 (cit. on pp. 32, 34).
- Wen, Y., Tran, D., & Ba, J. (2020). Batchensemble: An alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715* (cit. on p. 9).
- Wenzel, F., Snoek, J., Tran, D., & Jenatton, R. (2020). Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33, 6514–6527 (cit. on p. 9).
- Wiggers, K. (2020). Openai’s massive gpt-3 model [Accessed: 2023-03-13]. (Cit. on p. 4).
- Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2), 241–259 (cit. on p. 9).
- Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. (2022). Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *International Conference on Machine Learning*, 23965–23998 (cit. on p. 9).
- Wortsman, M., Ilharco, G., Li, M., Kim, J. W., Hajishirzi, H., Farhadi, A., Namkoong, H., & Schmidt, L. (2021). Robust fine-tuning of zero-shot models. *arXiv preprint arXiv:2109.01903* (cit. on p. 84).
- Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., & Farhadi, A. (2020). Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33, 15173–15184 (cit. on p. 6).
- Wu, W. (n.d.). Classifying images into 11k classes with pretrained model, 2016. URL <https://github.com/tornadomeet/ResNet>, https://github.com/aws-labs/deep-learning-benchmark/blob/master/image_classification/common/modelzoo.py L, 41 (cit. on p. 76).
- Wu, X., Dobriban, E., Ren, T., Wu, S., Li, Z., Gunasekar, S., Ward, R., & Liu, Q. (2019). Implicit regularization and convergence for weight normalization. *arXiv preprint arXiv:1911.07956* (cit. on p. 57).
- Xian, Y., Schiele, B., & Akata, Z. (2017). Zero-shot learning-the good, the bad and the ugly. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4582–4591 (cit. on p. 16).

Bibliography

- Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., & Le, Q. V. (2020). Adversarial examples improve image recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 819–828 (cit. on p. 77).
- Xie, C., Wu, Y., Maaten, L. v. d., Yuille, A. L., & He, K. (2019). Feature denoising for improving adversarial robustness. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 501–509 (cit. on pp. 76, 77).
- Xie, Q., Luong, M.-T., Hovy, E., & Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10687–10698 (cit. on p. 76).
- Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., & Mahajan, D. (2019). Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546* (cit. on p. 76).
- Yan, L., Dodier, R. H., Mozer, M., & Wolniewicz, R. H. (2003). Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. *ICML*, 848–855 (cit. on p. 34).
- Yang, X., Li, H., & Zhou, X. (2006). Nuclei segmentation using marker-controlled watershed, tracking using mean-shift, and Kalman filter in time-lapse microscopy. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(11), 2405–2414. <https://doi.org/10.1109/TCSI.2006.884469> (cit. on p. 43)
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27 (cit. on p. 15).
- You, K., Liu, Y., Wang, J., & Long, M. (2021). Logme: Practical assessment of pre-trained models for transfer learning. *International Conference on Machine Learning*, 12133–12143 (cit. on pp. 82, 83).
- Young, P., Lai, A., Hodosh, M., & Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2, 67–78 (cit. on p. 80).
- Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. *Proceedings of the IEEE/CVF international conference on computer vision*, 6023–6032 (cit. on p. 76).
- Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146* (cit. on p. 27).
- Zhan, S.-h., Lin, J., Zhang, Z.-j., & Zhong, Y.-w. (2016). List-based simulated annealing algorithm for traveling salesman problem. *Intell. Neuro-science*, 2016, 8 (cit. on pp. 60, 61).

Bibliography

- Zhang, C., & Ma, Y. (2012). *Ensemble machine learning: Methods and applications*. Springer. (Cit. on p. 9).
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. *Proceedings of the International Conference on Learning Representations* (cit. on pp. 1, 4, 12, 51).
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3), 107–115 (cit. on p. 49).
- Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (cit. on pp. 13, 76).
- Zhang, H., Dauphin, Y., & Ma, T. (2019). Residual learning without normalization via better initialization. *Proceedings of the International Conference on Learning Representations* (cit. on p. 70).
- Zhang, R. (2019). Making convolutional networks shift-invariant again. *International conference on machine learning*, 7324–7334 (cit. on p. 76).
- Zhang, R., Li, C., Zhang, J., Chen, C., & Wilson, A. G. (2019). Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932* (cit. on p. 9).
- Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020). Random erasing data augmentation. *Proceedings of the AAAI conference on artificial intelligence*, 34(07), 13001–13008 (cit. on p. 13).
- Zhou, A., Ma, Y., Zhu, J., Liu, J., Zhang, Z., Yuan, K., Sun, W., & Li, H. (2021). Learning n: M fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010* (cit. on p. 6).
- Zhu, M., & Gupta, S. (2017). To prune, or not to prune: Exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878* (cit. on pp. 8, 26–28).