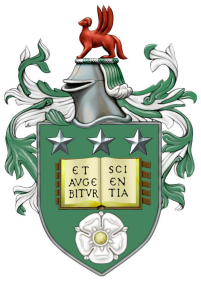
Impact of Local and Spatial Redundancy on Prunability of Convolutional Neural Networks and their Training Dynamics



Luis Alfredo Avendaño Muñoz

School of Computer Science

University of Leeds

A thesis submitted for the degree of

PhD in Computer Science

September 2025

Intellectual Property Statement

I confirm that the work submitted is my own, except where work which has formed part of jointly authored publications has been included. My contribution and the other authors' contributions to this work have been explicitly indicated below. I confirm that appropriate credit has been given within the thesis where reference has been made to the work of others.
This copy has been supplied on the understanding that it is copyright material, and that no quotation from the thesis may be published without proper acknowledgement.

The right of Luis Alfredo Avendaño Muñoz to be identified as the Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

The joint publications present in this thesis can be found in the following peer-reviewed conference, and under review journal paper:

- Chapter 3 Avendaño Muñoz. L. A., N. Cohen, N. Omidvar "Stochastic Pruning on Neural Networks" July 2025 International Joint Conference on Neural Networks (IJCNN), Rome, Italy, 2025.

- Chapter 4 (Submitted) Avendaño Muñoz. L. A., N. Cohen, N. Omidvar "Larger Receptive Fields Improve Pruning Performance in CNNs" *Transactions on Neural Networks and Learning Systems.*

For both publications, I took the lead on drafting the manuscript and performed the experiments and data visualisation. Meanwhile, Netta Cohen and Nabi Omidvar contributed valuable insights into the experimental concepts and offered editorial feedback on the manuscripts.

Acknowledgements

First and foremost, I would like to thank my supervisors, Dr. Nabi Omidvar and Professor Neta Cohen at the University of Leeds for their invaluable support and guidance throughout my PhD. From how to do research to encouraging me to trust my intuition, Nabi has been an invaluable mentor. Netta, with her eye for details and a seasoned experience, taught me what good research looks like and how to think like a scientist. I would like to thank my parents, who without their unconditional support and love, my PhD would not have been possible. I would like to thank my wife, Sherine, whose patience and love were essential during the most difficult times during my PhD. And finally, I would like to thank my colleagues and friends, Pedro Chavarrias e Ignacio Bellas for the fruitful discussions, academic and personal. This work was undertaken on the ARC3/ARC4 (both now decommissioned) and AIRE High-Performance Computing facilities at the University of Leeds, UK. I acknowledge the use of ChatGPT-4 (Open AI, https://chat.openai.com/) and Sonnet 4 (Antrophic, https://claude.ai/) to proofread my final draft. I would like to thank the authors Ronny Huang, Hussein Mobahi, Justin Shenk, Hao Li, Shankar Krishnan, Utku Evci and James Martens for granting me permission to use their figures in this work.

Abstract

Convolutional Neural Networks (CNNs) are widely used in computer vision, but despite their efficient use of weights via weight sharing, approaches to compressing CNNs remain a significant challenge. Among various efficiency-improving techniques, pruning has been extensively studied as it enables model compression with a controlled reduction in accuracy. This thesis explores pruning from 3 perspectives: the local neighbourhood of a solution, the receptive field, and the interplay between training dynamics and prunability.

Exploring the neighbourhood of a solution revealed that applying noise to dense CNNs before pruning, uncovers better pruned models than pruning the original model alone. These improved pruned models are found at extreme pruning rates (greater than 0.8), and their improvements are present with and without further fine-tuning

Results show that adding noise to a CNN model mitigates a detrimental phenomenon in which the variance discrepancy among network features grows dramatically during pruning, thereby disrupting internal inference dynamics. Next, this thesis investigates how receptive field size (i.e., the area of the input image that a neuron "attends" to) affects the pruning performance of CNNs trained for visual classification tasks. The findings reveal that pruning's impact on accuracy varies systematically across receptive fields. Surprisingly, larger receptive fields tend to mitigate pruning-induced accuracy loss, and this effect often persists even after fine-tuning. Although dense networks achieve peak accuracy with smaller receptive fields, the optimal receptive field for pruned networks is typically larger. This phenomenon can be attributed to highly redundant deep layers that exhibit low feature

saturation. This results in minimal impact on classification accuracy in the deep layers. Finally, this thesis explores how both a model's receptive field and the choice of optimiser influence the geometry of the loss landscape and, ultimately, the extent to which the model can be effectively pruned. Results show that the ruggedness of the loss landscape is directly related to the receptive field, making landscapes associated with larger receptive fields more challenging for Stochastic Gradient Descent (SGD) to navigate. Models trained with second-order optimisers are significantly less prunable than those trained with first-order or flatness-promoting optimisation methods. Furthermore, it is possible to manipulate the loss landscape (e.g., by increasing model width) so that SGD can partially circumvent the limitations imposed by the ruggedness of larger receptive fields. Additionally, the results suggest that the reduced prunability of models trained with second-order optimisers is attributable to the sharpness and filter diversity of the learned solutions.

Lastly, results show that increasing receptive fields recovers the pruned accuracy of models trained with second-order optimisers.

Abbreviations

DP	Deterministic Pruning
SP	Stochastic Pruning
GMP	Global Magnitude Pruning
LAMP	Layer-Adaptive Magnitud Pruning
GRASP	Gradient Signal Preservation
RF	Rceptive field
MOO	Multi-Objective Optimisation
GF	Gradient Flow
CNN	Convolutional Neural Network
γ	Pruning Rate
σ	Noise level
λ	Hessian Eigenvalue
ASAM	Adaptive sharpness-Aware Minimisation
EKFAC	Eigenvalue-corrected Kronecker factorization
SGD	Stochastic Gradient Descent
A_{dense}	Accuracy of Dense Network (Not pruned)
A_{OS}	Accuracy of One-Shot Pruned Network
A_{FT}	Accuracy of Fine-Tuned Pruned Network
Δ_{OS}	A_{Dense} - A_{OS}
Δ_{FT}	A_{Dense} - A_{TF}

Contents

Chapter 1

Introduction

Deep Neural Networks have emerged as the engine of breakthroughs in several machine learning applications ranging from image classification (Tan & Le, 2019), image segmentation (Rizwan I Haque & Neubert, 2020), natural language processing (Devlin *et al.*, 2019), to text-to-speech generation (van den Oord *et al.*, 2016), and even continuous control problems (Duan *et al.*, 2016). Advances in reinforcement learning, as a method of training neural networks, have facilitated deep network based solutions to a much richer and more challenging set of applications, from games such as Go (Silver *et al.*, 2016) and StarCraft II (Vinyals *et al.*, 2019) to scientific and industrial applications such as protein folding (Senior *et al.*, 2020).

These deep learning models have achieved considerably better performances than those obtained by other methods not based on neural networks. Hence, its current popularity among the machine learning community and the general public in recent years (OpenAI *et al.*, 2024).

The characteristics that make deep neural networks so widely adopted are:

- **Flexibility/Expressiveness**: Neural networks can solve many different problems thanks to their property of universal approximation. This property has been formally proven for deep neural networks with sufficient layers and the right activation functions (like ReLU). However, these theoretical proofs typically assume unlimited resources, infinite data, infinite width, or infinite depth (Bengio & Delalleau, 2011; Cybenko, 1989; Mhaskar & Poggio, 2016;

Mhaskar, 1993). In practice, modern machine learning has demonstrated that infinite resources are not necessary. With finite but sufficient data and appropriately sized networks, it is possible to learn a wide variety of functions effectively.

- **Scalability**: The expressive power of any given neural network can be easily augmented by simply adding more hidden layers/parameters. Unlike other machine learning models, the performance of neural networks does not stagnate when plenty of data is available (Bottou *et al.*, 2018). This is also why neural networks are ubiquitous in large-scale industrial applications such as digital assistants and text-to-speech applications.

The process of learning representations using deep neural networks is known as deep learning (Goodfellow *et al.*, 2016). The capability of deep learning to generate hierarchical representations by increasing the depth of the network has elevated it to a prominent position within the field of machine learning. Modern state-of-the-art deep learning algorithms rely on incredibly large models to achieve competitive performance. These large models were enabled due to the increased availability of training data and the increase in computational power. More training data means that the difficulty of statistical estimation is reduced (Goodfellow *et al.*, 2016). This allows neural networks to perform better without a substantial change in their training procedure. And more computational power (in the form of specialised hardware, i.e. GPUs, more memory, and access to cloud computing) means that training could happen faster and at the same time bigger models could be stored and executed within reasonable time. Modern state-of-the-art models often range from tens of millions to hundreds of billions of parameters, and in the most extreme cases beyond 1.5 trillion parameters (Fedus *et al.*, 2022). Having so many parameters makes these models extremely resource and energy intensive not only for training but also for deployment/inference (Thompson *et al.*, 2023). Moreover, these large models tend to overfit the data and do not generalise well on new data (Garbin *et al.*, 2020). For this reason, these models must be paired with training procedures, such as data augmentation or some form of regularisation, such as Dropout (Srivastava *et al.*, 2014), to ensure generalisation. Another

important aspect for generalisation in neural networks is the optimisation procedure. For example, Keskar *et al.* (2017) empirically demonstrated that using large batches for training neural networks leads to sharp minima of the loss function, which are known to have poor generalisation (Hochreiter & Schmidhuber, 1997).

Stochastic Gradient Decent (SGD) (Robbins & Monro, 1951) paired with Backpropagation (Rumelhart *et al.*, 1986) is the standard optimiser in deep learning. The optimisation landscape of deep neural networks is known to be highly non-convex, so why is SGD successful in optimising these models? Some argue that the reason for the success of SGD is that as neural networks grow in their number of parameters, more local minima with good generalisation properties (i.e., low training and test error) emerge in the loss landscape (Choromanska *et al.*, 2015; He *et al.*, 2020; Li *et al.*, 2018a; Nguyen *et al.*, 2018). This suggests that over-parameterisation in state-of-the-art models is used as a way to cope with the limitations of SGD as an optimiser. Additional dimensions increase the volume of high performing solutions, which increases the likelihood that SGD will identify these optima (Huang *et al.*, 2020).

Some of the implications of these large models that are often overlooked are their energy consumption and cost. Thompson *et al.* (2023) present a detailed analysis of the computational requirements of deep learning in practice, past and present. It is clear that deep learning is resource-intensive by design. In fact, in the past, the field was abandoned because the computational resources needed were so vast, until 2012, when Krizhevsky *et al.* (2012) developed AlexNet, a Convolutional Neural Network (CNN) that won the 2012 ImageNet Challenge with substantial difference in performance at the time. This seminal work reignited the interest of researchers in convolutional neural networks and deep learning as a whole. Between 2012 and 2019, the growth in computational requirements for achieving state-of-the-art performance grew ten times each year. Taking into account the 35 fold improvement in computational power due to the use of specialised hardware (Thompson *et al.*, 2023) (i.e. GPUs and TPUs) and the 44 fold improvement due to algorithmic progress (Hernandez & Brown, 2020) over the same period, it still does not account for the needed growth in computational requirements.

Recently, some researchers have advocated for a more holistic approach to deep learning research. Schwartz *et al.* (2019) argue that the focus of the publication

outlets on state-of-the-art performance has led the deep learning community to under-report the real cost of obtaining such high-performing methods. Everything from trial runs for hyperparameter tuning, to unsupervised pre-training on massive amounts of data are common practices that are typically not reported in the final papers. The authors propose to report at least two out of five of the following efficiency metrics: carbon emissions, electricity usage, elapsed wall-clock time, number of parameters, and total number of floating-point operations. In this way, the authors hoped that research focused on efficiency instead of state-of-the-art performance would be encouraged, pushing deep learning to be more sustainable. It is important to note that efficiency refers to the amount of computational resources used to achieve a desired level of performance. Then more efficient methods achieve more performance per unit of computation.

According to Thompson *et al.* (2023), there are three key areas that can enhance the cost-effectiveness of deep learning training: increasing computational capacity through hardware accelerators, decreasing computational complexity through network compression and acceleration, and discovering compact high-performance deep learning architectures using neural architecture search and meta-learning.

This thesis focuses on the second area, specifically neural network compression. Although Thompson *et al.* (2023) addresses training efficiency, which is straightforward to quantify and more frequently documented, this thesis will investigate neural network compression through pruning following the training process, known as post-training pruning, where efficiency improvements manifest during the inference phase.

The neural network pruning literature has shown that even a 90% reduction in the number of weights with post-training pruning allows the pruned network to achieve performance comparable to that of the unpruned network (Wang *et al.*, 2019a). Moreover, in (Frankle & Carbin, 2018), the authors were able to find subnetworks that perform better than the original networks.

Research shows that pruning neural networks before training cannot match the performance of pruning after training, particularly when connectivity patterns remain fixed during training (Frankle *et al.*, 2020b). Since trained large models can be heavily pruned while maintaining performance, this suggests that high-dimensional parameter spaces facilitate discovering solutions that can later

be compressed.While compressed solutions can generalise well, they can only be obtained by first training in high-dimensional parameter spaces. This suggests a paradox: high-dimensionality appears unnecessary for the final solution but essential for discovering it.

This begs the question, how are the loss landscape and compression/prunability of neural networks related? Is it possible to devise mechanisms that exploit the loss landscape to prune neural networks more effectively? What role plays the optimisation process in the prunability of models?

Finally, this study focuses on convolutional neural networks in vision tasks, as CNNs continue to play an important role in computer vision research and practice despite the rapid rise of vision transformers (ViTs) (Dosovitskiy *et al.*, 2022) as state-of-the-art models on many benchmarks. The global aggregation of information characteristic of ViTs has been useful in image classification, but they did not become a generic vision task until they incorporated some ideas from convolutional networks into the Swin Transformer (Liu *et al.*, 2021) such as limited self-attention computation to non-overlapping local windows and cross-window connections.

Unlike CNNs, ViTs require large amounts of data, generally for pre-training, to achieve high performance (Touvron *et al.*, 2021). Consequently, in domains such as medical imaging, where data is less abundant, CNNs still outperform ViTs in classifications of skin lesions and melanoma (Kawadkar, 2025). The local receptive field (the region of input space to which a particular neuron or unit responds) of CNNs is pivotal for their success, but is it possible to explore this spatial locality to further increase the compressibility and efficiency of CNNs?

1.1 Aims and Contributions of this Thesis

This thesis explores three aspects involved in the performance and efficiency (both at training and inference) of CNNs, the loss landscape, the spatial locality (local receptive field) and the optimisation process. Related to each of these three aspects, this thesis sets out to answer the following questions.

- Can one derive well-performing sparse networks by examining the basin of attraction of a minimiser (the set of all initial parameter configurations that,

under a given optimisation procedure, converge to the same minimiser)?

- In what manner do architectural elements, specifically the receptive field in convolutional networks, affect the performance of both dense and sparse networks?
- In what way do the dynamics of training influence the eventual compressibility of a neural network, and in particular the choice of optimiser?

This thesis addresses these questions through a series of systematic empirical studies. The contributions in this thesis are separated into three interrelated themes.

- **Stochastic Pruning for Neural Networks:**

  In order to explore the basin of attraction of a minimiser in search of well-performing sparse networks, additive Gaussian noise is added to a trained dense network before pruning. This two-step pruning method is referred to as Stochastic Pruning (SP) in this thesis. The results demonstrate that SP often outperforms traditional deterministic pruning at high pruning rates and appropriate noise levels, and that these advantages persist even after fine-tuning. Furthermore, this thesis examines the mechanisms underlying SP's efficacy, including its mitigation of the discrepancy in feature variance across layers, referred to here as feature variance explosion, in pruned networks. These findings have direct implications for applications where retraining is costly or infeasible.

- **Receptive Field Size and Pruning Performance in CNNs:**

  The experiments systematically explore the role of the receptive field in determining the prunability of convolutional neural networks (CNNs). The experiments reveal that increasing the receptive field size can substantially improve pruned model accuracy, often by inducing redundancy in deeper layers and thereby making them more amenable to aggressive pruning. The results indicate that this improvement in the pruned performance is attributable to the low saturation of the deeper layers. As the receptive field increases,

deeper layers become increasingly redundant, thereby improving prunability. Furthermore, results suggest that this redundancy in the deeper layer is evident in the representations of the ResNet architecture (He *et al.*, 2015).

- **Optimisation, Loss Landscape and Prunability:** Finally, to study how optimisation dynamics interact with prunability, the effect of manipulating the receptive field in the loss landscape is explored. The results show that larger receptive fields lead to a more rugged loss surface, complicating the optimisation. Subsequently, to smooth the loss landscape and facilitate optimisation, the model width was increased.

  With sufficient width, it was possible to match the performance of a model trained with a small receptive field and a small width to that of a model trained with a large receptive field and a large width, suggesting that it is possible to smooth the landscape.

  Then, instead of changing the landscape, advanced optimisers were used to test if they could better adapt to the rugged landscape. Curvature-aware optimisers (such as second-order (George *et al.*, 2018) and sharpness-minimisation (Kwon *et al.*, 2021)) were employed for this purpose.

  Interestingly, although curvature-aware optimisers are theoretically well-suited to such landscapes, they do not necessarily improve final accuracy relative to standard SGD. Furthermore, the results suggest that, for the second-order optimiser, the solution it finds lies in a shallower valley of the loss landscape than those found by SGD and the sharpness-aware optimiser. This also leads to reduced prunability, as sharp solutions are more susceptible to weight-space perturbations (e.g., pruning). Additionally, the results show that Second-order optimiser solutions exhibit higher feature saturation than SGD and sharpness-aware minimisation, making their representations less redundant. Finally, increasing the receptive field reduces feature saturation in second-order models, thereby increasing their prunability.

By integrating algorithmic, architectural, and optimisation perspectives, this thesis aims to advance the understanding of what makes neural networks prunable.

A limitation of this work is that, although the questions addressed here are fundamental, the approach employed is experimental. The focus of this thesis is on a particular kind of model and task, which limits the generalisability of the results presented. The remainder of the thesis is structured as follows: Chapter 2 reviews key background and related work on loss landscape for sparse and dense models, neural network pruning, and receptive field. Chapter 3 introduces and analyses the proposed stochastic pruning method. Chapter 4 investigates the influence of the size of the receptive field on the pruning outcomes in CNNs. Chapter 5 explores the interaction between the optimiser choice, the loss landscape geometry and prunability. Finally, Chapter 6 synthesises the findings and outlines directions for future research.

Chapter 2

Background

This thesis addresses the fundamental question of what affects the prunability of convolutional neural networks and employs multiple lenses to explore it. In this chapter, the concept of the loss landscape of neural networks is presented first, along with its dependence on architecture and its impact on training. Next, it examines the loss landscape in the sparse regime, where evidence suggests that sparsity imposes restrictions that hinder models' ability to reach good local minima. This relates to Chapter 3, where the search is performed in the basin of local minima for sparse networks that perform well.

Second, this chapter investigates the role of the receptive field in generating redundancy within the network by saturating representations. This relates to Chapter 4, where receptive field modulation was used to increase prunability of CNNs. Third, the concepts of Hessian and Hessian spectrum of neural networks are presented. These concepts measure the local curvature of the loss landscape and are used in two ways: to estimate how difficult it is to train a particular neural network and to better navigate the loss landscape with curvature-aware optimisers (second-order and sharpness-aware optimisers). Finally, the neural network pruning methods used in this thesis are presented: global magnitude pruning, layer-adaptive magnitude pruning, and gradient signal preservation.

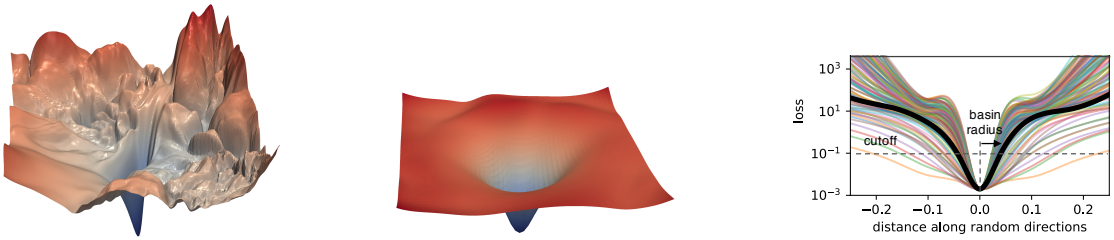
2.1 Loss Landscape of Neural Networks

The loss landscape of neural networks plays a central role in our understanding of optimisation and generalisation in deep learning. Traditionally, deep neural networks were believed to be plagued by highly non-convex loss landscapes, peppered with numerous poor local minima that could impede optimisation and result in poor generalisation. This perception stemmed from the complexity and high dimensionality of the neural network parameter spaces, where intuition from classical optimisation theory suggested that finding global minima would be extremely difficult (Huang *et al.*, 2020; Li *et al.*, 2018b).

Several theoretical results contribute to our understanding of the unexpected high performance of SGD on such a highly non-convex landscape. Kawaguchi *et al.* (2019) proved theoretically that for deep nonlinear neural networks with squared loss, increasing their depth and width reduces the quality gap between local minima and the global minimum. Furthermore, the local structure of deep nonlinear neural networks gives theoretical guarantees that local minima are at least as good as the globally optimal values of the corresponding classical machine learning models. Kawaguchi & Bengio (2019) proved the above statement, but in the case of ResNets, with additional guarantees of further improvement derived from the residual representations.

However, advances in neural network architecture design have transformed this narrative. Empirical studies and visualisation techniques have revealed that architectural innovations, most notably the introduction of skip connections in residual networks, can dramatically alter the geometry of the loss landscape (Figure 2.1). For well-defined tasks such as image classification, these advances often yield landscapes that are nearly convex in the vicinity of global minima, with wide, gently sloped basins rather than the chaotic and rugged terrains previously imagined (Li *et al.*, 2018b). This surprising result suggests that, rather than being hindered by poor local minima, modern neural networks can exploit the structure of their loss surfaces to achieve both efficient optimisation and strong generalisation.

From an empirical perspective, Huang *et al.* (2020) explained how these over-parametrized models achieve good performance due to the "blessing" of dimensionality. The authors ask why does SGD, as simple as it is, find good flat minima



(a) No skip connections

(b) With skip connections

(c) Basin radius

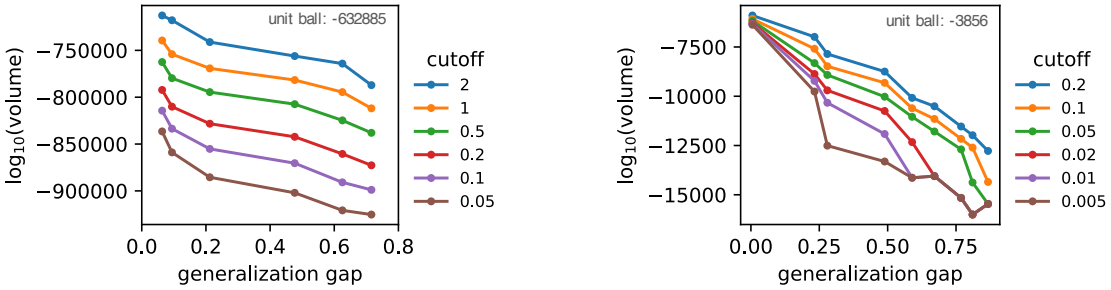
Figure 2.1: **Visualisation of Loss Landscape of Neural Networks: a-b)** The loss surfaces of ResNet-56 with/without skip-connections on ImageNet (Li *et al.*, 2018b) **c)** SVHN loss along random directions, the basin is estimated as all the values that lie beneath the cut-off value (Huang *et al.*, 2020).

while avoiding bad sharp ones? They argue that the optimiser's bias toward good solutions arises from the volume disparity between the basins around good and bad minima. Flat minimisers that generalise well lie in wide basins that occupy a large volume of parameter space, while sharp minimisers lie in narrow basins that occupy a comparatively small volume of parameter space. In consequence, random search algorithms are more likely to land in the attraction basin for a good minimiser than a bad one.

In Figure 2.1c are illustrated the 3k random directions used by Huang *et al.* (2020) to estimate the volume of a basin. Huang *et al.* (2020) defined a basin as the set of points in a neighbourhood of the minimiser that have a loss value below a cut-off of 0.1. Once the basin radius is estimated, they compute the volume using a Monte Carlo integration method. The authors found a correlation between the volume/radius of the solution basin and the accuracy gap, as shown in Figure 2.2.

Loss Landscape of Sparse Models

The previous section explores the characteristics of the loss landscape of neural networks in the dense regime. This section examines the loss landscape in the sparse regime, which presents greater optimisation challenges than its dense counterpart. Generally, pruning a network and subsequently training in a sparse space does not match, let alone surpass, the performance of sparse solutions obtained



(a) SVHN (b) Swissroll

Figure 2.2: **Relationship Between Generalisation and Volume:** Data was gathered from multiple random runs of the optimiser, specifically 10 for the swissroll dataset and 4 for SVHN, along with 3,000 random directions to evaluate the basin's radius. It can be seen that volume correlates with a diminishing generalisation gap, even for small cutoff to estimate the volume. (Huang *et al.*, 2020).

through the conventional approach of dense training followed by pruning (Evci *et al.*, 2020a).

The best results are obtained when this training-pruning process is repeated throughout training, with the pruning rate adjusted at each stage. What Evci *et al.* (2019, 2020a) discovered is that a loss barrier prevents solutions pruned at initialisation from reaching the superior local minima accessible to solutions pruned after training (Figure 2.3).

The left panel of Figure 2.3 illustrates this phenomenon: the training loss of models pruned after training (Pruning) is lower than that of models pruned before training (Static), where the mask remains static during training. Following the methodology of Garipov *et al.* (2018), the authors perform linear and Bézier curve interpolations to find low-loss paths between these two solutions in both sparse (with mask) and dense (without mask) spaces. The results clearly show that all interpolations in sparse space present a barrier between the two solutions, whereas the interpolation in dense space does not.

To avoid this problem, many researchers have developed Dynamical Sparse Training (DST) methods, which allow the connectivity of sparse networks to change during training (Evci *et al.*, 2020a; Liu *et al.*, 2020; Mocanu *et al.*, 2018). For example, when RigL (Evci *et al.*, 2020a) is applied to further train a static

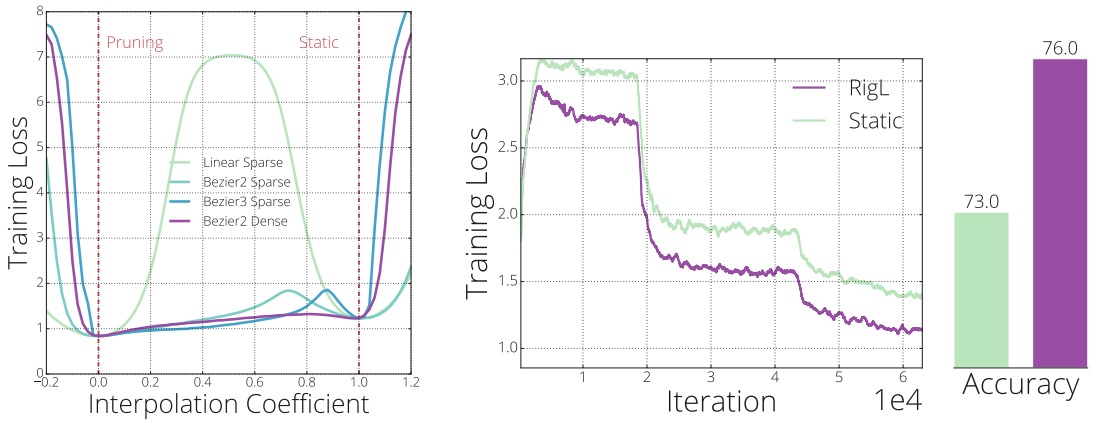
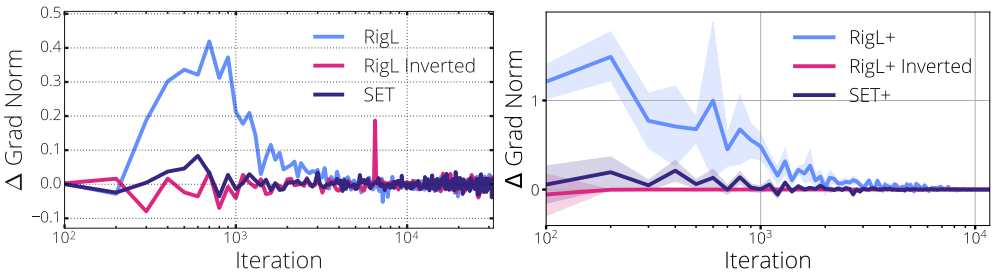


Figure 2.3: **ResNet50 Trained on CIFAR10:(left)** Training loss measured at different points along the interpolation curves between a magnitude-pruned model (0.0) and a model with fixed sparsity (1.0). **(right)** Training loss and final accuracies of the *RigL* and *static* approaches, both initiated from the static sparse solution (Evci *et al.*, 2020a).

solution (a model that was pruned and then trained with a static mask), it is observed that RigL can escape the local minimum, while retraining with static sparse training cannot (Figure 2.3, right panel).

An exception to this is the work of Frankle & Carbin (2018) on the Lottery Ticket Hypothesis (LTH). They demonstrated that it is possible to identify and train deep neural network subnets that can match or exceed the test accuracy of the original network when trained from scratch. These sparse, trainable subnets were termed "winning tickets" by the authors, who argued that pruning can successfully identify such subnets.

The study found that network initialisation significantly impacts performance, as reinitialising pruned subnetworks led to worse results. Through experiments on MNIST with fully connected networks and on CIFAR10 with convolutional networks (Conv-4, Conv-6, VGG19, ResNet18), the authors showed that iterative pruning outperformed one-shot pruning, enabling pruned networks to retain only 13.5% of the original weights while maintaining or even improving accuracy. Beyond this threshold, performance declined. Notably, random reinitialisation did not hinder training of pruned networks (or "winning tickets") at moderate sparsity



(a) ResNet-50 (b) LeNet5

Figure 2.4: **Impact of Mask Adjustments in Dynamic Sparse Training.** Influence of mask adjustments on gradient norms. *RigL Inverted* opts for connections with minimal magnitude. The authors evaluate the gradient norm both before and after the mask adjustments, denoted by Δ. The '+ ' symbol denotes the initialisation suggested by the authors that enhances the GF utilised in MNIST trials (Evci *et al.*, 2022).

levels (e.g., 92.5% pruning), and applying dropout further boosted their effectiveness. The choice of optimiser and learning rate was crucial: ADAM (Kingma & Ba, 2017) consistently discovered winning tickets, whereas the success of SGD depended heavily on a smaller learning rate.

Only LTH or DST can prune to achieve performance comparable to or better than post-training solutions. Researchers have studied the mechanisms underlying LTH and DST performance (Evci *et al.*, 2022). The main findings of (Evci *et al.*, 2022) are as follows:

- **Sparse neural networks exhibit poor Gradient Flow (GF) at initialisation and during training:** The authors assume that "Gradient Flow" serves as a proxy for both generalisation and optimisation performance, defining it as:

$$\Delta L \approx -\epsilon \nabla L^T(\theta) \nabla L(\theta) = \text{GF}, \tag{2.1}$$

where ϵ is a sufficiently small learning rate.

They hypothesise that poor gradient flow at initialisation is a primary cause of the suboptimal performance observed when pruning at initialisation. To

address this, the authors propose an initialisation scheme that improves GF at the start. For training, they argue that Dynamic Sparse Training (DST) methods such as Sparse Evolutionary Training (SET) (Mocanu *et al.*, 2018) and Rigged Lottery (RigL) (Evci *et al.*, 2020a) enhance gradient flow, which explains their superior performance compared to fixed masks (Figure 2.4).

- **Lottery tickets do not improve Gradient Flow; instead, they relearn their original pruning solution:** The authors measured the gradient flow of lottery tickets at initialisation and during multiple training runs, finding that the Gradient Flow remains consistently low. This suggests that the Lottery Ticket Hypothesis does not address the fundamental gradient flow problem (Figure 2.5).

In addition, the authors examined linear paths between the pruning solution and four other points: LT initialisation/solution and random (scratch) initialisation/solution. Each experiment was repeated 5 times with different random seeds and the mean values are provided with 80% confidence intervals.

In both experiments, the linear path between LT initialisation and the pruning solution exhibits a loss function that decreases more rapidly than the path from scratch initialisation to the pruning solution. However, after training, the linear paths towards the pruning solution change dramatically. The path from the scratch solution reveals a loss barrier, suggesting that the solution obtained from pruning followed by training from scratch resides in a different basin than the post-training pruning solution (Figure 2.6).

2.2 Receptive Field and Redundancy on Neural Networks

The receptive field of convolutional neural networks plays a crucial role in their representations and expressive power (Araujo *et al.*, 2019; Koutini *et al.*, 2019). This section examines how the receptive field affects redundancy in neural networks and how researchers have leveraged it to improve efficiency.

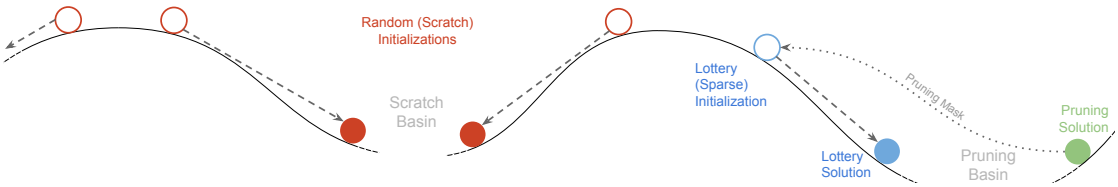
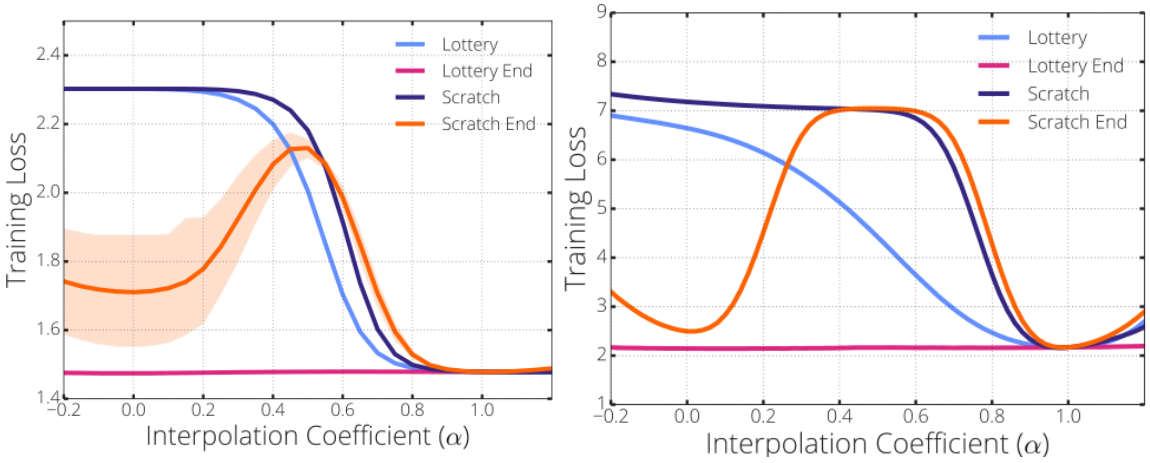


Figure 2.5: **Lottery Tickets are Biased Towards the Pruning Solution, Unlike Random Initialisation.** An illustrative cartoon depicts the loss landscape of a sparse model, derived from pruning a dense solution to form an LTH sub-network. An LTH initialisation typically resides within the basin of attraction of the pruned model's solution. Conversely, a random initialisation is less likely to fall near the basin of the dense solution (Evci *et al.*, 2022).



(a) LeNet5 trained on MNIST

(b) ResNet50 trained on ImageNet

Figure 2.6: **Experiments on Linear Interpolation:** A linear trajectory between the pruning solution ($\alpha = 1.0$) at both the initialisation and the endpoint (end of training) LT/scratch is explored (Evci *et al.*, 2022).

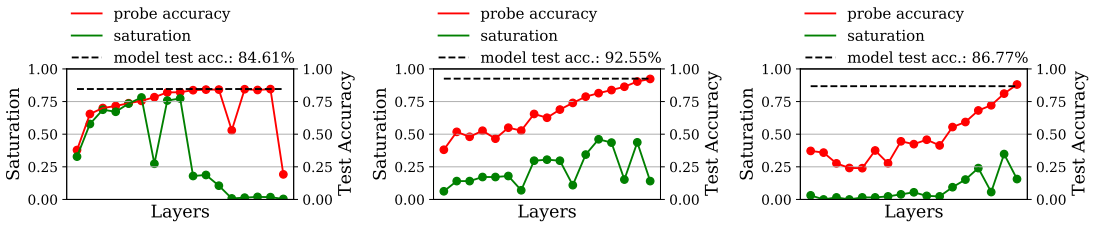
Richter *et al.* (2021c) introduced Feature Space Saturation (FSS) as an online, computationally efficient metric to analyse how much of a neural network layer's output space is actually utilised for the task at hand. The saturation is defined as:

$$s_l = \frac{\dim E_l^k}{\dim Z_l} \tag{2.2}$$

where dim E_l^k is the dimension of the principal eigenspace (obtained by PCA) required to explain a large fraction (for example, 99%) of the variance in the feature map of layer l, and $Z_l \coloneqq \sum_{i=0}^{n} \frac{z_{l,i}}{n}$ is the sum of all outputs of layer l over n samples of data. Then dim Z_l is the dimension of the matrix Z_l, which is the number of filters / neurons in the layer l.

Using the saturation metric s_l in conjunction with logistic probes that predict the desired output from intermediate layer representations, similar to Alain & Bengio (2018), the authors evaluated the efficacy of inference in these layers. They found that the output space is often partially unused or redundant, with useful information confined to a low-dimensional subspace (Figure 2.8). They noted that this redundancy (low saturation) is related to a mismatch between the network architecture, especially its depth and width, and the input resolution and growth of the receptive field in deeper layers (Figure 2.7).

Furthermore, the authors identified a "tail pattern" in the saturation of deeper layers, characterised by a sequence of layers with very low saturation. The layers associated with this tail pattern could be removed without degrading model performance (Figure 2.9).



(a) 32×32 (CIFAR10 native) (b) 224×224 (model standard) (c) 1024×1024

Figure 2.7: **Input Resolution Experiments:** Variations in input resolution impact the distribution of inference within the model, which can be seen through the probe's accuracy and saturation patterns. Optimal accuracy and balanced distribution are attained at the model's native resolution of 224×224 pixels (Richter *et al.*, 2021c)

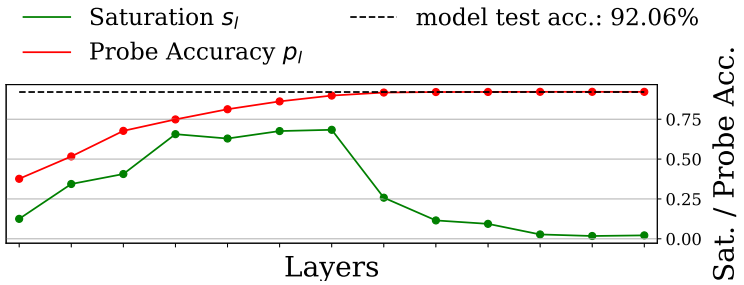


Figure 2.8: **Tail Pattern:** The x-axis is the index of the layer of the model. VGG16 begins to show a *tail characteristic* from the 8th layer onwards. Within this tail, the saturation s_l remains low, and the logistic regression probes p_l exhibit stagnation, indicating that these layers do not meaningfully enhance the model's predictions (Richter *et al.*, 2021b).

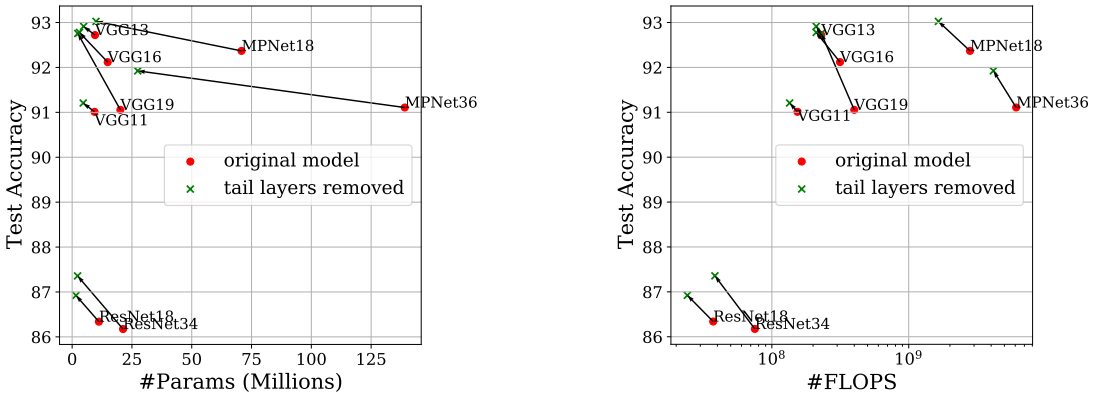


Figure 2.9: **Truncated Models trained on CIFAR10:** An elementary modification inspired by receptive field analysis involves replacing all layers where $r_{l-1,\min} > i$ with a straightforward output head. This effectively eliminates all subsequent layers from the "tail". This pruning enhances efficiency by improving performance and reducing the number of parameters and computational demands (Richter *et al.*, 2021b).

2.3 Second-Order Information in Neural Networks

2.3.1 What is the Hessian of a Function

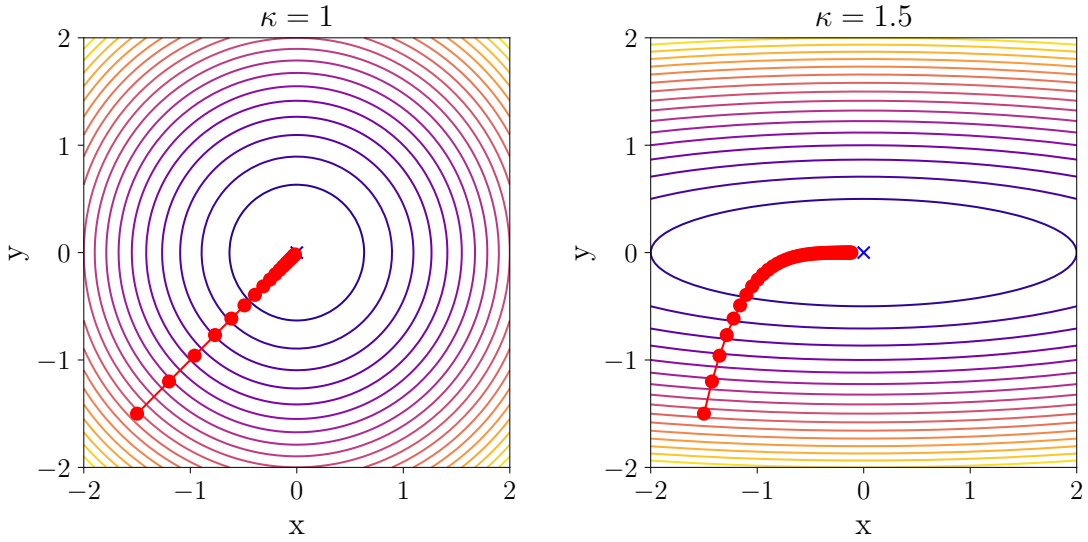
The Hessian of a multivariate function, commonly denominated as $H(f)$, or simply H is a matrix containing all the second partial derivatives of each variable x_i in the domain of f. For a function $f : \mathbb{R}^n \to \mathbb{R}$ with variables $x_1, x_2, \ldots, x_n$, the Hessian matrix H is defined as:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

For a twice-differentiable smooth function, the second-order necessary conditions establish that for a critical point x^* to be a *local minimiser*, the Hessian matrix H must be positive semidefinite (Nocedal & Wright, 2006). The positive definiteness of matrix H relates directly to its eigenvalues: all eigenvalues are greater than or equal to zero for positive semidefinite matrices ($\lambda_i \geq 0$) and less than or equal to zero for negative semidefinite matrices ($\lambda_i \leq 0$).

The eigenvalues and eigenvectors of H are of particular interest for optimisation, as they measure the local curvature of the loss function at the current point x_i in the optimisation process. Since eigenvalues characterise the curvature of the local geometry, they are used to estimate the difficulty of optimising a convex function f. This difficulty is quantified by the *condition number*, defined as $\kappa = \frac{\lambda_N}{\lambda_1}$, where λ_N and λ_1 are the largest and smallest eigenvalues of H, respectively. The effect of the condition number on the optimisation procedure is illustrated in Figure 2.10.

Advanced optimisers such as L-BFGS (Liu & Nocedal, 1989), line search, and trust region methods (Nocedal & Wright, 2006) all employ H^{-1} or an approximation of it in their update rules at every iteration. The convergence rate of these methods is bounded by κ.



(a) (b)

Figure 2.10: **Condition Number Effect in Optimisation:** On the left, **a)** the contour plot shows the optimisation trajectory for a function with a small condition number κ. For this case, 20 optimisation steps were performed. On the right **b)** There is the contour plot for a function with a large condition number, and, in this case, 50 optimisation steps were performed. It is clear that a larger condition number severely hinders the optimisation process: in a), the optimum is reached within 20 steps, whereas in b), the optimum is not reached even after 50 steps.

2.3.2 Hessian Spectrum of Neural Networks

The previous section explored the Hessian of a function, including its eigenvalues and eigenvectors. This section focuses on interpreting the Hessian in the context of neural networks.

Despite being highly non-convex and non-linear, neural networks can still achieve good local minima using simple optimisers such as Stochastic Gradient Descent (SGD). The shape of the landscape around a solution is closely linked to that solution's generalisation capabilities, specifically the performance gap between training and testing data. This landscape shape also relates to optimisation measures such as gradient noise and optimisation speed (Jiang *et al.*, 2019).

To approximate the curvature of the loss landscape and assess the complexity

involved in training a specific model, it is customary to estimate the eigenvalues of the Hessian. Since calculating the Hessian is computationally intractable, approximation methods typically estimate different statistics that do not require explicit computation of H, such as its trace, the N largest eigenvalues and the density of the eigenvalue spectrum (Ghorbani *et al.*, 2019; Yao *et al.*, 2020).

2.3.3 Structure of the Hessian Spectrum in Neural Networks

Researchers (Sagun *et al.*, 2016, 2018) have identified that fully connected feedforward neural networks exhibit a specific spectral structure: a bulk of the spectrum centred around zero, with large outliers linked to the data, specifically to the number of classes minus one. Furthermore, when fixing the model dimension and manipulating the data by making it less separable or by adding more clusters, only the outliers are affected; by contrast, fixing the data and increasing the number of parameters increases the size of the bulk. The authors assume the that the loss function can be decomposed into two functions: $f_\bullet : \mathbb{R}^M \longrightarrow \mathbb{R}$ is the real-valued output of a network that depends on the parameters; and the loss function $\ell_\bullet : \mathbb{R} \longrightarrow \mathbb{R}^+$ is a convex function (e.g., mean-square loss, negative log-likelihood loss). Here, $\bullet$ refers to the given example. With some simplifications the hessian matrix (see section 2.2 of Sagun *et al.* (2018) for a full derivation)

$$\nabla^2 \mathcal{L}(\hat{w}) \approx \frac{1}{N} \sum_{i=1}^{N} [\sqrt{\ell_i''(f_i(\hat{w}))} \nabla f_i(\hat{w})][\sqrt{\ell_i''(f_i(\hat{w}))} \nabla f_i(\hat{w})]^T \tag{2.3}$$

Here, Equation 2.3 is the sum of rank one matrices (via the outer products of gradients of f multiplied by some non-negative number), therefore, the sum can be written as a product of an $M \times N$ matrix with its transpose where the columns of the matrix are formed by the scaled gradients of f. Immediately, this implies that there are at least $M - N$ many trivial eigenvalues of the right-hand side in Equation 2.3.

Additionally, Sagun *et al.* (2018) points out that negative eigenvalues exist, which, during training, seem to bring no further improvement, although their magnitude is much smaller than the outliers. These observations led the authors to

suggest that the notion of basins in the loss landscape of neural networks might be misleading given the flatness of the local region. Fort & Ganguli (2019) generated a theoretical model that explains the observation that there are C large outlier eigenvalues. This observation is valid for large (Ghorbani *et al.*, 2019; Gur-Ari *et al.*, 2018) and small (Sagun *et al.*, 2016, 2018) networks trained on a classification task, where C is the number of classes. The authors show that there are exactly C directions with high positive curvature, and these gradient directions are largely confined to this extremely low-dimensional subspace of positive Hessian curvature, leaving the vast majority of directions in the weight space unexplored.

The Hessian spectrum is also used to explain why certain architectural choices improve neural network training. For example, Ghorbani *et al.* (2019) demonstrates that batch normalisation (Ioffe & Szegedy, 2015) suppresses large positive Hessian eigenvalues, thus improving training. Similarly, Yao *et al.* (2020) reports that removing residual connections slightly increases the top eigenvalue, trace and the Hessian eigenvalue spectrum-density support ranges, which correlate with training difficulty. Although Li *et al.* (2018b) observed that adding residual connections increases the smoothness of the loss landscape using the filter-normalised random direction method to plot 3D loss landscapes, Yao *et al.* (2020) demonstrates the same result exclusively using the Hessian.

2.3.4 Why Hessian Outliers Make Training More Difficult

Ghorbani *et al.* (2019) considered a simple quadratic approximation to the loss around the optimum θ^*:

$$\mathcal{L}(\theta) \approx \mathcal{L}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H(\theta - \theta^*) \tag{2.4}$$

Without loss of generality, assume that $H = \text{diag}(\lambda_1, \ldots, \lambda_n)$ with $\lambda_i > 0$. It can be shown that when optimised with gradient descent using a learning rate $\eta < 2/\lambda_i$ sufficiently small for convergence, in the eigenbasis we have:

$$|\hat{\theta}_t - \theta^*|_i \leq \left|1 - \frac{2\lambda_i}{\lambda_1}\right|^t |\hat{\theta}_0 - \theta^*|_i \tag{2.5}$$

For all directions where λ_i is small relative to λ_1, convergence is expected to be slow. One might hope that these small λ_i do not contribute significantly to

the loss; unfortunately, when Ghorbani *et al.* (2019) measured this in a ResNet-32 without batch normalisation, a small ball around zero accounts for almost 50% of the total L_1 energy of the Hessian eigenvalues for a converged model (the L_1 reflects the loss function $\sum_i \lambda_i(\theta - \theta^*)_i^2$).

The conclusion is that, to achieve successful optimisation, we must optimise these slowly converging directions. The authors note that although the loss function in deep networks is not quadratic, the intuition provided by the result above remains valid in practice.

The second reason lies in the interaction between the Hessian's large eigenvalues and the stochastic gradients. Define the covariance of the (stochastic) gradients at time t as:

$$\Sigma(t) \equiv \frac{1}{N} \sum_{i=1}^{N} \nabla \mathcal{L}_i \nabla \mathcal{L}_i^T \tag{2.6}$$

The eigenvalue density of Σ characterises how the energy of the gradients (mini-batch) is distributed. As with the Hessian, we observe that in non-BN networks the spectrum of Σ has outlier eigenvalues (Figure 2.11). Throughout the optimisation, almost all of the gradient energy is concentrated in these outlier subspaces (Figure 2.12), reproducing the observations of Gur-Ari *et al.* (2018).

2.3.5 Second-Order Information Optimisers for Deep Learning

Second-order methods are a natural extension of SGD. These methods estimate the local curvature of the loss landscape; they can, in theory, navigate the loss landscape of networks better than SGD. For this reason, this section presents research on second-order methods for deep learning.

In Xu *et al.* (2020), the authors empirically evaluate two Newton-type methods against momentum SGD to assess their practical impact more accurately. They want to assess the effectiveness of employing curvature information as a remedy for the well-known deficiencies of SGD, such as relatively slow convergence, sensitivity to hyperparameter settings, stagnation, and entrapment near saddle points. The methods used by the authors are: Trust region methods (Nocedal & Wright, 2006),

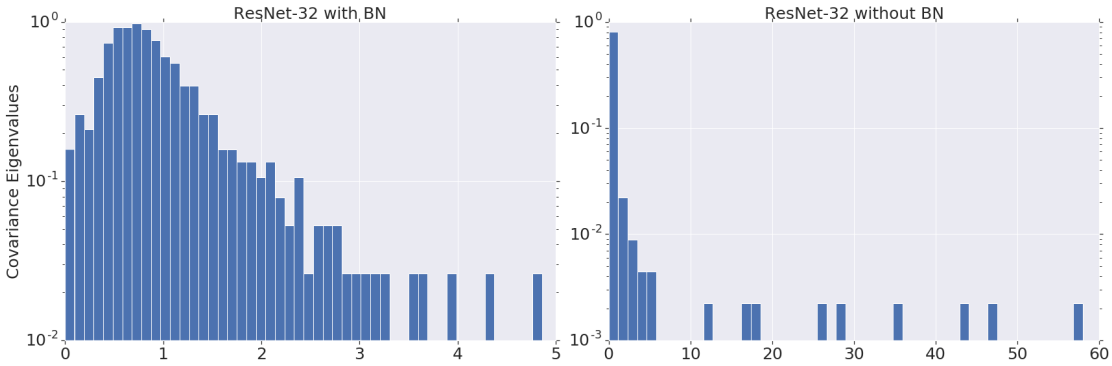


Figure 2.11: **Histogram of the Eigenvalues of Hessian matrix of Resnet-32:** On the left and right, it can be seen that the models trained for $9k$ steps with and without batch normalisation, respectively. When the model is trained without batch normalisation, almost 99% of the energy is in the top few subspaces. For easier comparison, the distributions are normalised to have the same mean (Ghorbani *et al.*, 2019).

and cubic regularisation methods (Cartis *et al.*, 2011). The authors posed four questions:

- **Q.1 (Computational efficiency)** Can these subsampled Newton-type methods be computationally efficient enough to be competitive with hand-tuned SGD with momentum?
- **Q.2 (Robustness to hyperparameters)** Does the performance of such Newton-type methods exhibit robustness to hyperparameter tuning?
- **Q.3 (Escaping Saddle Points)** Does employing Hessian information help with avoiding saddle points and converging to lower training errors in highly non-convex problems?
- **Q.4 (Generalisation Performance)** Can second-order methods be beneficial for obtaining a low generalisation error in machine learning problems?

Their experimental setup consists of a one-layer MLP used in CIFAR10 and a deep autoencoder used in MNIST. The experiments they carried out showed that these two methods are computationally competitive with "well-tuned" SGD but

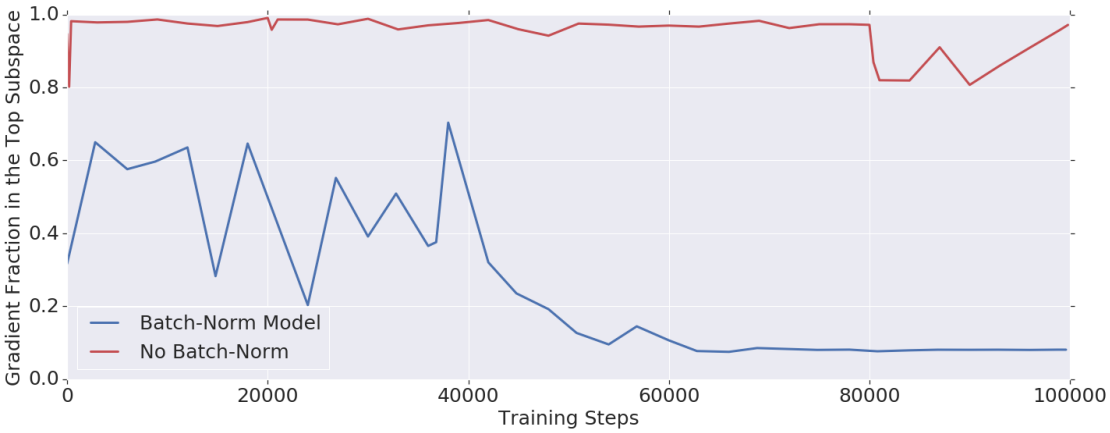


Figure 2.12: **Projection of the Gradient in the Eigenspace:** $\frac{\|P\nabla\mathcal{L}(\hat{\theta}_i)\|_2^2}{\|\nabla\mathcal{L}\hat{\theta}_i\|_2^2}$ for a Resnet-32. Here P is the projection operator to the subspace spanned by the 10 most dominant eigenvectors of $\nabla^2\mathcal{L}(\hat{\theta}_i)$. Almost all the variance of the gradient of the model with no batch normalisation is in this subspace (Ghorbani *et al.*, 2019).

are slightly slower. In addition, these methods proved to be highly robust to their hyperparameters, achieving low training and test errors for widely different hyperparameter settings. Lastly, these methods can effortlessly escape saddle points and consistently achieve a low generalisation gap. Thus, Xu *et al.* (2020) paints a broader picture of how the trust-region and adaptive cubic regularisation methods (as second-order methods) compare to first-order methods in non-convex machine learning.

Other papers used approximations of the natural gradient that are not based on Newton or Gauss-Newton type methods. One of such methods is explored by Martens & Grosse (2015) where they used a Kronecker factorisation to efficiently estimate the update $F^{-1}\nabla\mathcal{L}(x)$ where F is the Fisher information matrix. The main approximation comes from the fact that each element of the Fisher matrix can be written as

$$F_{ij} = \mathbb{E}\left[\operatorname{vec} a_{i-1} \operatorname{vec} a_{j-1}^T \otimes \operatorname{vec} g_i \operatorname{vec} g_j\right],$$

with $\operatorname{vec} a_i$ and $\operatorname{vec} g_j$ denoting the activations of the i-th layer and the gradient of the j -the layer, respectively. Then, despite the fact that in general, the expected value of a Kronecker product is not equal to the Kronecker product of the expected

values, the approximation is

$$F_{ij} = \mathbb{E}\left[\operatorname{vec} a_{i-1} \operatorname{vec} a_{j-1}^T \otimes \operatorname{vec} g_i \operatorname{vec} g_j^T\right] \approx \mathbb{E}\left[\operatorname{vec} a_{i-1} \operatorname{vec} a_{j-1}^T\right] \otimes \mathbb{E}\left[\operatorname{vec} g_i \operatorname{vec} g_j^T\right],$$

where the underlying assumption is that every pair of products $\operatorname{vec} a_{i-1} \operatorname{vec} a_{j-1}^T$ of unit activations are statistically independent of the gradient product for each unit $\operatorname{vec} g_i \operatorname{vec} g_j^T$. The authors applied this algorithm to three deep autoencoder optimisation problems in Hinton & Salakhutdinov (2006), which use the MNIST, CURVES, and FACES datasets. The baseline for comparison is SGD with momentum. The authors utilise three types of approximations: block tri-diagonal with momentum, block diagonal with momentum, and block tri-diagonal without momentum. They also used an exponentially increasing batch size schedule that was proven to help K-FAC during optimisation. K-FAC is exponentially faster per iteration than SGD with momentum. Not only that, but when computation time is taken into account, K-FAC is faster than SGD. It is important to note that momentum also plays a big role in K-FAC, since when removed, K-FAC takes more or equal computation time than SGD in two out of three of the benchmarks.

Implicit Second-Order Information Optimisers

Other optimisers exist that use second-order information implicitly rather than explicitly (without matrix or vector-matrix computation). Foret *et al.* (2021) devised a Sharpness-Aware Minimisation (SAM) optimiser whose success can be explained through second-order information.

The loss objective defined for SAM is:

$$\min_w L_S^{SAM}(w) + \lambda \|w\|^2 \quad \text{where} \quad L_S^{SAM}(w) \triangleq \max_{\|\epsilon\| \leq \rho} L_S(w + \epsilon) \tag{2.7}$$

This means that SAM seeks not only a good solution, but also a solution whose worst neighbour is also good. Through a series of simplifications (for a detailed derivation, see (Foret *et al.*, 2021)), the authors arrive at the following approximation:

$$\nabla_w L_S^{SAM}(w) \approx \nabla_w L_S(w)\big|_{w+\hat{\epsilon}(w)}$$

where $\hat{\epsilon}(w)$ is defined as:

$$\hat{\epsilon}(w) = \frac{\rho \cdot \text{sign}(\nabla_w L_S(w)) \cdot |\nabla_w L_S(w)|^{q-1}}{\|\nabla_w L_S(w)\|_q^{q/2}}$$

and $|\cdot|^{q-1}$ denotes the element-wise absolute value and power operation and ρ is the neighbourhood radius.

One update step of SAM requires two gradient evaluations; therefore, the authors allow SGD to perform twice as many iterations as SAM for a fair comparison. Table 2.1 shows the test error rates for ResNets trained on ImageNet, with and without SAM.

Model	Epoch	SAM		Standard Training (No SAM)	
		Top-1	Top-5	Top-1	Top-5
ResNet-50	100	**22.5±0.1**	6.28±0.08	22.9±0.1	6.62±0.11
	200	**21.4±0.1**	5.82±0.03	22.3±0.1	6.37±0.04
	400	**20.9±0.1**	5.51±0.03	22.3±0.1	6.40±0.06
ResNet-101	100	**20.2±0.1**	5.12±0.03	21.2±0.1	5.66±0.05
	200	**19.4±0.1**	4.76±0.03	20.9±0.1	5.66±0.04
	400	**19.0±<0.01**	4.65±0.05	22.3±0.1	6.41±0.06
ResNet-152	100	**19.2±<0.01**	4.69±0.04	20.4±<0.0	5.39±0.06
	200	**18.5±0.1**	4.37±0.03	20.3±0.2	5.39±0.07
	400	**18.4± <0.01**	4.35±0.04	20.9±<0.0	5.84±0.07

Table 2.1: Top-1 and Top-5 error rates (in %) on the ImageNet validation set (Foret *et al.*, 2021).

This improvement in model generalisation occurs because SAM finds flatter solutions than SGD. Flatness is known to correlate with generalisation (Hochreiter & Schmidhuber, 1997; Jiang *et al.*, 2019), thereby explaining the observed performance increase. This metric is intrinsically related to second-order information.

The spectrum density of the Hessian eigenvalues of WideResNet-28-10 trained on CIFAR-10 for 300 steps with and without SAM is shown in Figure 2.13. When the model is trained with SAM, the spectrum is much narrower than when trained with SGD, indicating a flatter landscape.

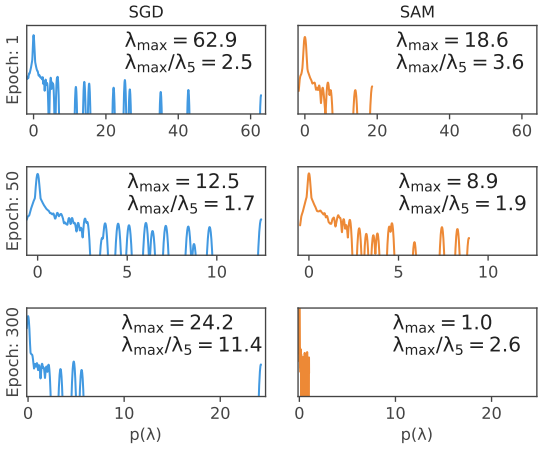


Figure 2.13: **Eigenvalue Spectrum of Wide-ResNet40-10 Trained on CIFAR10:** Notice that SAM greatly reduces the $\frac{\lambda_m ax}{\lambda_5}$ ratio of the solution found (Foret *et al.*, 2021).

Although SAM does not explicitly use second-order information, it acknowledges its importance for generalisation by optimising a measure intrinsically connected to second-order information, namely the sharpness of the loss landscape. Proving that even when it is not explicitly exploited, second-order information can aid the optimisation process.

2.4 Neural Network Pruning Methods

Approaches to compressing neural networks range from designing efficient architectures from scratch (Tan & Le, 2019) to pruning-based methods. Among these techniques, pruning remains the most widely used because of its simplicity and practicality. In general, pruning algorithms can be classified into three categories: *pre-training*, *training*, and *post-training* pruning.

As their names indicate, pre-training pruning methods focus on designing a pruning criterion that enables the pruned model to achieve good performance during training. Methods that prune during training can range from simple iterative pruning schedules (Frankle & Carbin, 2018), where cycles of pruning and training alternate, to more complex algorithms such as dynamic sparse training, which al-

low modification of the model's sparse connectivity (Evci *et al.*, 2020a; Liu *et al.*, 2020; Mocanu *et al.*, 2018).

This focuses on pruning after training, as it is the simplest approach and can be applied to off-the-shelf pre-trained models. Pruning methods differentiate themselves by the type of saliency score they use to assess the importance of weights. This thesis employs three main methods: Global Magnitude Pruning (GMP), Layer-Adaptive Magnitude Pruning (LAMP) (Lee *et al.*, 2022), and Gradient Signal Preservation (GraSP) (Wang *et al.*, 2019a). GMP is the simplest and most widely used method. It ranks all model weights by absolute value and sets any weight below the desired threshold to zero.

Layer-Adaptive Magnitude Pruning

Alternatively, LAMP (Lee *et al.*, 2022) operates similarly to the magnitude pruning applied per layer, but automatically selects the pruning rates for each layer. This technique uses a saliency score that accounts for the model-level ℓ^2 distortion caused by pruning, called Layer Adaptive Magnitude-Based Pruning (LAMP). The LAMP score facilitates the automatic selection of per-layer pruning rates for magnitude pruning. Assuming the weights are sorted in ascending order according to the given index map, which means $|W[u]| \leq |W[v]|$ whenever $u < v$, where $W[u]$ is the weight entry mapped by the index u. Thus, the LAMP score for the u-th index of the weight tensor W is defined to measure the relative importance of the target connection compared to all remaining connections in the same layer. Connections with smaller weight magnitudes in the same layer have already been pruned, and as a result, two connections with identical weight magnitudes can have different LAMP scores depending on the index map used.

Gradient Signal Preservation (GraSP)

In the case of GraSP, this method was devised as a pre-training pruning method. Is a method that preserves gradient flow when pruning, since gradient flow has been proven to be critical for the successful training of sparse networks (Evci *et al.*, 2019, 2022). Mathematically, a larger gradient norm indicates that, to the first

order, each gradient update achieves a greater loss reduction, as characterised by the following directional derivative:

$$\mathcal{L}(\boldsymbol{\theta}) = \lim_{\epsilon \to 0} \frac{\mathcal{L}\left(\boldsymbol{\theta} + \epsilon \nabla \mathcal{L}(\boldsymbol{\theta})\right) - \mathcal{L}(\boldsymbol{\theta})}{\epsilon} = \nabla \mathcal{L}(\boldsymbol{\theta})^\top \nabla \mathcal{L}(\boldsymbol{\theta}) \tag{2.8}$$

The authors would like to preserve or even increase (if possible) the gradient flow *after pruning* (i.e. the gradient flow of the pruned network). Following LeCun *et al.* (1989), the authors made the pruning operation as adding a perturbation $\boldsymbol{\delta}$ to the initial weights. Then they used a Taylor approximation to characterise how removing one weight will affect the gradient flow *after pruning*:

$$\begin{aligned} \mathbf{S}\left(\boldsymbol{\delta}\right) &= \Delta \mathcal{L}(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) - \underbrace{\Delta \mathcal{L}(\boldsymbol{\theta}_0)}_{\text{Const}} = 2\boldsymbol{\delta}^\top \nabla^2 \mathcal{L}(\boldsymbol{\theta}_0) \nabla \mathcal{L}(\boldsymbol{\theta}_0) + \mathcal{O}(\|\boldsymbol{\delta}\|_2^2) \\ &= 2\boldsymbol{\delta}^\top \mathbf{H} + \mathcal{O}(\|\boldsymbol{\delta}\|_2^2), \end{aligned} \tag{2.9}$$

where $\mathbf{S}(\boldsymbol{\delta})$ approximately measures the change to (2.8). The Hessian matrix $\mathbf{H}$ captures dependencies among weights and thus helps predict the effect of removing multiple weights. When $\mathbf{H}$ is approximated as the identity matrix. However, it has been observed that different weights are highly coupled (Hassibi *et al.*, 1993), indicating that $\mathbf{H}$ is, in fact, far from the identity.

GraSP uses Equation (2.9) to compute the score of each weight, which quantifies the change in gradient flow resulting from pruning that weight. Specifically, if $S(\delta)$ is negative, removing the corresponding weights reduces the gradient flow; if it is positive, it increases it. Therefore, in this case, the larger the weight's score, the lower its importance. The authors first remove weights whose removal does not reduce the gradient flow. For each weight, the score can be computed in the following way (by abuse of notation, bold $\mathbf{S}$ denotes vectorised scores):

$$\mathbf{S}(-\boldsymbol{\theta}) = -\boldsymbol{\theta} \odot \mathbf{H} \tag{2.10}$$

For a given pruning ratio p, we obtain the resulting pruning mask by computing the score of every weight and removing the *top* p fraction of the weights (see Algorithm 1).

Hence, GraSP accounts for gradient flow during pruning. GraSP is efficient and easy to implement; the Hessian-gradient product can be computed without explicitly constructing the Hessian using higher-order automatic differentiation (Pearlmutter, 1994; Schraudolph, 2002).

Algorithm 1 Gradient Signal Preservation (GraSP).

Require: Pruning ratio p, training data $\mathcal{D}$, network f with initial parameters $\boldsymbol{\theta}_0$

1: $\mathcal{D}_b = \{(_i,_i)\}_{i=1}^{b} \sim \mathcal{D}$ ▷ Sample a collection of training examples
2: Compute the Hessian-gradient product $\mathbf{H}$ (see Eqn. (2.10)) ▷ See Algorithm 2
3: $\mathbf{S}(-\boldsymbol{\theta}_0) = -\boldsymbol{\theta}_0 \odot \mathbf{H}$ ▷ Compute the score of each weight
4: Compute p_{th} percentile of $\mathbf{S}(-\boldsymbol{\theta}_0)$ as τ
5: $= \mathbf{S}(-\boldsymbol{\theta}_0) < \tau$ ▷ Remove the weights with the *largest* scores
6: Train the network $f_{\odot\boldsymbol{\theta}}$ on $\mathcal{D}$ until convergence.

Algorithm 2 Hessian-gradient Product.

Require: A batch of training data $\mathcal{D}_b$, network f with initial parameters $\boldsymbol{\theta}_0$, loss function $\mathcal{L}$

1: $\mathcal{L}(\boldsymbol{\theta}_0) = \mathbb{E}_{(,y)\sim\mathcal{D}_b}[\ell(f(;\boldsymbol{\theta}_0), y)]$ ▷ Compute the loss and build the computation graph
2: $= \text{grad}(\mathcal{L}(\boldsymbol{\theta}_0), \boldsymbol{\theta}_0)$ ▷ Compute the gradient of loss function with respect to $\boldsymbol{\theta}_0$
3: $\mathbf{H} = \text{grad}(^{\top}\text{stop\_grad}(), \boldsymbol{\theta}_0)$ ▷ Compute the Hessian vector product of $\mathbf{H}$
4: Return $\mathbf{H}$

Chapter 3

Stochastic Pruning for Neural Networks

3.1 Introduction

Pruned networks share the same basin of attraction as the dense networks they originate from (Evci *et al.*, 2019) and are robust to minor perturbations (Frankle & Carbin, 2018). This observation raises the central question of this chapter: Can stochastic perturbations help us better explore and exploit these local basins? This chapter investigates the basin of a minimiser by adding Gaussian noise prior to pruning, thereby creating models that maintain similar performance while remaining within the same basin. This process is referred to as stochastic pruning (SP). The results show that, across different datasets and models, SP with high pruning rates achieves superior one-shot performance (no fine-tuning) compared to deterministic pruning applied to the original unperturbed network, provided an appropriate noise amplitude.

This is relevant since one-shot pruning may suffice in cases where it is competitive with iterative pruning, e.g., as observed in some cases for CNNs Frankle & Carbin (2018). Finally, one-shot pruning is useful as a first step before fine-tuning, as experiments show that the advantage of stochastic pruning is preserved after fine-tuning, albeit depending on the pruning methodology. For more sophisticated pruning methods, it is evident that the noise amplitude exhibits an inverse rela-

tionship with the final fine-tuned performance, as it affects the gradient flow, thus reducing the efficacy of training (Evci *et al.*, 2022).

The procedure proposed here involves two adjustable parameters: the noise amplitude σ and the pruning rate γ. To systematically search for "good" parameters, that is, parameters (σ, γ) that allow Stochastic Pruning to outperform deterministic pruning, two approaches were used: An exhaustive grid search, which quantifies one objective at a time, and multi-objective evolutionary optimisation.

Next, a phenomenon termed feature variance explosion is identified. This phenomenon consists of highly (deterministically) pruned networks that have a much larger feature map variance discrepancy compared to their dense counterpart. Discrepancies in feature variance interfere with inference and, consequently, with accuracy. The results show that SP alleviates this explosion, explaining in part how stochastic pruning enhances one-shot performance.

The main contributions of this chapter are as follows.

- Stochastic Pruning of pre-trained networks (SP) outperforms deterministic GMP in one-shot settings, with fine-tuned models often matching or exceeding deterministic ones.
- Improved performance of SP solutions arises from the combination of (i) the pruning mask and (ii) altered weights.
- Using SP with GMP one-shot SP generally alleviates *the feature variance explosion.*
- This chapter demonstrates a trade-off between one-shot accuracy and trainability of SP solutions mediated by noise level σ

The remainder of this chapter is organised as follows. After introducing related work (Section 3.2), the methodology of Stochastic Pruning is described (Section 3.3). This chapter evaluated the various factors contributing to the performance of stochastic solutions (3.4.1) and demonstrated the generalisability of the results across different models and test data (3.4.2). 3.4.3 describes the use of multi-objective optimisation to extend the previous analysis. Mechanistic insight in terms of feature variance explosion is presented in 3.4.4. Finally, 3.4.5 shows that

stochastic models can surpass deterministic pruning solutions after fine-tuning, depending on the pruning method used. The chapter ends with a brief conclusion (Section 3.5).

3.2 Related Work

As reviewed in Chapter 2, pruning algorithms can be classified into three categories, *pre-training*, *training*, and *post-training* pruning. This section explores more in depth the literature for each one of these three categories.

Pre-Training Pruning: This family of algorithms prunes a network on randomly initialised weights, i.e. before training. Synaptic flow (SynFlow) (Tanaka *et al.*, 2020) uses no data for pruning yet has been shown to avoid layer collapse. Gradient Signal Preservation (GraSP) (Wang *et al.*, 2019a) prunes randomly initialised networks while preserving the gradient flow signal throughout the network. Yet a different approach, Single-Shot Network Pruning Based on Connection Sensitivity (Lee *et al.*, 2018) adds parameters that represent connections (but not weights) between neurons and uses the approximate gradient of the loss function with respect to the new parameters as a salience score for pruning.

Training: These algorithms prune connections during the training process; some prune gradually until they reach the final sparsity (Frankle & Carbin, 2018). This approach, commonly known as Iterative Magnitude Pruning (IMP), keeps the mask fixed after pruning. In contrast to IMP, Dynamical Sparse Training (DST) methods dynamically change the mask during training, starting with a sparse model before training, followed by pruning and regrowing the remaining connections based on different criteria. The authors in Mocanu *et al.* (2018) randomly prune a portion of the surviving connections and then regrow them following an Erdös-Rényi distribution. Sparse Networks from Scratch (SNFS) (Dettmers & Zettlemoyer, 2019) regrows weights based on each weight's momentum, reallocating weights from less efficient layers to more weight-efficient layers. The authors in Evci *et al.* (2020a) randomly prune a fraction of the surviving weights but regenerate the weights based on the gradients of the zero connections, reactivating the connection that would incur the maximum loss decrease.

Post-Training Pruning: These techniques perform pruning after the training phase and are typically accompanied by a fine-tuning schedule. Each method has a different importance score to detect unimportant weights. The simplest is the magnitude of a weight (Han *et al.*, 2015) where the lowest-magnitude weights are deemed unnecessary. However, this method can remove weak but important weights. Some methods use first-order Taylor expansion coefficients of the loss function (i.e. gradient information) as an importance score for pruning (Karnin, 1990; Mozer & Smolensky, 1988). Hessian-based methods measure the importance of weights as the effect they have on the second-order approximation of the training loss function (LeCun *et al.*, 1989; Singh & Alistarh, 2020). Many of these methods require a fine-tuning phase to mitigate accuracy loss, particularly at high pruning rates.

As mentioned in Chapter 2, this thesis focuses on post-training pruning. This approach capitalises on the availability of pre-trained models for a multitude of tasks, while acknowledging that network size often limits deployment in real-world scenarios, e.g. off-the-network, on mobile phones, and on edge devices. The experimental analysis predominantly used Global Magnitude Pruning (GMP); however, a subset of the findings were compared with the layer-adaptive magnitude pruning method (LAMP) (Lee *et al.*, 2022).

3.3 Stochastic Pruning

Neural network pruning is based on the observation that networks are often over-parameterized, meaning similar performance can be achieved with a smaller network (Gupta *et al.*, 2024). Various pruning techniques aim to identify a sparse network that maintains or gracefully degrades the performance of the original. According to Lee *et al.* (2018), the pruning problem can be framed as follows: Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ and a desired pruning rate γ (representing the fraction of total weights set to zero), the optimisation problem for neural network pruning can be formalised as follows:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{m}} \quad & \frac{1}{n}\sum_{i=1}^{n} \ell(F(\mathbf{m}\odot\mathbf{w}; x_i), y_i) \\ s.t. \quad & \mathbf{w}\in\mathbb{R}^d, \\ & \mathbf{m}\in\{0,1\}^d \quad \|\mathbf{m}\|_0 \geq \lfloor(1-\gamma)\cdot d\rfloor, \end{aligned} \tag{3.1}$$

where $\ell(\cdot)$ denotes the loss function, $\odot$ represents the Hadamard product, $F(\mathbf{m}\odot\mathbf{w}; x)$ is the network output given the weight set $\mathbf{w}$ and masks $\mathbf{m}$, d is the total number of parameters and $\|\cdot\|_0$ represents the ℓ_0-norm.

Most post-training pruning methods reveal a static mask $\mathbf{m}$ and fine-tune the remaining weights $\tilde{\mathbf{w}} = \mathbf{m}\odot\mathbf{w}$ to minimise the accuracy drop. Stochastic pruning is represented by the following equations:

$$\mathbf{w}_\mathrm{p} = \mathbf{w}^* + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \tag{3.2}$$

$$\hat{\mathbf{w}} = \mathbf{m}(\mathbf{w}_\mathrm{p})\odot\mathbf{w}_\mathrm{p}, \tag{3.3}$$

where $\mathbf{w}^*$ is a pre-trained set of weights, $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ represents Gaussian noise with standard deviation σ, and $\mathbf{m}(\mathbf{w}_\mathrm{p})$ is the mask dependent on the perturbed weights $\mathbf{w}_p$ via some function $f(\cdot)$. For GMP, $f(\cdot) = |\cdot|$, and for LAMP, $f(\cdot)$ corresponds to the LAMP score (Lee *et al.*, 2022). Equation (3.2) produces an intermediate network, which we call "Dense Stochastic". Finally, we call the original pruned network with $\sigma = 0$, "deterministic".

3.4 Experiments

The Stochastic Pruning approach was evaluated using three models, ResNet18, ResNet50, and VGG19 and two datasets, CIFAR-10 and CIFAR-100. All models were adapted for the CIFAR datasets[1]. The pre-trained models were trained for 200 epochs using Stochastic Gradient Descent (SGD) with an initial learning rate of 0.1, a cosine annealing schedule ($T_\mathrm{max} = 200$), and a batch size of 128. For post-pruning fine-tuning experiments, SGD was used with 100 epochs, an initial learning rate of 0.0001, a cosine annealing schedule ($T_\mathrm{max} = 100$), a weight decay of 5×10^{-5} and a gradient clipping of 0.1.

[1]Models adapted from https://github.com/kuangliu/pytorch-cifar

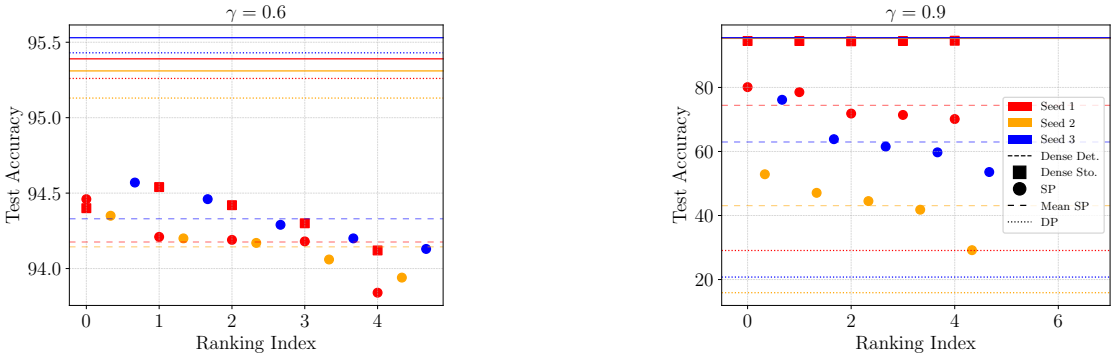
Section 3.4.1 analyses the one-shot performance of stochastic pruning compared to its deterministic counterparts for various γ values. To clarify the sources of its superior performance, the effect of σ and γ on stochastic pruning (SP) is measured by performing a mask transfer analysis. Section 3.4.2 performs a grid search over SP's hyperparameters, σ and γ, which illustrates that stochastic pruning is able to outperform deterministic pruning when optimal hyperparameter values are used. Section 3.4.3 uses a multi-objective evolutionary optimisation framework to explore the trade-offs between stochastic performance and γ, on the one hand, and between performance and performance gain (over deterministic models) on the other. Section 3.4.4 shows that there is a negative correlation between a phenomenon named feature Variance explosion (FVE) and SP models, suggesting that the reduction of FVE is the mechanism that gives SP its superior performance.

Finally, Section 3.4.5 presents the fine-tuning of the SP and DP solutions across different noise levels and pruning methods. The results suggest a trade-off between one-shot accuracy and model trainability.

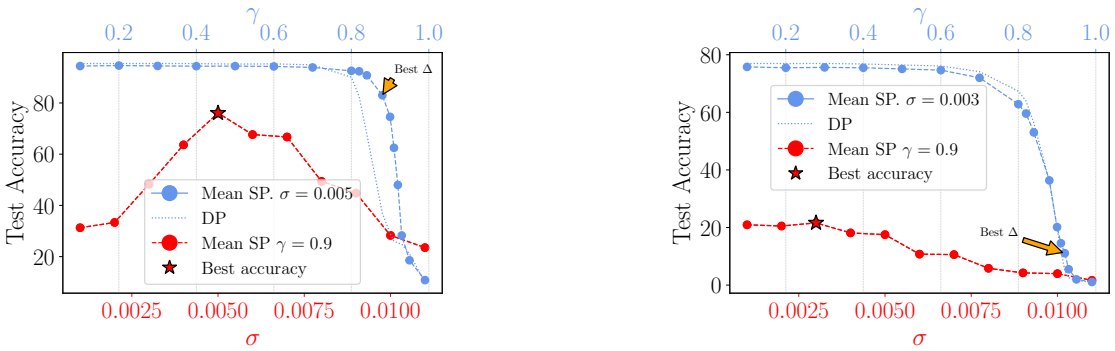
3.4.1 Good One-Shot Sparse Solutions

There is both theoretical (Li *et al.*, 2021) and empirical (Neelakantan *et al.*, 2015) evidence to highlight the vital role of noise during neural network training. Furthermore, LTH solutions are recoverable when the training has become stable to SGD noise (Frankle *et al.*, 2020a). Hence, it is possible to explore the local basin of a pre-trained network by injecting Gaussian noise, with the knowledge that this does not affect the probability of finding a good sparse solution (LTH) as by the late epochs of training, the network has already become stable to SGD noise. However, one would naively expect that random injection of noise into a highly trained network should not provide a performance advantage. Contrary to this intuition, SP generates better solutions than their deterministically pruned counterparts, for high pruning rates and appropriately chosen noise levels

In Figure 3.1a different ResNet18 seeds were trained in CIFAR10 for different pruning rates. All stochastic dense networks maintain performance similar to that of the original dense network.



(a) One-shot performance of multiple training runs.



(b) CIFAR10 (c) CIFAR100

Figure 3.1: **Unveiling Good One-Shot Sparse Solutions:** Using ResNet18 on CIFAR10/100 for:**a)** CIFAR10 test accuracy, comparing three ResNet18 seeds with SP parameters $\sigma = 0.005$ and $\gamma = \{0.6, 0.9\}$. Here, 3 different seeds of ResNet18 were trained. For each seed, 5 dense stochastic models (square points) were created by adding noise. The pruned stochastic models (SP) are denoted by round points. The effects of noise are reproducible across seeds and more prevalent for a large pruning rate. **b)** and **c)** show test set accuracy as a function of γ and σ, exploring values around optimal parameters (see Table 3.1). The arrow indicates the pruning rate yielding the best difference between SP and Deterministic Pruning (DP). The star indicates the noise level yielding the best performance.

For an intermediate pruning rate ($\gamma = 0.6$), individual stochastic models can sometimes achieve higher accuracy, but on average, SP performance is slightly worse than deterministic pruning, although the accuracy drop is similar ($\approx 1\%$) for both stochastic and deterministically pruned models.

As accuracy drops with higher pruning rates, SP begins to outperform deterministic models. At $\gamma = 0.9$, SP is by far superior to deterministic pruning. This advantage is revealed only when pruning is applied, since dense stochastic networks (coloured squares) perform similarly to the dense original (unperturbed pre-trained) models. The consistency of the results, and in particular the correlation of SP performance around the high pruning rates (which degrade accuracy), appears to indicate a general principle underpinning the success of SP. As the experiments demonstrate that SP is not limited to one seed, seed 1 was used for the results in the remainder of the chapter.

In Figures 3.1b and 3.1c, SP performance is presented as a function of σ and γ for the CIFAR10 and CIFAR100 test sets, respectively. For the CIFAR10 dataset, a discernible optimal noise level emerges around $\sigma = 0.005$. Above the crossover point ($\gamma \approx 0.8$), SP significantly outperforms the deterministic approach. This phenomenon is also observed with the CIFAR100 dataset, although the peak performance for σ and the disparity between SP and deterministic pruning (DP) are diminished, presumably because this disparity occurs for pruning rates with a considerably degraded performance, relative to the dense network. This can be attributed to CIFAR100 being a more difficult task.

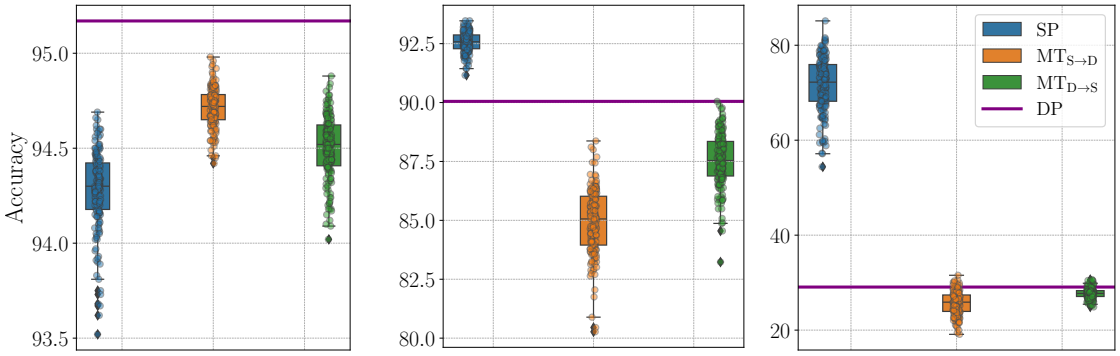
As SP is a two-step algorithm, the source of the improved one-shot accuracy is ambiguous. Each step suggests a possible factor: perturbation of the weights themselves and a stochastically generated mask. Then the question is whether stochastic weights $\mathbf{w}_\mathrm{p}$, stochastic mask $\mathbf{m}(\mathbf{w}_\mathrm{p})$, or a combination of both $\hat{\mathbf{w}} = \mathbf{m}(\mathbf{w}_\mathrm{p}) \odot \mathbf{w}_\mathrm{p}$ are responsible for enhanced performance. To this end, two operations are defined:

- $\mathrm{MT}_{\mathrm{S}\to\mathrm{D}}$: Apply the pruning mask from stochastic models to the original deterministic weights, i.e. $\mathbf{w}^* \odot \mathbf{m}(\mathbf{w}_\mathrm{p})$.
- $\mathrm{MT}_{\mathrm{D}\to\mathrm{S}}$: Apply the pruning mask from the deterministic model to the dense stochastic model, i.e. $\mathbf{m}(\mathbf{w}^*) \odot \mathbf{w}_\mathrm{p}$.

If operation $\mathrm{MT}_{\mathrm{S}\to\mathrm{D}}$ yields models with better performance than deterministic pruning, it would indicate that stochastic pruning can uncover effective masks. In contrast, if operation $\mathrm{MT}_{\mathrm{D}\to\mathrm{S}}$ yields models superior to deterministic pruning,

it indicates that the weights identified by stochastic pruning are superior to the deterministic weights in the pruned networks.

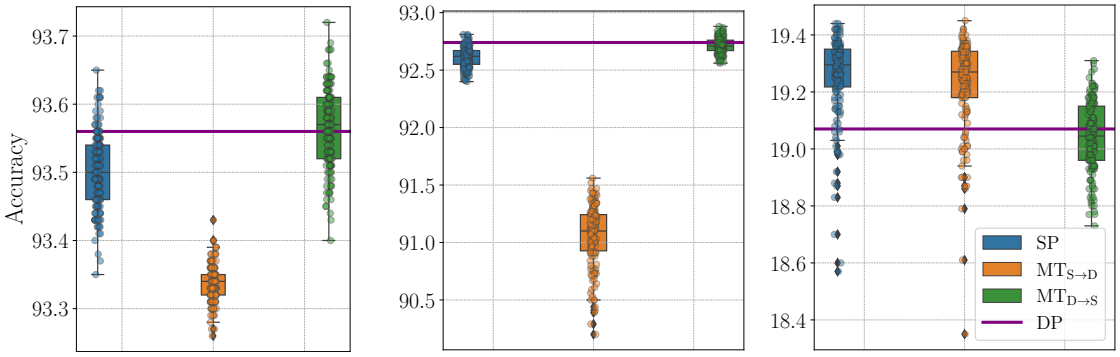
For this experiment, 80 stochastically pruned models for ResNet18 on CIFAR10 ($\sigma = 0.005$, Figure 3.2) were obtained. To test whether the architecture changes the results of stochastic pruning, or the contribution of the mask and weights, another 80 SP solutions using VGG19 were obtained, also tested on CIFAR10 ($\sigma = 0.001$, Figure 3.3). The pruning rates for VGG19 were selected such that both models have a similar number of non-zero weights.



(a) $\gamma = 0.5$ (b) $\gamma = 0.8$ (c) $\gamma = 0.9$

Figure 3.2: **Mask Transfer ResNet18 on CIFAR10:** SP one-shot performance draws from both the mask and stochastic weights, showing: SP (blue, left), $\text{MT}_{\text{S}\to\text{D}}$ (orange, middle) and $\text{MT}_{\text{D}\to\text{S}}$ (green, right). $\sigma = 0.005$. Purple line: deterministic pruning. For large pruning rates (≥ 0.8) SP reliably outperforms $\text{MT}_{\text{D}\to\text{S}}$ and $\text{MT}_{\text{S}\to\text{D}}$, which suggests that it is the combination of stochastic weights and stochastic masks that is responsible for the enhanced performance.

The results demonstrate that the high accuracy achieved by SP generalises across architectures, but that the relative contributions of the two operations vary. For ResNet18, stochastic weights and the stochastic masks (blue box plot) are both required to explain SP's accuracy. Neither operation alone outperforms deterministic pruning, but their combination yields the best accuracy (outperforming deterministic pruning for sufficiently high pruning rates). This phenomenon is particularly strong for extreme pruning rates ($\gamma = 0.9$). For VGG19, transferring the stochastic mask to the original unperturbed weights yields similar performance to



(a) $\gamma = 0.72$ (b) $\gamma = 0.88$ (c) $\gamma = 0.94$

Figure 3.3: **Mask Transfer VGG19 on CIFAR10:** SP one-shot performance generalises to different models, showing: SP (blue, left), $\text{MT}_{\text{S}\to\text{D}}$ (orange, middle) and $\text{MT}_{\text{D}\to\text{S}}$ (green, right). $\sigma = 0.001$. purple line: deterministic pruning. In this case, for large pruning rates (> 0.8) SP performs similarly to $\text{MT}_{\text{S}\to\text{D}}$ while both have better performance than $\text{MT}_{\text{D}\to\text{S}}$, this suggest that is the stochastic masks are responsible for the enhanced performance.

deterministic pruning for all three pruning rates tested, but the combined SP outperforms deterministic pruning only for extreme pruning rates ($\gamma = 0.94$), where the stochastic weights play a key role. This discrepancy in behaviour could be attributed to skip-connections present in ResNet18. It is important to note that the optimal noise levels (necessary to outperform deterministic pruning) differ across experiments, depending on both the model and the dataset.

In summary, in the high pruning regime, SP reveals effective sparse models that surpass deterministic pruning. There are examples in which this performance arises from the combination of stochastic weights and the stochastically derived mask (ResNet18), or where the mask itself is sufficient to produce the upgraded performance (VGG19).

3.4.2 Generalisability of Stochastic Pruning

The previous sections shows that SP provides promising solutions in the extreme pruning rate regime, for both ResNet18 and VGG19. The question of this section is whether and to what extent SP performance surpasses deterministic pruning across

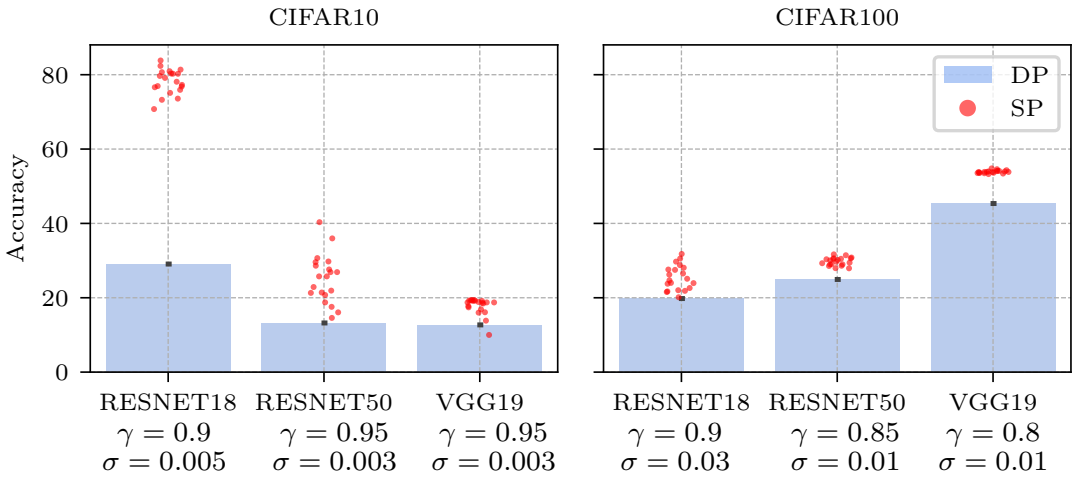


Figure 3.4: **SP One-Shot Performance Optimal Parameters (Table 3.1) for CIFAR10 and CIFAR100:** SP is able to find better pruned solutions than DP consistently for all dataset and model combinations.

3 models (adding ResNet50) and two datasets (CIFAR10, CIFAR100). A restricted grid search for the pruning rate was conducted with $\gamma = \{0.8, 0.85, 0.9, 0.95\}$ and $\sigma = \{0.001, 0.003, 0.005\}$ for each model-dataset combination (the best parameters are in Table 3.1). To this end, the median one-shot performance of five SP models was used. The results show that for all models and datasets, it is possible to find a combination σ and γ that generates SP models that are statistically superior to deterministic ones (all with p values< 0.05).

To ensure that this result applies robustly, 20 stochastically pruned models per model and dataset pair are considered. The results show that, with only rare exceptions, SP solutions consistently demonstrate superior performance relative to deterministic pruning (Figure 3.4).

3.4.3 Exploration of Hyperparameter space with Multi-Objective Evolutionary Optimisation

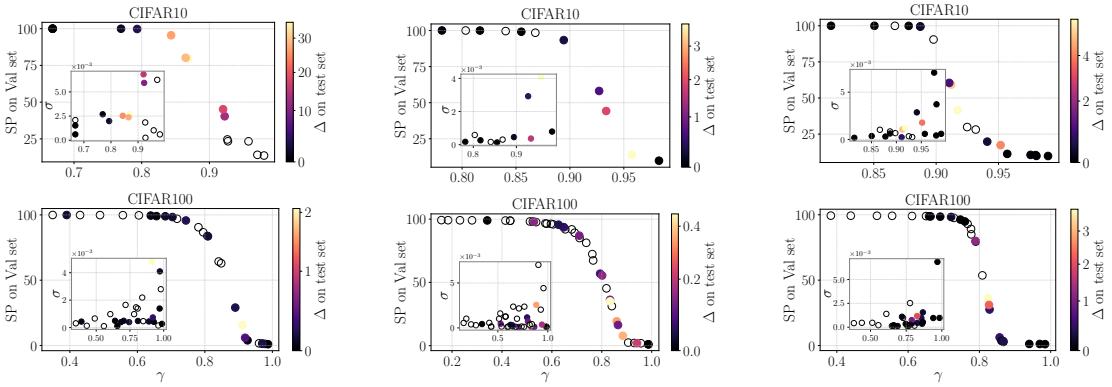
To more thoroughly explore the hyperparameter space, to more efficiently search for optimal hyperparameters, and to expand the search to additional objectives, a

Dataset	Model	Grid Search					MOO Search - F_2				
		γ	σ	SP	DP	Δ	γ	σ	SP	DP	Δ
CIFAR10	ResNet18	0.9	0.005	80.9	29.0	**51.9**	0.87	0.0038	88.4	39.7	**48.6**
	ResNet50	0.95	0.003	29.3	13.2	**13.1**	0.94	0.0028	33.9	16.3	**17.6**
	VGG19	0.95	0.003	19.3	12.7	**6.6**	0.91	0.0013	54.6	40.8	**13.7**
CIFAR100	ResNet18	0.9	0.003	26.3	19.7	**6.5**	0.92	0.0036	13.9	5.18	**8.7**
	ResNet50	0.85	0.001	30.5	24.9	**5.6**	0.76	0.0012	63.2	59.9	**3.3**
	VGG19	0.8	0.001	53.7	45.3	**8.4**	0.83	0.0025	28.4	16.5	**11.9**

Table 3.1: **Optimal Parameters for One-Shot SP for Grid and MOO Search:** All SP measures are the median of five models. The optimal γ are in the higher end, 0.8-0.95. The MOO search found similar parameters (σ and γ) to the grid search, which tells us that these regions of parameters are the most beneficial for SP.

Multi-objective Evolutionary algorithm, namely, NSGA-II (Deb *et al.*, 2002) was used.

NSGA-II is a genetic algorithm for multi-objective optimisation that maintains a population of solutions, using non-dominated sorting and crowding distance to select high-performing, diverse individuals. After a set number of generations, it outputs the non-dominated solutions (Pareto front) found in the final population. Two sets of functions were used consisting of two fitness functions ($F_1 = \{f_{11}, f_{12}\}, F_2 = \{f_{21}, f_{22}\}$). In the initial search (using F_1), solutions that maximise performance and pruning rates were prioritized, setting the objective function at $F_1 = \{f_{11}, f_{12}\} = \{\text{Acc}_{\text{validation}}(\hat{\mathbf{w}}), \gamma\}$, where $\text{Acc}_{\text{validation}}(\hat{\mathbf{w}})$ denotes the median performance of ten stochastic models in the validation set. Here, γ serves both as a decision variable and as an objective. This approach seeks to verify whether there exists a range of pruning rates that allows SP to outperform DP without explicitly optimising for this outcome. The second search uses the set of fitness functions defined by $F_2 = \{f_{21}, f_{22}\} = \{\text{Acc}(\hat{\mathbf{w}}), (\text{Acc}(\hat{\mathbf{w}}) - \text{Acc}(\mathbf{w})) \cdot \gamma\}$, where the two objectives are to maximise the median performance of SP and to maximise the disparity between stochastic and deterministic pruning ($\text{Acc}(\mathbf{w})$). The difference is multipied by the pruning rate to encourage solutions that demonstrate a pronounced difference between stochastic and deterministic pruning at



(a) ResNet18 (b) ResNet50 (c) VGG19

Figure 3.5: **Pareto Front $\mathbf{F_1}$:** Median SP performance ($n = 10$) versus γ in the validation set for F_1. Δ is calculated on the test set. There exist a pruning rate region in which SP surpasses DP even when is not optimised for this outcome. This region is concentrated in the transition region, from medium to high pruning rates. Bright colours indicate the greatest improvement which is measured in absolute values. Empty circles indicate models that performed better with DP than with SP.

elevated pruning rates.

Accuracies at search time were calculated using 5000 training images in all cases. The search space for both functions sets is $\sigma \in (0.0001, 0.01)$ and $\gamma \in (0.01, 0.99)$, 150 individuals were sampled in the hyperparameter space for both function sets and then, the Pareto Front was obtained from them. The combination ResNet18-CIFAR10 produces the highest improvement over DP in the test set, while all others have modest improvements, with maximum improvements between 0.4% and 5% (Figure 3.5). The Pareto fronts are S-shaped. These are to be expected, since large pruning rates invariably worsen one-shot performance. A more intriguing aspect of these figures is that the most substantial improvement over deterministic pruning in the test set is observed at high or extreme γ, between 85-94% for CIFAR10 and 75-95% for CIFAR100. The results show a wide range of pruning rates in which stochastic pruning can outperform deterministic pruning. This range is mostly confined to the pruning rates where performance starts to rapidly deteriorate, but does not render the network useless yet.

In Table 3.1 are the results for the parameters that are present in the Pareto front for F_2 and maximise f_{22}. Values are calculated over the test set. In particular, SP outperforms deterministic pruning in the high γ and low σ regimes, indicating that NSGA-II effectively identify hyperparameters where SP excels.

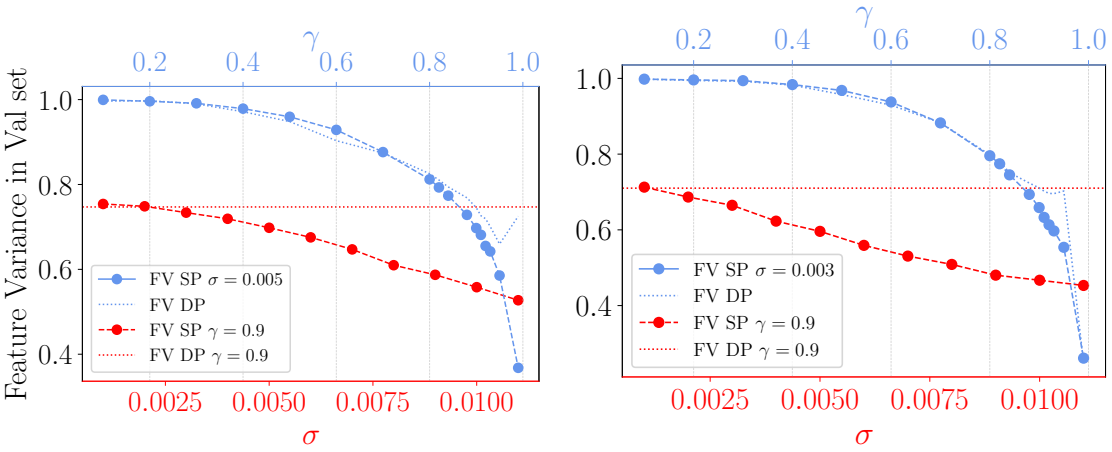
3.4.4 Feature Variance Explosion

Model	One-Shot				Fine-tuned			
	CIFAR10		CIFAR100		CIFAR10		CIFAR100	
	DP	SP	DP	SP	DP	SP	DP	SP
ResNet18	0.759	**0.698**	0.720	**0.675**	**0.999**	1.003	**1.000**	1.001
ResNet50	0.423	**0.401**	0.479	**0.477**	1.000	**0.972**	1.000	**0.999**
VGG19	**0.467**	0.470	0.739	**0.736**	1.000	**0.990**	**0.999**	1.000

Table 3.2: **Feature Variance Explosion:** Optimal grid search parameters were used (Table 3.1). For almost all model - dataset combinations, SP exhibits a lower feature variance than DP at one-shot. For the fine-tuned versions, almost all values are close to 1, since the pruned model, after fine-tuning, has similar dynamics to the original dense model.

Why does a simple procedure, such as introducing noise followed by pruning, substantially improve one-shot performance? Introducing noise to the weights increases the variance of the weight distribution, as $\mathrm{Var}(\mathbf{w} + e) = \mathrm{Var}(\mathbf{w}) + \mathrm{Var}(e)$ due to e being independent of $\mathbf{w}$, resulting in a bimodal distribution with more extreme weights after pruning. This distribution potentially enhances model accuracy by structuring pruned filters more effectively. This, I hypothesise, mitigates the *feature variance explosion*, a phenomenon characterised by a large discrepancy in the variance of feature maps throughout the network.

First, the variance of each neuron is measured across a batch of training data. Then, all the neurons variances are summed up across the layer. Finally, let this sum of the j^{th} layer be denoted $\phi_j(\cdot)$ for pruned models (either stochastic or deterministic) and $\Phi_j(\cdot)$ for dense models, the ratio $v_j = \frac{\phi_j(x_i)}{\Phi_j(x_i)}$ for each layer is



(a) CIFAR10 (b) CIFAR100

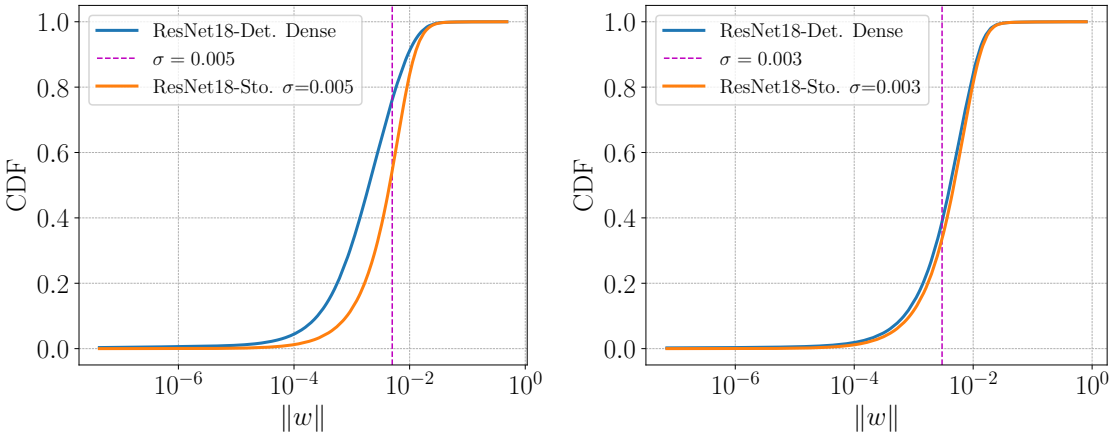
Figure 3.6: **Feature Variance Across σ and γ:** ResNet18 on CIFAR10/100. Note that SP mainly alleviates the feature variance explosion for high pruning rates (≥ 0.8)with σ fixed, and it decreases as σ increases.

calculated and whole model feature variance is calculated as follows:

$$V = \frac{1}{N\,L}\sum_{i}^{N}\sum_{j}^{L} = \frac{\phi_j(x_i)}{\Phi_j(x_i)} = v_j(x_i), \tag{3.4}$$

where N is the number of batches and L is the number of layers in the network. In the case of stochastic models, $\phi_j(x_i)$ is calculated for five different stochastic models (obtained by applying noise to the original model 5 times) before averaging these results to obtain a single $\bar{\phi}_j$. This average then is divided by the variance of the dense stochastic model, Φ_j, for a particular layer. For most combinations, the stochastically pruned model has a lower value of V (see Table 3.2), suggesting that alleviating feature variance explosion is a putative mechanism for enhancing one-shot performance in SP. When these models are fine-tuned, nearly identical values of V across the board, which is expected since the internal dynamics of the pruned network resembles those of a trained network. Furthermore, in Figure 3.6 it is illustrated that SP mainly alleviates the feature variance explosion for high pruning rates ($\geq$ 0.8), that is, a regime in which the extreme sparsity of the network due to pruning results in a performance degradation. In this regime, the surviving weights matter more, and the effect of injecting noise (to the weights

and/or to the mask) is expected to be highly deleterious. Indeed, in Figure 3.7a the Cumulative Density Function (CDF) shows that for ResNet18 on CIFAR10, $\sigma = 0.005$ is larger than more than half of the network weights, suggesting that a substantial portion of the model is overshadowed by the noise. It is therefore no surprise that a large pruning rate is required to see an effect on feature variance. For ResNet18 on CIFAR100, the optimal noise level $\sigma = 0.003$ is larger than 40% of the weights of the model.



(a) CIFAR10 (b) CIFAR100

Figure 3.7: **Cumulative Distribution of Dense ResNet18 Weights trained on CIFAR10/100:** For a $\sigma = 0.005$ more than half of the network's weights are smaller than the noise amplitude, suggesting that a substantial portion of the model is overshadowed by the noise. For CIFAR100, the proportion of weights that are smaller than the optimal σ is 40%, which is still a considerable percentage but not as high as for CIFAR10.

3.4.5 Fine-Tuning Stochastic Solutions

So far, this chapter has explored the one-shot regime of stochastically pruned models. This section explores the robustness of fine-tuned solutions to stochastic pruning. To this end, the behaviour of stochastic GMP and a closely related but more subtle pruning method, LAMP (Lee *et al.*, 2022) is analysed. LAMP is selected for comparison because of its status as a more sophisticated pruning

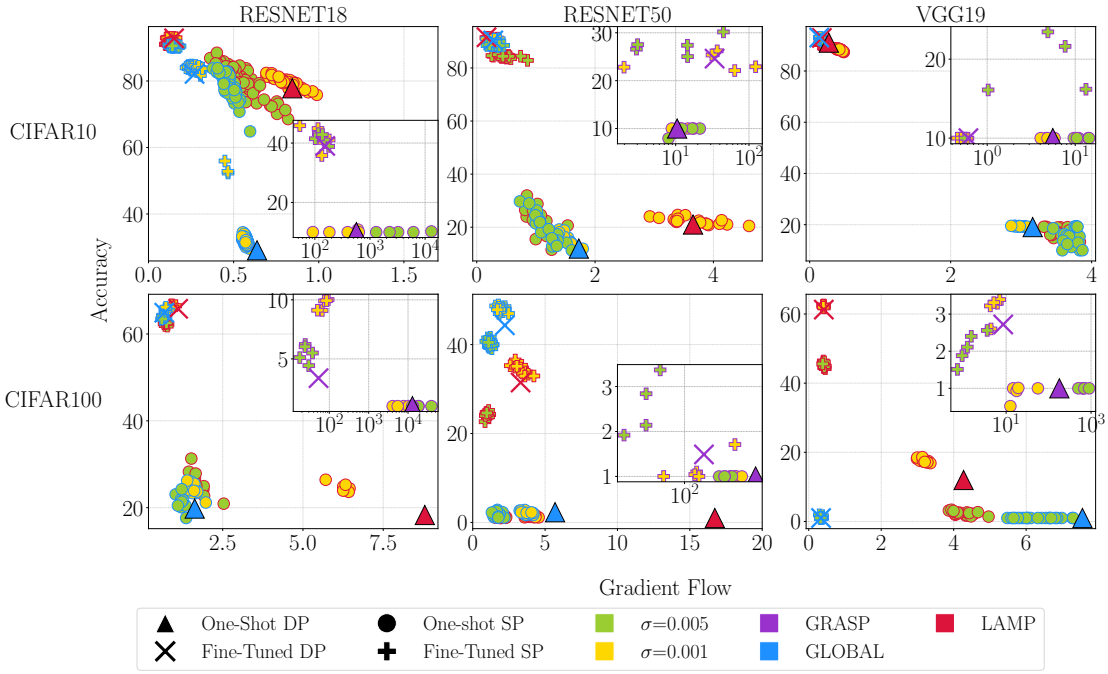


Figure 3.8: **Fine-tuned Accuracy and Gradient Flow for All Datasets and Architectures:** $\gamma = \{0.9, 0.95, 0.94\}$ for ResNet18, ResNet50 and VGG19 respectively to ensure a similar number of parameters between these architectures. On average, increased one-shot performance suppresses gradient flow. High-σ models outperform DP, but low-σ models are better after fine-tuning.

method that nevertheless employs the weight magnitude to compute the saliency score, thereby maintaining computational efficiency. The analysis focusses on the trade-offs between trainability and performance (Table 3.3 and Figure 3.8), using gradient flow (GF), a critical metric of trainability for sparse models (Evci *et al.*, 2022), often used as a proxy of their trainability. The results show that fine-tuned SP does not perform well with LAMP.

Dataset	Model	LAMP								GMP								GRASP							
		One-Shot					Fine-Tuned			One-Shot					Fine-Tuned			One-Shot					Fine-Tuned		
		SP		DP		Δ_{OS}	SP	DP	Δ_{FT}	SP		DP		Δ_{OS}	SP	DP	Δ_{FT}	SP		DP		Δ_{OS}	SP	DP	Δ_{FT}
		GF	Acc	GF	Acc					GF	Acc	GF	Acc					GF	Acc	GF	Acc				
CIFAR10	ResNet18	9.9	84.1	18.9	77.8	**6.3**	91.9	92.8	-0.9	8.0	80.9	11.5	29.0	**51.8**	91.8	89.6	**2.2**	$4.6{\cdot}10^3$	10.0	11.9	10.0	0.0	43.9	47.0	-3.1
	ResNets50	35.9	28.4	89.6	22.4	**6.0**	88.8	91.2	-2.4	16.0	29.3	19.1	13.2	**16.1**	90.6	90.6	0.0	12.0	10.0	37.9	10.0	0.0	24.8	25.5	-0.7
	VGG19	62.9	20.4	8.7	88.8	-68.4	91.1	92.6	-1.5	46.0	19.3	46.2	12.7	**6.6**	92.7	92.3	**0.4**	5.8	10.0	3.9	10.0	0.0	10.1	10.0	**0.1**
CIFAR100	ResNet18	54.4	26.1	100.1	18.3	**7.8**	66.8	67.6	-0.8	18.4	26.3	27.0	19.7	**6.6**	67.6	66.9	**0.7**	$2.1{\cdot}10^4$	1.0	$1.3{\cdot}10^5$	1.0	0.0	10.3	10.0	**0.3**
	ResNets50	23.4	38.7	29.5	30.0	**8.7**	72.4	72.1	**0.3**	26.6	30.5	27.1	24.9	**5.6**	74.1	73.9	**0.2**	$1.8{\cdot}10^4$	1.0	$7.4{\cdot}10^4$	1.0	0.0	1.1	1.6	-0.5
	VGG19	6.6	69.3	7.3	68.8	**0.5**	71.9	72.0	-0.1	18.4	53.7	23.3	45.3	**8.4**	72.4	72.2	**0.2**	307.0	1.0	766.7	1.0	0.0	6.0	3.2	**2.9**

Table 3.3: **Accuracy and GF for One-Shot and Fine-Tuned:** This table uses the optimal grid search parameters σ and γ in Table 3.1. Here, SP surpasses DP in the fine-tuned regime only when GMP is used. For LAMP, the one-shot accuracy (both for SP and DP) is higher than that of GMP, which also translates to a smaller Δ_{OS} and, in some cases, negative Δ_{OS} (VGG19 on CIFAR10). Also, GRASP has low one-shot accuracy and low fine-tuned accuracy, despite having a large GF. This suggests that arbitrarily large GF at initialisation does not necessarily translate into better fine-tuned solutions.

Why does SP with LAMP not perform as well after fine-tuning? One possibility is that DP for LAMP achieves remarkably high performance compared to DP with GMP, leaving little room for improvement. This high bar, combined with the relatively poor one-shot performance, is therefore insufficient to outperform DP. Note that fine-tuned SP with GMP tends to perform better than fine-tuned SP models with LAMP for larger models (Table 3.3). Once again, this can be attributed to a combination of one-shot performance and the effect of fine tuning. For GMP, a one-shot boost in performance can help the stochastic model outperform the deterministic model despite presenting a smaller GF (ResNet18-CIFAR10 in Figure 3.8). In contrast, for LAMP (with $\sigma = 0.005$), the initial increase in accuracy is insufficient for fine-tuned models to outperform the deterministic model.

In general, Figure 3.8 reveals a similar inverse relationship for both LAMP and GMP: One-shot models with higher one-shot accuracies tend to have lower GF, suppressing trainability during fine-tuning, and hence losing their advantage over DP, and even risking lower performance. The high level of noise in these models ($\sigma = 0.005$) is therefore detrimental to finding competitive fine-tuned solutions. However, models generated with more modest noise levels ($\sigma = 0.001$) maintain their advantage and can even outperform DP models after fine-tuning.

One question that arises from these observations is whether, given a set level of accuracy, maximising gradient flow is always beneficial. To test this hypothesis, GraSP was used (Wang *et al.*, 2019a), which explicitly preserves gradient flow during pruning. Using GraSP, the GFs obtained are orders of magnitude larger, up to $O(10^4)$ (Figure 3.8 and Table 3.3). Not only do such high GF fail to enhance trainability, but in fact, they suppress it. The results suggest that optimising GF is only fruitful for GF$<$ 10. Note that as GraSP was designed to operate at initialisation (before training), its unbounded maximisation of GF improves trainability in random networks, but not in pre-trained ones.

3.5 Conclusion and Discussion

This chapter shows evidence that SP provides a simple alternative to deterministic pruning. Its effectiveness is evidenced in the one-shot and fine-tuning regime

at high pruning rates (> 0.8) when using GMP. In the best case (ResNet18-CIFAR10), SP achieves improvements of 51.9% for one-shot and 2.2% for the fine-tuned regime over deterministic pruning. In general, SP alleviated the phenomenon of feature variance explosion, thereby normalising the network's internal statistics, similar to how batch normalisation works (Ioffe & Szegedy, 2015).

From the perspective of the loss landscape, one interpretation is that SP perturbations facilitate beneficial shifts to reach better local minima since their perturbations are small enough to not reach outside the basin (pruning solutions obtained from a dense solution are robust to small perturbations (Evci *et al.*, 2020b)) and that solutions tend to land in the periphery of local optima (Izmailov *et al.*, 2019), perturbing via addition of Gaussian noise is a cost-effective way of circumventing this suboptimal convergence.

One limitation of this chapter's approach is the use of unstructured pruning. Despite the availability of modern libraries for unstructured, sparse matrix multiplication (Okanovic *et al.*, 2024; Yan *et al.*, 2024), structured pruning is preferred by deep learning practitioners due to its immediate hardware compatibility. Removing entire filters reduces the number of matrix multiplications, thereby physically shrinking the network architecture and yielding direct memory savings. Consequently, the benefits claimed in this chapter would be difficult to realise in practice without the use of specialised libraries.

This chapter shed light on the trade-offs of different pruning rates, noise levels, and pruning algorithms, in the one-shot and fine-tuning settings. The results suggest that feature variance explosion is a concern for post-training pruning. Previously, Li *et al.* (2020) demonstrated that the adjustment of batch normalisation layer running statistics using a small subset of data significantly improves one-shot performance, which is strongly correlated with final fine-tuning accuracies. Taken together with the results of this chapter, this suggests the fundamental importance of understanding feature variance and controlling it during pruning.

Future work can explore the validity of this findings for visual transformer architectures (Dosovitskiy *et al.*, 2022), since these architectures rely on large models, they are a prime candidate for compression.

Chapter 4

Larger Receptive Fields Improve Pruning Performance in CNNs

4.1 Introduction

A fundamental characteristic of both biological and artificial visual systems is the receptive field, the region of input space to which a neuron or unit responds. In biological vision, receptive fields are central in shaping the way information is processed (Hubel & Wiesel, 1962; Lehky & Sejnowski, 1988). Modern theories propose that feedback connections from higher to lower-order visual cortical areas carry predictions of lower-level neural activities (Rao & Ballard, 1999), suggesting that receptive fields play an active role in predictive processing that supports recognition and decision-making.

The first receptive fields identified in the primary visual cortex (V1) of mammals were those of simple and complex cells (Hubel & Wiesel, 1962). Simple cells, with relatively small receptive fields, exhibit distinct "ON" and "OFF" regions arranged in elongated patterns, responding strongly to edges or lines at precise positions. Their responses can be approximated by adding the inputs in their receptive field. Complex cells, by contrast, lack clearly defined ON/OFF regions, are less sensitive to exact stimulus position, and display direction selectivity. Importantly, their responses are non-linear and cannot be predicted by linear summation. Together, these receptive fields form a hierarchy: complex cells build upon

the outputs of simple cells, yielding increasingly abstract representations. Even in these early findings, receptive field size was observed to increase with cortical depth, with complex cells having larger receptive fields than simple cells (Hubel & Wiesel, 1965).

The influence of receptive fields extends beyond low-level feature extraction to higher-level encoding of spatial and hierarchical information. For example, Golomb & Kanwisher (2012) demonstrated that high-level visual representations in the human brain encode the position of objects retinotropically (relative to the eye), rather than spatiotopically (relative to the world), underscoring how receptive fields shape representational frameworks.

An analogous principle applies to convolutional neural networks (CNNs), where receptive fields govern how features are extracted, integrated, and represented across layers (Koutini *et al.*, 2019; Seif & Androutsos, 2018; Tan & Le, 2019; Zhou *et al.*, 2015). However, not all pixels within a receptive field contribute equally. Luo *et al.* (2016) showed that CNNs naturally develop foveal-like representations, where responses are concentrated in a Gaussian-distributed effective receptive field that expands during training. Beyond feature integration, receptive fields are also linked to sparse encoding in visual processing.

Olshausen & Field (1996) argued that natural images exhibit a sparse structure, which means that any given image can be represented using only a small subset of basis descriptors. In particular, they represented images as a linear superposition of (not necessarily orthogonal) basis functions $\phi_i(\cdot)$. That is, an image is represented as $I(x, y) = \sum_i a_i \phi_i(x, y)$. They formalised this with a sparse coding algorithm that penalised representations requiring many non-zero coefficients ($a_i \neq 0$), thereby favouring compact encodings. Applied to natural image patches, this framework yielded receptive fields with striking similarities to those in mammalian V1: spatial localisation, orientation selectivity, and multi-scale sensitivity. This suggests that sparse coding is an emergent property of receptive field organisation.

Sparsity also has practical significance for CNNs, where computational costs increase dramatically with model width and depth. Comparisons between species underscore this: Optimal computational models of mouse visual cortex models favour shallower architectures and low-resolution, general-purpose representations,

reflecting the limited energetic resources of the mouse brain, unlike the deeper, high-resolution visual systems of primates (Nayebi *et al.*, 2023). In particular, the human brain operates continuously at only ∼20W (Sokoloff, 1960), a fraction of the 300–700W consumed by modern CNNs on GPUs during training and inference (Massed Compute, 2025). Inspired by such efficiency, this chapter investigates how the size of the receptive field influences the compression capacity of a model.

Given that receptive fields shape feature extraction, representation, and sparsity, a key question arises: How does receptive field size affect the prunability of a network? This chapter addresses this question by examining the interaction between the size of the receptive field and the prunability of CNNs. Specifically, how larger receptive fields improve the robustness of pruned networks and how network redundancy (the extent to which layers contribute to classification accuracy) modulates pruning performance. Throughout this chapter, one-shot pruning accuracy refers to the accuracy immediately measured after pruning, without fine-tuning. The contributions of this chapter are as follows.

- The results show that increasing the size of the receptive field, by increasing the size of the kernel in the max-pooling layer, maintains or improves the accuracy of the pruned model, in contrast to the decline observed with smaller receptive fields, even after fine-tuning (see Section 4.3.1).
- The results show that this improvement can be explained by the reduced saturation of the deeper layers (see Section 4.3.2). As receptive fields enlarge, deeper layers become increasingly redundant, enhancing prunability.
- The results show that for ResNet-type architectures, the redundancy in deeper layers is correlated by an increase in dissimilarity of the representations on the network (Section 4.3.3)
- Stochastic Pruning and receptive field modulation are combined to further increase the accuracy gains for pruned networks (Section 4.3.4).

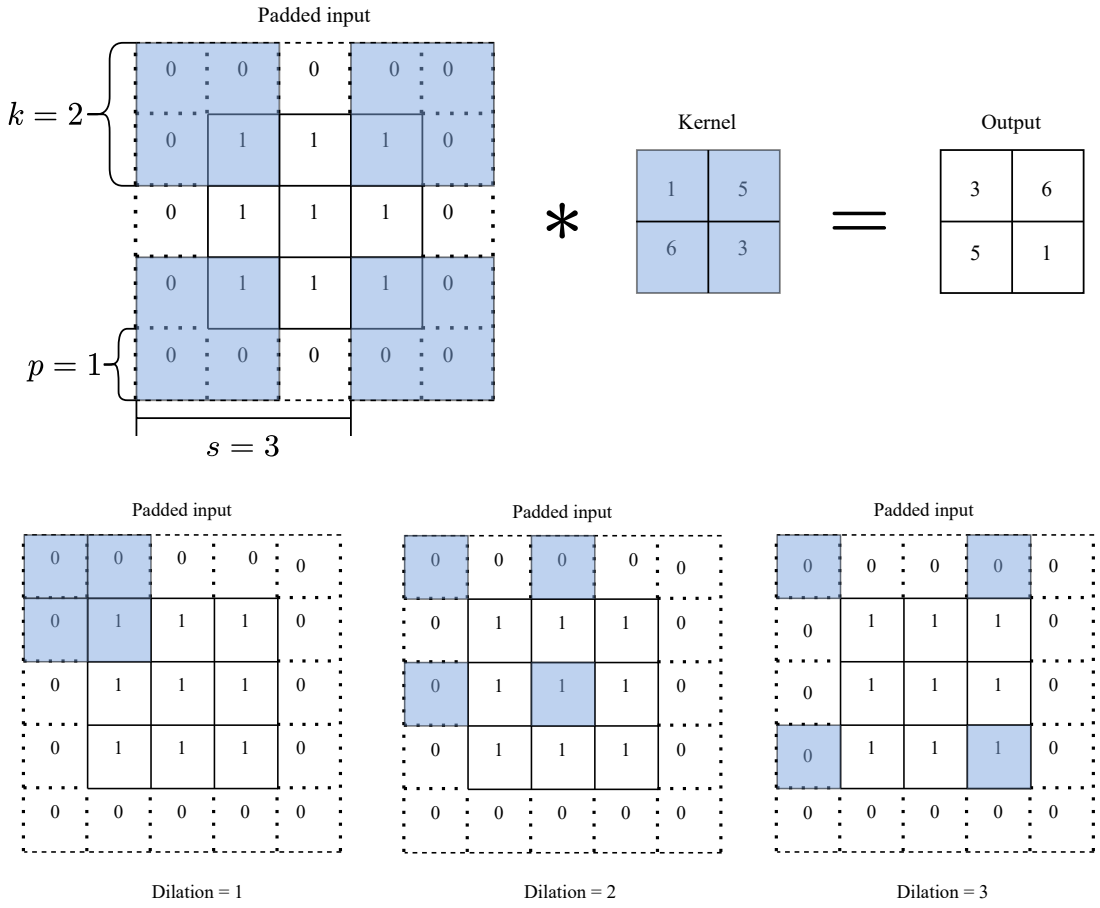


Figure 4.1: **Convolution General Concepts Schematic:** In this schematic are depicted 4 different concepts in the convolution operation of CNNS, padding p, kernel size k, stride s and dilation d. Padding, represented by the dashed lines, is the number of additional pixels added to the image (solid lines) to control the size of the output feature map and prevent information loss at the borders. The kernel size is the dimension of the convolutional filter. The stride is the step size by which the kernel moves across the input, and the dilation is the spacing between the filter's weights when the filter is applied to the image.

4.2 Related Work

In CNNs, each feature in a given layer perceives only a specific region of the input; this region is called the receptive field, and understanding it is vital for effective

model design and interpretation. For single-path architectures such as AlexNet and VGG, the receptive field can be calculated in closed form.

Following Araujo *et al.* (2019)'s notation, consider a fully convolutional network (FCN) with L layers, $l = 1, 2, \ldots, L$. Define the feature map $f_l \in \mathbb{R}^{h_l \times w_l \times d_l}$ to denote the output of the l-th layer, with height h_l, width w_l, and depth d_l. The input image is denoted by f_0 and the final output feature map by f_L. For simplicity, only one-dimensional input signals and feature maps are considered.

Define r_l as the receptive field size of the final output feature map f_L with respect to the feature map f_l. In other words, r_l corresponds to the number of features in feature map f_l that contribute to generating one feature in f_L. Note that $r_L = 1$. For example, consider a model with two sequential convolutional layers with kernel size k and stride s (see Figure 4.1 for an illustration of these concepts). The receptive field relationship is:

$$r_1 = s_2 \times r_2 + (k_2 - s_2)$$

where r_1 is the receptive field of f_2 over the features of f_1 and $r_2 = 1$ since f_2 is the last layer.

Araujo *et al.* (2019) derived the following recursive equation to calculate the receptive field of a single-path convolutional model:

$$r_0 = \sum_{i=0}^{L} \left((k_i - 1) \prod_{j=1}^{i} s_j \right) + 1 \tag{4.1}$$

where r_0 is the receptive field over the input image f_0 of the last convolutional feature map f_L.

Beyond single-path architectures, multi-path models such as ResNet or Inception require additional considerations. In these architectures, multiple paths connect to a single feature at a given layer, and these paths are not aligned in their receptive fields from the input. Thus, when multiple inputs combine (e.g., skip connections), one must compute the receptive field of each path separately and then take the union (the minimal start and maximal end positions) (Araujo *et al.*, 2019).

An alternative method to calculate the receptive field uses backpropagation. This approach performs a single forward pass on an input of all ones, followed

by a backpropagation step with respect to this input. For the backpropagation step, a mask is used with only those in the spatial centre of the first channel of the last convolutional layer. All non-zero elements of the gradient with respect to the input then constitute the receptive field of the last convolutional layer. This approach for receptive field calculation was used in this chapter, as it is the most straightforward method and can be applied to both single-path and multi-path architectures. An open-source library[1] was used to calculate the receptive field of the models.

Next, prior work on characterising receptive fields in CNNs, their optimal configurations, and their broader significance in deep learning architectures is reviewed.

Characterising Receptive Fields in CNNs. Luo *et al.* (2016) introduced the concept of the *Effective Receptive Field* (ERF), which quantifies the region within the theoretical receptive field that significantly influences the output of a unit. They found that the ERF is smaller than the theoretical receptive field and expands as training progresses, highlighting the impact of subsampling and dilation on its growth. Kobayashi & Shouno (2020) further explored the structure of receptive fields in ResNets, identifying orientation-selective and double-opponent colour neurons by analysing the preferred stimulus of specific neurons.

Optimal Receptive Field. Gao *et al.* (2023) investigated whether the optimisation of the receptive field could be automated. To that end, the authors use a global-to-local search scheme. They find coarse combinations via a global search, followed by an expectation-guided iterative local search to refine them. They controlled receptive fields through varying dilation rates (see Figure 4.1 for a visual representation of dilation in CNNs) and developed an evolutionary search strategy to dynamically refine these configurations. The main objective of this chapter is not to search for an optimal receptive field for pruned accuracy, but to understand the relationship between the two. Future work would focus on exploiting this relationship to search for an optimal receptive field for pruning. Furthermore,

[1] https://github.com/jotaf98/easy-receptive-fields-pytorch

Richter *et al.* (2021a) shows that the performance of CNNs is significantly influenced by input image size, regardless of image detail. They demonstrate that an upscaled version of a low-resolution image can partially recover lost performance, thereby contradicting the hypothesis that accuracy in high-resolution images is driven by greater detail. Their research reveals that, for a given model and receptive field, there is an input size at which the model performs best. Based on this, Richter *et al.* (2022) analysed the expansion of the receptive field across layers to identify redundant layers that do not contribute to test performance. Their proposed method, Receptive Field Analysis (RFA), enables the estimation of feasible input resolutions to optimise network performance. Other studies have examined receptive field selection from various perspectives. Coates & Ng (2011) proposed a similarity-based metric for grouping feature patches that exploits local receptive fields in scenarios with limited prior knowledge. From these works, only Richter *et al.* (2021a, 2022) deal with receptive field and reduction of network's size simultaneously by removing entire layers of the models to adjust the model to a particular resolution. Because the model's capacity may be reduced by reduced depth, this thesis employs pruning instead. Additionally, this thesis uses the layer saturation measure (Equation (2.2)) developed by the authors to assess the prunability of layers in the model.

Importance of Receptive Field. Koutini *et al.* (2019) observed that ResNet architectures, despite outperforming VGG networks on general vision tasks, suffer from overfitting in the small-scale Acoustic Scene Classification (ASC) setting. They attributed this discrepancy to the receptive field of ResNets and demonstrated that optimising the receptive field size significantly impacts accuracy, often more than the number of parameters or network depth.

Similarly, Wang *et al.* (2020) investigated the relationship between the size and depth in Single Image Super-Resolution (SISR), where the goal is to recover a high-resolution image from its corresponding single low-resolution image. This problem is ill-posed due to information loss during the high-resolution-to-low-resolution transformation, rendering the high-resolution counterpart of a low-resolution image non-unique.

Their findings highlight the interplay between depth and receptive field size, showing that an appropriately sized receptive field can compensate for the limita-

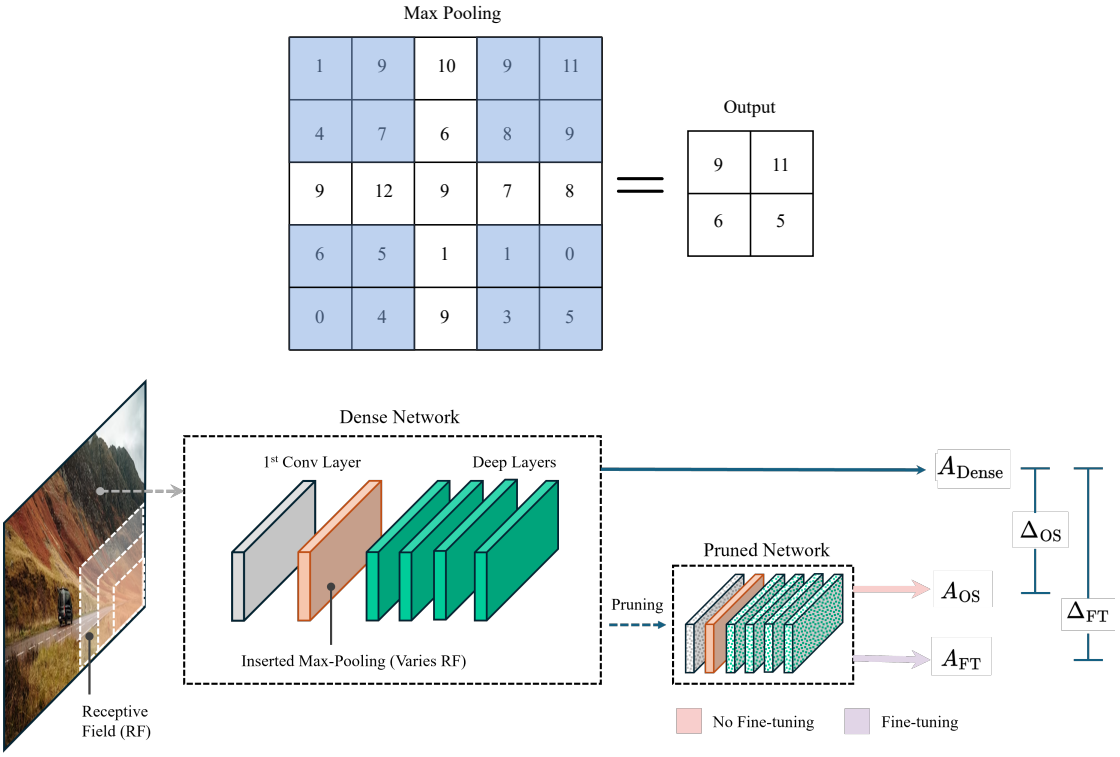


Figure 4.2: **Schematic Explanation of Receptive Field Manipulation Experiments.** A max-pooling layer (up) is inserted after the first convolutional layer in each architecture, adjusting the kernel size and stride to generate different receptive field levels. In the figure (down), A_{Dense}, A_{OS}, and A_{FT} represent the accuracy of the dense network, the pruned network without fine-tuning (one-shot), and the fine-tuned pruned network, respectively.

tions of insufficient model depth (non-linearity). In contrast, simply increasing the model depth without adjusting the receptive field does not improve performance. This suggests that the impact of receptive field size is highly context-dependent and interacts synergistically with other network design choices.

4.3 Experimental Results

This section investigates the relationship between receptive field size and pruning performance in CNNs. Controlled experiments are designed to systematically anal-

yse this. These experiments modify the receptive field while keeping the number of parameters fixed across models.

This section is designed to conduct controlled experiments in which the receptive field is modified while keeping the number of parameters fixed across models, to systematically analyse this relationship. This ensures that any observed differences in pruned accuracy arise directly from receptive field variations rather than changes in model capacity.

A range of CNN models is trained, with the receptive field adjusted by adding a max-pooling layer after the initial convolutional layer. Varying the kernel size and stride of this layer creates different receptive field sizes while maintaining the same number of model parameters. The stride is set to the kernel size minus one to ensure that the pooling operation spans the entire feature map. Then, Global Magnitude Pruning (GMP) (Gupta *et al.*, 2024) is applied to prune the networks with various receptive field sizes. The accuracy of pruned networks is compared with and without fine-tuning (one-shot). This approach is illustrated in Figure 4.2.

For statistical validation, experiments are performed on three architectures: VGG, ResNet50, and a custom ResNet25, which is designed to have a reduced receptive field while maintaining depth. Three datasets were used: CIFAR10, Tiny ImageNet, and Small ImageNet. The Small ImageNet dataset mirrors Tiny ImageNet in terms of class structure and sample count, but contains high-resolution images sourced from ImageNet. Each architecture-dataset-receptive field combination is trained five times using the Stochastic Gradient Descent (SGD) optimiser with an initial learning rate of 0.1, a cosine annealing schedule, a momentum of 0.9, and a weight decay of 5×10^{-5}. Models are trained for 200 epochs unless stated otherwise, using an Nvidia P100 GPU. This study is structured to examine two key scenarios. First, pruning performance is evaluated when the receptive field is larger than the input image, a common setting in CNN-based classification tasks. Subsequently, cases where the receptive field is smaller than the input image are analysed using ResNet25 and Small ImageNet. In both scenarios, GMP is employed as default pruning method.

Section A.5 presents the results of preliminary experiments using dilation and padding to ensure that the size of the feature map remains constant in the receptive fields. This tests whether a constant information density across receptive fields

would change the results observed in this chapter. These preliminary results show that when the receptive field is altered via dilation while keeping the information density constant, the results differ from those obtained in the next section. This suggests that the information density within the network (i.e., the size of the feature map) drives the pruning results in this chapter.

4.3.1 Large Receptive Field Prevents Accuracy Drop from Pruning

This section explores if expanding the receptive field alleviates the decline in accuracy due to pruning. Two critical cases are examined for this analysis: one where the receptive field exceeds the size of the input image and another where it is smaller.

Receptive Field Larger than the Input Image Regime

Popular CNN architectures used in classification benchmarks have receptive fields that are larger than the resolution of the input (larger than 224, the standard ImageNet resolution), such as Inception (Szegedy *et al.*, 2015), MobileNet (Howard *et al.*, 2017; Sandler *et al.*, 2019), ResNet-101 (He *et al.*, 2015), and Inception-ResNet (Szegedy *et al.*, 2016) (see Araujo *et al.* (2019) for the computation of receptive fields for these models).

Furthermore, there is evidence suggesting a positive logarithmic relationship between receptive field size and network accuracy, meaning that models with better performance tend to have larger receptive fields (Araujo *et al.*, 2019). However, these results should be interpreted with caution, as model performance is multifactorial and does not depend solely on the receptive field.

Given that practitioners are likely to choose models with receptive fields larger than the input image in their application, this study explores this regime by modulating receptive fields. Specifically, this section ask the question, how does receptive field modulation affect the prunability of models?

To examine the relationship between receptive field size and network sparsity, the accuracy of both dense and pruned neural networks was evaluated across different receptive field configurations. In this regime, experiments were conducted

using two model architectures (VGG and ResNet50) on two datasets (CIFAR10 and Tiny ImageNet).

The pruning process was performed using a default pruning rate of 0.9 (Table 4.1), with additional experiments for different pruning rates provided in Section A.1. The accuracy of pruned models before and after fine-tuning are compared. By default, models were fine-tuned for 10 epochs, as this was sufficient to stabilise performance, particularly on CIFAR10. Extended fine-tuning results for 100 epochs are available in Table A.3 in the appendix.

The results in Table 4.1 and Section A.1 reveal several key findings. First, the effect of receptive field only appears for pruning levels that lead to a large drop in accuracy (Δ_{OS}). For both models on both datasets, this occurs at any pruning rate greater than or equal to 0.8. What constitutes a large Δ_{OS} is dataset and model dependent. For VGG on CIFAR10, this means $\Delta_{OS} \approx 20$, while for ResNet50 on CIFAR10 it means $\Delta_{OS} \approx 2$. Similarly, for VGG on Tiny ImageNet this corresponds to $\Delta_{OS} \approx 40$, while for ResNet50 it means $\Delta_{OS} \approx 20$.

Second, for all dataset and model combinations (VGG and ResNet50 trained on CIFAR10 and Tiny ImageNet), increasing the receptive field decreases the difference between pruned one-shot accuracy and dense accuracy ($A_{\text{Dense}} - A_{\text{OS}} = \Delta_{OS}$). In extreme cases, models do not lose accuracy when pruned, achieving $\Delta_{OS} = 0$.

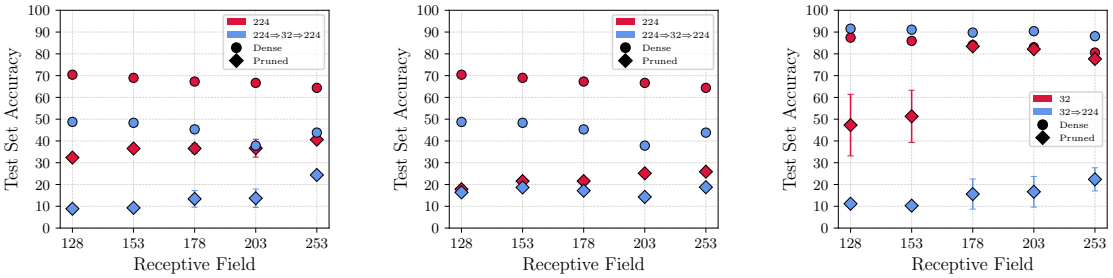
Third, the optimal receptive field for fine-tuning depends on the model and dataset used. For CIFAR10, the best pruned accuracy after fine-tuning (A_{FT}) is achieved with a small receptive field (181 for VGG and 213 for ResNet50). In contrast, for Tiny ImageNet, the opposite holds: the best pruned A_{FT} is achieved with larger receptive fields (537 for VGG and 423 for ResNet50). This suggests that the discriminating features in Tiny ImageNet images are larger than those in CIFAR10, possibly due to the difference in input resolution (64×64 for Tiny ImageNet compared to 32×32 for CIFAR10) (Richter *et al.*, 2021a).

In summary, while dense networks achieve optimal accuracy with small receptive fields, pruned networks generally benefit from intermediate or large receptive fields. This observation is both intriguing and practically significant, as models with larger receptive fields offer enhanced computational and memory efficiency through two factors: smaller feature maps and reduced network size from pruning.

As a result, pruned networks with large receptive fields can achieve comparable performance to denser models (with lower pruning rates) that require greater computational resources.

	RF	CIFAR10					Tiny ImageNet				
		Dense	No Fine-Tuning		Fine-Tuned (10 epochs)		Dense	No Fine-Tuning		Fine-Tuned (10 epochs)	
			$A_{OS} \uparrow$	$\Delta_{OS} \downarrow$	$A_{FT} \uparrow$	$\Delta_{FT} \downarrow$		$A_{OS} \uparrow$	$\Delta_{OS} \downarrow$	$A_{FT} \uparrow$	$\Delta_{FT} \downarrow$
VGG	180*	**94.0±0.24**	10.04 ±0.160	83.9±0.318	90.2± 0.276	3.42 ± 0.683	**62.5±0.30**	0.92±0.32	61.6± 0.26	13.8 ± 12.9	48.6± 13.0
	181	93.5±0.11	10.9 ± 2.03	82.6 ±2.02	**91.42±2.03**	2.106±2.11	61.5±0.33	0.75± 0.09	60.8 ±0.32	30.21±2.60	31.37±2.53
	359	91.1±0.23	32.4 ±15.7	58.7 ±15.8	90.81±0.29	0.348±0.37	53.2±0.20	0.63± 0.17	52.6 ±0.36	3.56±2.03	49.69±2.19
	537	87.8±0.19	**87.6 ± 0.30**	0.18±0.16	87.85±0.22	**0.032±0.07**	41.0±1.91	16.7 ± 7.50	24.3 ±5.81	**38.21±2.70**	2.84±0.88
	715	85.8±0.21	85.8 ± 0.22	**0.07±0.04**	85.78±0.29	0.096±0.14	38.5±1.69	**21.8 ± 6.57**	**16.7 ±7.21**	36.45±1.55	**2.13±0.76**
Recovered accuracy:			93.29%	–	97.36%	–	Recovered accuracy:	34.77%	–	60.94%	–
ResNet50	107 *	**95.6± 0.076**	77.5 ± 8.89	18.1 ± 8.91	**93.7 ± 0.17**	1.74 ±1.60	Training limited by memory constraints				
	110	94.8±0.29	56.0 ±20.7	38.7 ±20.5	90.86±0.70	3.830±0.80					
	213	94.0±0.16	**91.7 ± 0.98**	2.25±0.90	93.55±0.21	0.477±0.20	**61.8±0.40**	5.91± 0.89	55.9 ±0.99	44.56±1.21	17.27±1.35
	318	92.2±0.18	91.4 ± 0.45	0.85±0.49	92.03±0.20	0.190±0.05	59.1±0.36	8.56± 2.66	50.5 ±2.55	44.28±0.65	14.82±0.63
	423	90.4±0.30	90.2 ± 0.37	**0.18±0.09**	90.23±0.15	**-0.007±0.09**	56.5±0.27	21.4 ± 2.87	**35.0 ±2.99**	**46.16±0.72**	**10.37±0.58**
	1415	72.8±0.78	72.8 ± 0.78	0.0 ±0.0	–	–	32.0±0.19	**32.0 ± 0.19**	0.0 ±0.0	–	–
	1920	53.3±1.29	53.3 ± 1.29	0.0 ±0.0	–	–	29.2±0.62	29.2 ± 0.62	0.0 ±0.0	–	–
	3100	50.1±0.25	50.1 ± 0.25	0.0 ±0.0	–	–	22.8±0.37	22.8 ± 0.37	0.0 ±0.0	–	–
Recovered accuracy:			96.73%	–	98.68%	–	Recovered accuracy:	51.78%	–	74.69%	–

Table 4.1: **Large Receptive Field Enhances Prunability:** Network accuracy in dense, pruned, and fine-tuned networks. Pruning rate: 0.9. Default receptive fields (RF) marked by *. The larger the receptive field, the lower the dense network accuracy and the smaller the accuracy drop between dense and pruned models. For smaller receptive fields, where the accuracy drop due to pruning is largest, fine-tuning more significantly enhances performance. Due to this trade-off, the best-performing pruned networks after fine-tuning use receptive fields that are larger than the optimal receptive fields for dense networks, but smaller than those of the optimal one-shot pruned networks. Note the exception for VGG tested on CIFAR10. For ResNet50 experiments on Tiny ImageNet, memory constraints prevented testing receptive fields smaller than 213.



(a) **Small ImageNet** with no batch normalisation and pruning rate 0.8.

(b) **Small ImageNet** with batch normalisation adjustment and pruning rate 0.95.

(c) **CIFAR10** with no batch normalisation adjustment with pruning rate of 0.9.

Figure 4.3: **RF < Input Image Size Regime: (a) and (b)** Increasing receptive field enhances one-shot accuracy with full detailed images (red) with and without batchnorm adjustment. When details are striped from the image (blue), only (a) shows that increasing receptive field enhances pruned accuracy but to a much lower degree **(c)** Interpolating CIFAR10 to a 224x224 resolution (blue) is detrimental for one-shot accuracy in the absolute sense but increasing receptive field marginally improves it. Leaving CIFAR10 at native resolution (red) of 32x32 leaves ResNet25 on the RF > Input Image regime, where the results are on par with those on Table 4.1.

Receptive Field Smaller than the Input Image

All experiments in the previous section considered the regime in which the receptive field significantly exceeds the size of the input image. In such cases, numerous layers may learn representations at a fixed resolution, effectively covering the entire input image. To examine whether the observed pruning behaviour persists when the receptive field is smaller than the input, a custom architecture was designed which maintains depth while having a constrained receptive field, named ResNet25. The details of the implementation can be found in Section A.3.

To complement this architectural modification, these experiments designed a dataset with larger images and 200 classes, drawn from ImageNet (Deng *et al.*, 2009), which is refered to as Small ImageNet. This dataset retains the same

class structure as Tiny ImageNet (Le & Yang, 2015), with 500 images per class. However, in this case, the images are resized to 224 while preserving their high-resolution characteristics.

Figure 4.3a shows that even when the receptive field is smaller than the input, the beneficial effect of a larger receptive field on pruned accuracy remains. In both cases, full detailed (red markers) and non-detailed (blue markers) images there is an increase in one-shot pruning performance as the receptive field increases. Figure 4.3c shows ResNet25 trained on CIFAR10 at both 32×32 and 224×224 resolutions. These results demonstrate that input resolution plays a crucial role in one-shot accuracy. Even in cases where pruning reduces one-shot accuracy more severely at 224×224 resolution, models with larger receptive fields consistently outperform those with smaller receptive fields after pruning.

Beyond the relationship between the receptive field size and input dimensions, the relationship between image detail and pruned performance is also investigated. To test this, the resolution of Small ImageNet images is artificially reduced to 32×32 using interpolation, followed by upscaling them back to 224×224. The effect of this transformation is shown in Figures 4.3a and 4.3b, where a reduction in dense accuracy is observed (blue markers).

Then, the ResNet25 experiments are repeated on Small ImageNet while applying batch normalisation adjustment, following (Li *et al.*, 2020). This adjustment ensures that one-shot pruning accuracy is more closely correlated with final fine-tuned accuracy. As shown in Figure 4.3b, for the original 224×224 resolution and pruning rate of 0.95, a larger receptive field still correlates with higher pruned accuracy. However, in the case of resized images (Figure 4.3b), no clear correlation between receptive field size and one-shot accuracy is observed. This could be attributed to the high pruning rate and the reduced image detail.

4.3.2 Increasing Receptive Field Enhances Redundancy and Prunability

Having established that increasing the receptive field improves pruned accuracy, this section investigates the underlying reasons for this effect. The hypothesis is that as the receptive field increases, redundancy in deeper layers also increases,

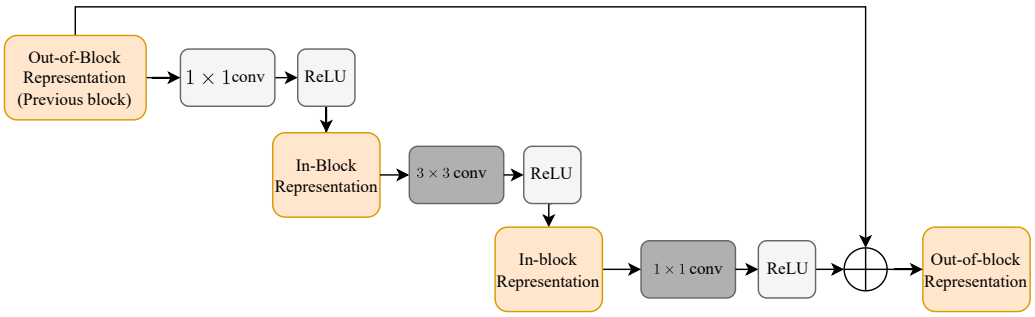
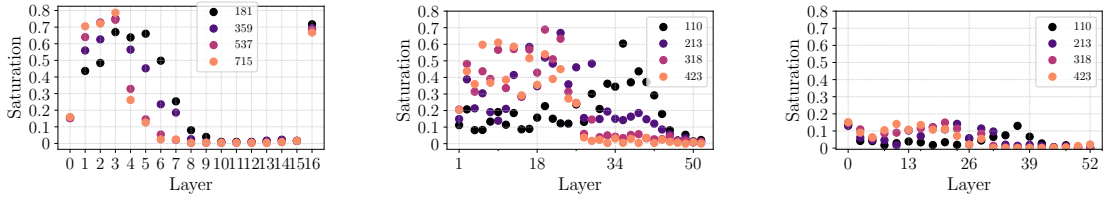


Figure 4.4: **Representations Schematic of "Bottleneck" block of ResNet50:** The ResNet50 architecture is composed of units called "Bottleneck-blocks" that stacked together compose the whole model. These blocks contain 3 convolutional layers (two 1 ×1 and one 3 × 3) and one residual connection between its input and output. Within this block, the feature maps (representations) of these three representations are named "In-block" and "Out-of-block" depending if the representations belong to a residual connection or not. Similarly, the layers that produce such representations are named the same.

allowing for more aggressive pruning while preserving accuracy. To quantify this effect, the utility of the layer was evaluated using two methods: measuring layer "saturation", as defined in Equation (2.2), and evaluating the test accuracy of logistic probes trained with intermediate representations, similar to (Alain & Bengio, 2018). The ResNet50/25 layers are divided into two groups: those whose representations lie inside the residual blocks (in-block) and those outside residual blocks (out-of-block); see Figure 4.4 for a schematic of the categorisation.

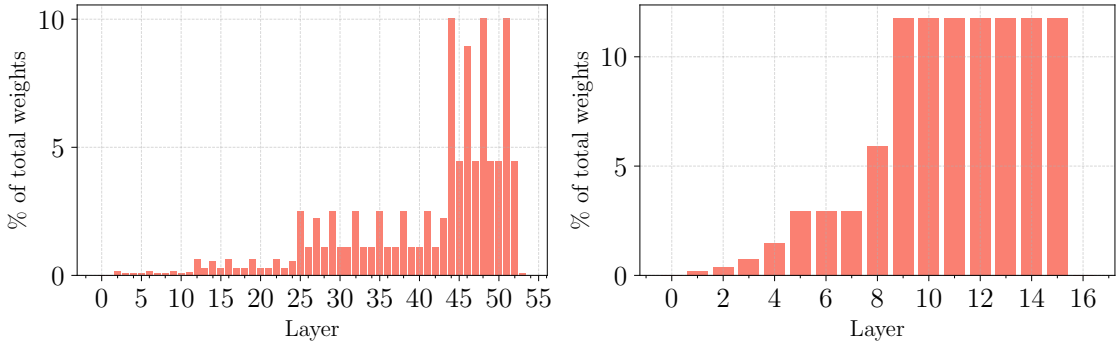
Figure 4.5 shows that for both models studied, increasing the receptive field shifts the saturation peak toward shallower layers (see layer 2 in Figure 4.5b and layer 18 in Figure 4.5b) while decreasing saturation in deeper layers, effectively increasing their redundancy. Figure 4.7 shows the probe accuracy for each layer for VGG and ResNet50 trained on CIFAR10. These plots show that as receptive fields increase, the accuracy of probes in shallow layers increases, suggesting that larger receptive fields can integrate more image context and, in turn, earlier layers can solve the classification task more effectively. However, this causes accuracy to stagnate earlier and, consequently, larger receptive fields have lower dense accuracy than smaller ones. Additionally, it is evident that, within larger receptive fields,



(a) VGG (b) ResNet50 Inside-block layers (c) ResNet50 out-of-block layers

Figure 4.5: **Saturation on CIFAR10: (a)** The saturation of deeper layers diminishes as the receptive field increases, which correlates with increased pruned performance. **(b) and (c)** From layer 30 onwards, the saturation of deep inside-block layers diminishes as the receptive field expands, whereas out-of-block layers persist in an undersaturated state irrespective of the receptive field size. This suggest that the lower saturation of deeper layers present in larger receptive fields is responsible for the increase in prunability.

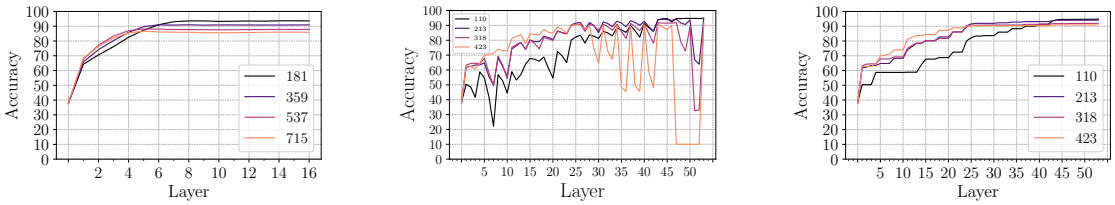
deeper layers do not contribute further to dense accuracy, as no increase is observed in these layers.



(a) ResNet50 (b) VGG

Figure 4.6: **Deeper layers contain majority of the weights: (a)** In ResNet50, more than 80% of weights are contained between the layers 25 and 54 **(b)** Similarly for VGG, more than 70% of weights are present between layers 8 and 15

To further explore the effect of the size of the receptive field on the redundancy of the layers, cumulative pruning experiments conducted, progressively pruning the layers from the deepest up to the layer i^{th}. Figure 4.8 illustrates this process:



(a) VGG (b) ResNet50 Original (c) ResNet50 Smoothed

Figure 4.7: **Probe Accuracy on CIFAR10:** Here the feature maps of each convolutional layer were downsampled to a 7x7 feature map using adaptive average pooling. These feature maps are produced by a trained model, using the CIFAR-10 training set. The entire training set is then replicated L times (the number of layers in the given model). For each dataset, a logistic regression model is trained to solve the original classification problem, and the reported accuracy is the test accuracy. Logistic probes trained on representations from shallow layers of dense models with a large receptive field achieve higher accuracy than their counterparts trained on representations from models with a smaller receptive field. Early layers can effectively handle classification, but as the network deepens, probe accuracy stagnates or decreases, especially at higher receptive fields. For the smoothed version, Figure 4.7c, the original probe accuracy per layer is processed by retaining only the best accuracy achieved up to the i^{th} layer. This indicates that logistic probes in earlier layers within larger receptive fields increase accuracy more than those in smaller receptive fields. Still, this effect persists in deeper layers, reducing final accuracy.

for example, the accuracy bar in layer 6 represents the accuracy of the model after collecting all the weights from layers 6-15 and pruning this group of weights (all weights from layer 6-15) by 90%. Note that the accuracy of the last layer (15 in this example) corresponds to the network's accuracy when the last layer alone is pruned by 90%. Most of the weights for both models are concentrated in the deeper layers, as shown in Figure 4.6, indicating that pruning these layers results in a significant reduction in the network's overall weight count.

The findings indicate that an increase in receptive field size correlates with a reduction in saturation in deeper layers, and that, consequently, pruning layers with low saturation has a minimal effect on overall accuracy.

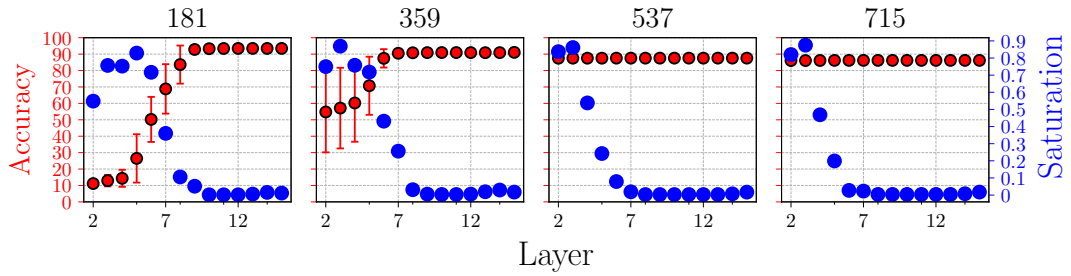


Figure 4.8: **Saturation and One-Shot Accuracy for VGG on CIFAR10:** Increasing the receptive field mitigates saturation within deep layers while preserving high accuracy in one-shot pruning. As demonstrated in Figure 4.6b, a significant proportion of the weights resides in the deeper layers (starting from layer 7), thus, pruning 90% of merely a few of these layers impacts a substantial quantity of the models weights.

Next, the in-block and out-of-block representations are investigated separately. As shown in Figures 4.9 and 4.10, out-of-block layers generally exhibit low saturation, with average values ranging from 0.1 to 0.3, while in-block layers have substantially higher saturation values, ranging from 0.3 to 0.9. The results indicate that pruning out-of-block layers has little impact on accuracy, whereas pruning highly saturated in-block layers leads to noticeable accuracy degradation. Furthermore, as indicated in Figure 4.9, the incorporation of shallow layers (beginning from $18th$) into the pruning process leads to a substantial deterioration of the accuracy of the one-shot (represented by red dots) for small receptive fields. In contrast, larger receptive fields (Figures 4.9 and 4.10) experience minimal one-shot accuracy loss, even when shallow layers are included in the pruning process.

Finally, this analysis is extended to scenarios where the receptive field is smaller than the input image, using ResNet25 on Small ImageNet. Figures 4.11a to 4.11c show both, the saturation and the accuracy of the probes in representations of the intermediate layer. A similar pattern emerges: as the receptive field increases, saturation in deeper layers (from layer 22 onwards) decreases, reinforcing their redundancy. Meanwhile, in Figure 4.11c, larger receptive fields lead to an accelerated increase in accuracy for shallow layers, but plateau at a lower level, again

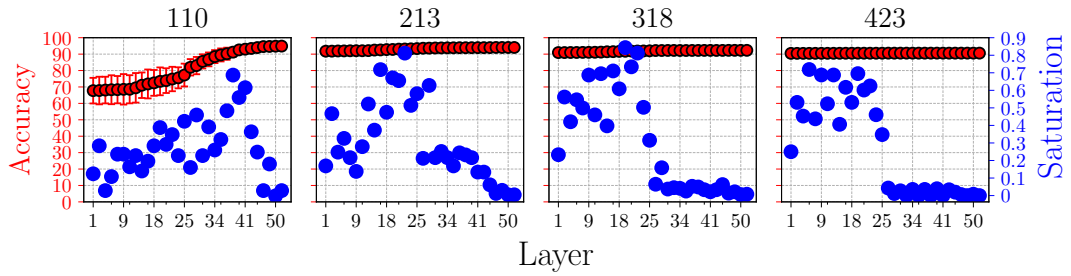


Figure 4.9: **Saturation and One-Shot Accuracy for ResNet50 In-block layers CIFAR10:** As the receptive field (RF) increases, layer saturation (blue points) decreases, while one-shot pruning accuracy (red points) improves, highlighting the enhanced prunability of models with larger receptive fields.

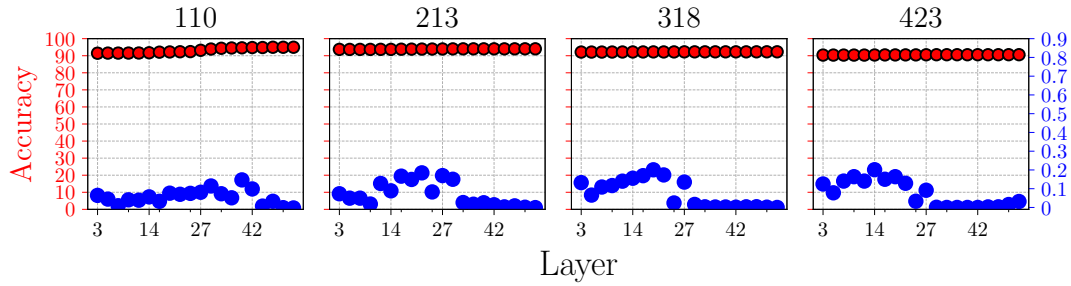
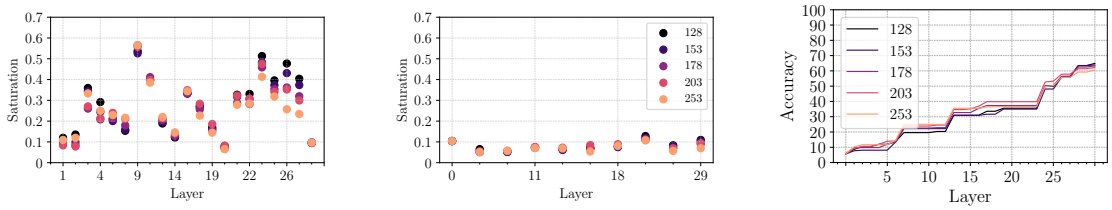


Figure 4.10: **Saturation and One-Shot Accuracy for ResNet50 Out-of-block layers on CIFAR10:** Here, when pruning undersaturated outside-block layers the model maintain almost equal accuracy.

indicating that deeper layers do not contribute meaningfully to the classification task.

4.3.3 Similarity of Representations and Receptive Field

The previous section demonstrated that larger receptive fields lead to decreased saturation in deeper layers, indicating increased redundancy. To further validate this finding, this sections examines whether this redundancy manifests itself in the similarity structure of network representations. By analysing representational similarity patterns across layers and their correspondence with layer-wise pruning



(a) ResNet25 inside-block representations

(b) ResNet25 out-of-block representations

(c) Probe Accuracy Dense Model (Smooth)

Figure 4.11: **Saturation and Probe Accuracy for RF < Input Image: (a) and (b)** Inside-block layers (from 22^{th} onwards) have decreased saturation with larger receptive fields for RF < input images, while out-of-block layers have consistently low saturation irrespective of the size of the receptive fields, explaining the increased prunability (Figures 4.3a and 4.3c). **(c)** Consistent with the findings in Figure 4.7, a larger receptive field results in logistic probes within shallow representations that achieve superior accuracy compared to their counterparts with a smaller receptive field, although the difference is less noticeable than the case of RF > input image.

rates, this section provides additional evidence that larger receptive fields induce a specific form of redundancy concentrated in deeper layers, particularly in architectures with residual connections. Thus, this section poses two hypotheses:

1. Larger receptive fields induce representational similarity between deeper and shallower layers, particularly in ResNet architectures, as shallow layers capture sufficient information to perform the task, making the development of distinct complex features in deeper layers unnecessary.

2. This representational similarity between layers creates redundancy concentrated in deeper layers, enabling aggressive pruning of these layers without substantial accuracy loss, as evidenced by layer-wise pruning rate patterns that increase with depth.

To evaluate the initial hypothesis, the similarity of features within the network is calculated. The hypothesis postulates that since a network with a large receptive field predominantly acquires “simple” features within its shallow layers (Ro &

Choi, 2021; Yosinski *et al.*, 2014), the majority of representations throughout the network would resemble those found in these layers.

This section uses the methodology described in Kornblith *et al.* (2019), which involves using two distinct trained instances of the identical model. From these, the output generated by the ReLU activation function across all layers are recorded for a selected number of samples. Next, the Linear Centred Kernel Alignment measure is calculated, as introduced by Kornblith *et al.* (2019), to evaluate the similarity between the layer representations derived from one seed and the corresponding layer representations of the alternative seed.

For each receptive field, two seeds of the ResNet50 model were used, and 1,000 images from the CIFAR10 test set to obtain the representations (feature maps) of each layer. The analysis of the VGG architecture, presented in Figure A.1, reveals that it does not exhibit a similarity pattern comparable to that observed in ResNet50, potentially due to the absence of residual connections.

Figures 4.12a to 4.12c shows that the in-block and out-of-block representations behave differently, and the following remarks can be stated.

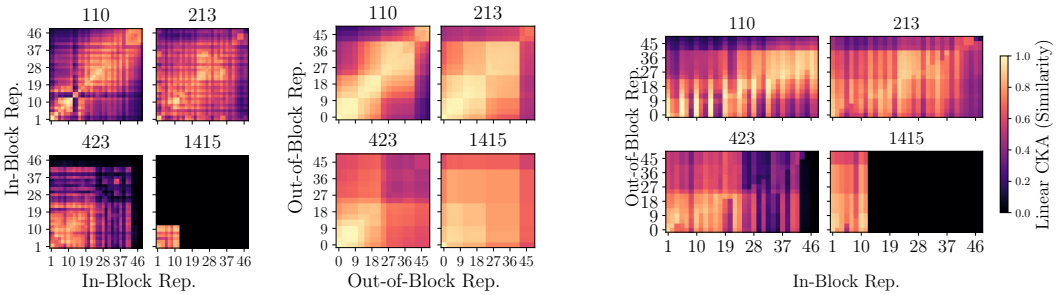
Remark 4.3.1. As the receptive field increases, the similarity between in-block representations is reduced (Figure 4.12a)

Remark 4.3.2. In contrast, the representations outside of blocks consistently exhibit a high degree of similarity. Additionally, there is a discernible increase in similarity as the receptive field increases (Figure 4.12b)

Remark 4.3.3. As the receptive field increases, the similarity between deep in-block and out-of-block representations decreases (Figure 4.12c).

Notably, only shallow in-block representations exhibit similarity to all or most out-of-block representations. For example, the coloured stripes in Figure 4.12c do not extend beyond layer 16 for a receptive field of 1415. With an increase in the receptive field, the resemblance between deep in-block representations and out-of-block representations diminishes, while notably, only the shallow in-block representations demonstrate similarity to the majority of out-of-block representations.

Remark 4.3.2 can be ascribed to the inclusion of residual connections in the architecture. These connections are implemented either as identity mappings or



(a) In-block (b) Out-of-block (c) In & Out block

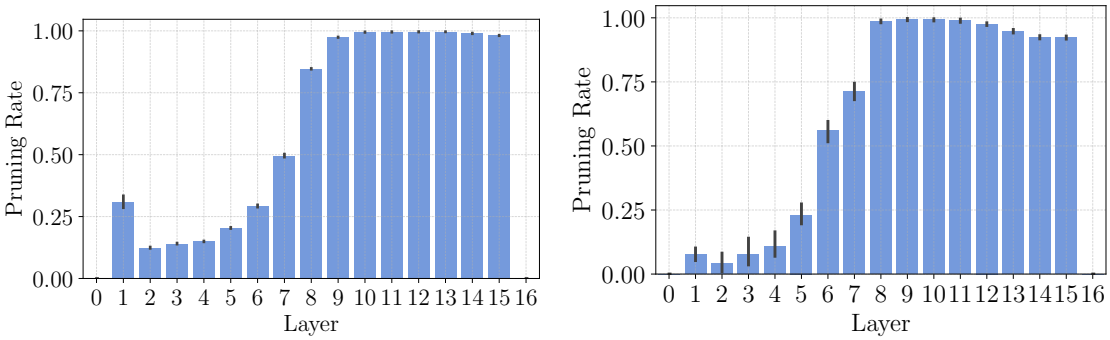
Figure 4.12: **Similarity between pattern between representations for ResNet50 trained on CIFAR10.** Fig. 4.12a shows that as the receptive field is expanded, a discernible reduction in the similarity among in-block representations becomes evident. Moreover, the deeper layers are the first to exhibit the dissimilarity pattern with other deeper in-block representations. Fig. 4.12b shows that for out-of-block representations, not only does the similarity remain high, but also as the receptive field increases, their similarity increases. Fig. 4.12c shows that as the receptive field increases, the deeper in-block representations become less similar to any out-of-block representations. Only the shallow in-block representations are similar to all or most outside block representations. Figure A.1 shows the similarity pattern of VGG; contrary to ResNet50, VGG does not show a pattern of increasing dissimilarity as the receptive field increases.

through the incorporation of a 1x1 convolutional layer that exclusively increases the channel count of the input. In the case of 1x1 convolutions, the 1x1 convolutions produce a scalar transformation of the inputs without the integration of spatial information from prior feature maps. As a result, the majority of out-of-block representations exhibit a considerable degree of similarity irrespective of the receptive field. Given that the output of the initial convolutional layer is the first to be replicated by the residual connections, subsequent representations inherently incorporate this fundamental information. This factor explains why out-of-block representations exhibit similarity regardless of variations in receptive fields.

One observation is that deeper in-block layer representations grow progressively dissimilar from any other in-block representation, as emphasised in Remark 4.3.1. In contrast, shallow in-block representations exhibit similarity both among them-

selves and with the majority of out-of-block representations, as indicated in Remark 4.3.3. This pattern extends to out-of-block representations, which retain similarity to the initial layers throughout the network.

These collective observations yield a significant inference: the learning process predominantly uses shallow representations. Deeper in-block representations diverge from similar entities (evident in the black strips in deeper layers of Figure 4.12a), while only shallow in-block representations maintain similarity either among themselves or with out-of-block representations.

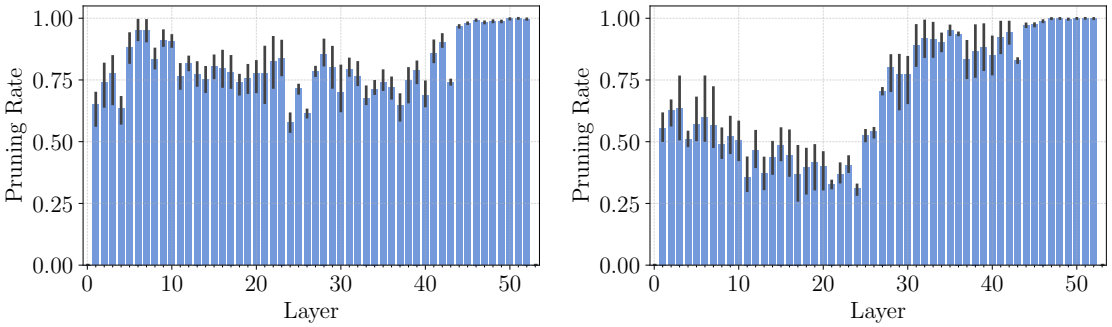


(a) 181 (b) 537*

Figure 4.13: **Pruning rates per layer of VGG on CIFAR10:** Pruning Rate of 0.9, best A_{OS} denoted by *. Here, increased receptive field results in increased pruning rates for deeper layers but at the same time this results in increased prunability (low Δ_{OS})

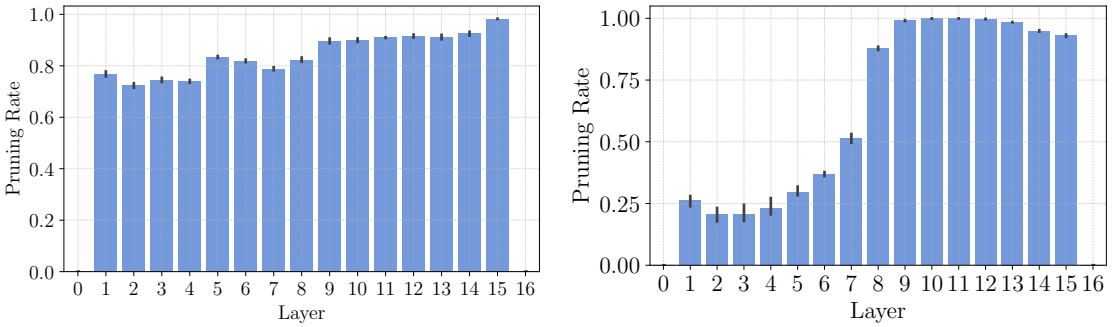
In support of the second hypothesis, the results demonstrate that as the receptive field increases, the pruning rates of individual layers increase more significantly in deeper layers relative to shallower ones. Figures 4.13 and 4.14 present the pruning rates per layer for both ResNet50 and VGG, applying a pruning rate of 0.9 on the CIFAR10 dataset across 5 different seeds.

Figure 4.14 clearly illustrates the effect of a large receptive field for ResNet50. The smallest receptive field demonstrates a uniform pruning rate pattern; however, with a large receptive field, deeper layers are pruned more intensely. Tiny ImageNet follows a similar trend: Figures 4.15 and 4.16 reveal that increasing the receptive field shifts the pruning distribution from uniform to concentrated, with higher pruning rates in deeper layers.



(a) 110 (b) 213*

Figure 4.14: **Pruning rates per layer of ResNet50 on CIFAR10:** Pruning rate of 0.9, best A_{OS} denoted by *. Before layer 24 the pruning rates per layer decrease while the pruning rates after said layer increase or stay the same.

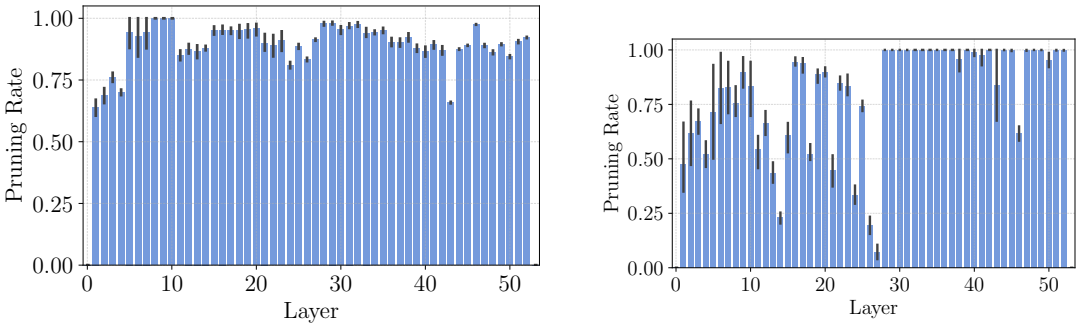


(a) 181 (b) 715*

Figure 4.15: **Pruning rates per layer of VGG in Tiny ImageNet:** Pruning rate of 0.9, best A_{OS} denoted by *. Similar to Figure 4.13, pruning rates increase in deeper layers, although for Tiny ImageNet the pruning rate imbalance between smaller and larger RF is more pronounced.

Even in Figure 4.13, where the VGG pruning rate pattern is already unbalanced, an increase in the receptive field generally results in reduced pruning rates for shallow layers and increased pruning rates for deeper layers. This, combined with the observation that the largest receptive fields do not experience considerable accuracy loss when pruned, implies that a larger receptive field renders deeper representations redundant (Richter *et al.*, 2021b).

This behaviour is consistent with the observations of decreased saturation in



(a) 213 (b) 1415*

Figure 4.16: **Pruning rates per layer of ResNet50 in Tiny ImageNet for different receptive fields**: Pruning rate of 0.9, best A_{OS} denoted by *. Similar to Figure 4.14 pruning rates increase in deeper layers, although for Tiny ImageNet the pruning rate imbalance between smaller and larger RF is more pronounced.

deeper layers and stagnating accuracy of layer probes (Figures 4.5 and 4.7), as these two measures capture the contribution of these layers to task performance. The fact that the pruning rates of deep layers increase naturally as the receptive field increases indicates that the usefulness of deep layers can be inferred from their relative magnitude compared to shallow layers.

4.3.4 Receptive Field and Stochastic Pruning

Having established that larger receptive fields induce representational redundancy concentrated in deeper layers, with corresponding increases in layer-wise pruning rates,this section investigate whether this phenomenon can be further exploited through Stochastic Pruning methods shown in the previous chapter.

For each of the receptive fields explored, a grid search with $\gamma = \{0.8, 0.85, 0.9, 0.95\}$ and $\sigma = \{0.001, 0.002, 0.003\}$ wes performed. Table 4.2 shows the best result for a particular receptive field.

Stochastic pruning can outperform deterministic pruning with very little margin. One possible explanation is that the contribution of the redundancy introduced by the receptive field is larger than the stochastic pruning one, diminishing the accuracy gains.

	Tiny ImageNet						CIFAR 10					Small ImageNet					
	RF	γ	σ	DP	SP	Δ_{OS}	γ	σ	DP	SP	Δ_{OS}	RF	γ	σ	DP	SP	Δ_{OS}
ResNet50	110	-	-	-	-	-	0.9	0.001	62.50	63.70	**1.20**						
	213	0.8	0.003	36.22	38.61	**2.39**	0.95	0.003	44.00	45.79	**1.79**	128	0.85	0.003	12.43	12.99	**0.56**
	318	0.8	0.003	31.04	31.63	**0.59**	0.95	0.003	60.13	66.71	**6.57**	153	0.85	0.003	13.56	15.24	**1.68**
	423	0.8	0.003	39.92	40.24	**0.31**	0.95	0.001	32.99	36.96	**3.97**	178	0.85	0.001	18.35	18.75	**0.40**
VGG19	181	0.85	0.001	5.72	6.02	**0.30**	-	-	-	-	-	203	0.85	0.003	18.79	20.49	**1.70**
	359	0.8	0.003	10.10	14.92	**4.82**	-	-	-	-	-	253	0.95	0.001	1.99	2.12	**0.13**
	537	0.9	0.001	18.71	19.45	**0.73**	0.95	0.001	82.96	83.25	**0.29**						
	715	0.9	0.001	20.06	20.61	**0.55**	-	-	-	-	-						

Table 4.2: **Stochastic Pruning and Receptive Field Modulation on Tiny ImageNet, CIFAR10 and Small ImageNet.** For VGG19 on CIFAR10, we could only find parameters that allowed SP to surpass DP for one RF, which suggests that the increase in DP caused by the increase in the receptive field.

For ResNet50 in CIFAR10, the receptive fields 213 and 318 have the same optimal parameters as Table 3.1 but with better DP performance, explaining their smaller accuracy gap between DP and SP. This may be due to the increased accuracy from using a larger receptive field.

For Tiny ImageNet, it was possible to find parameter settings where SP surpasses DP, but most gains are below 1%. Despite DP increasing with the RF, this does not appear to have a consistent effect on SP accuracy for VGG19, whereas for ResNet50, there is a trend of diminishing returns, possibly due to the large receptive field. For ResNet25 on Small ImageNet, there are modest gains in accuracy $< 2\%$ for SP.

Overall, SP effectiveness decreases as receptive field size increases, even when optimised parameters for that receptive field are used. It is possible that the feature variance explosion is also reduced as the receptive field increases, which also explains the reduced accuracy gains from SP.

4.4 Discussion and Conclusion

This chapter presented evidence that increasing the receptive field improves a model's prunability. The results showed that undersaturation in the deep layers is the reason for this enhanced prunability. As shown in Figure 4.7, deeper layers do not contribute to accuracy beyond what is achieved by shallow layers, suggesting that no new information is being integrated in those layers. For cases where the receptive field is smaller than the input image, deeper layers contribute to the classification accuracy, as seen in Figure 4.11c, but the contribution decreases as the receptive field increases.

A hypothesis explored in this chapter was that larger receptive fields may limit the model's capacity to develop complex or specific representations, which are typically constructed within the deeper layers (Yosinski *et al.*, 2014). Consequently, deeper layers in models with large receptive fields tend to generate rudimentary or generalised features, similar to those produced by shallow layers.

An important insight from these findings is that pruning improves performance not only because larger receptive fields introduce redundancy but also because they concentrate useful information in shallow layers. In networks with small receptive fields, this "usefulness" is distributed throughout the network, so pruning leads to a greater loss of precision as useful layers are affected. In contrast, large receptive fields concentrate useful information in shallower layers, which are less affected by pruning. As a result, dense accuracy is lower due to redundancy in deeper layers, but pruning selectively removes the less useful deeper layers while preserving performance-critical shallow layers, thereby improving pruned accuracy.

Furthermore, the preliminary results on similarity of representations suggest that the representations of shallow layers may function similarly to those of deeper layers in models with large receptive fields; this can be interpreted as the model only learning "shallow" features that result in less dense accuracy but increased robustness to pruning, since features in shallow layers are considered more general descriptors of the images.

A fundamental question that remains unresolved is the nature of the relationship between representation similarity and saturation. Suppose two layers demonstrate a significant degree of similarity, and one layer presents a high level

of saturation. Does this necessarily imply that the other layer will also display a similarly high level of saturation? What factors differentiate them, and what precisely leads one layer to be more saturated while preserving similarity with another, potentially less saturated layer? Furthermore, how does this phenomenon relate to the pruning process? These questions are left for future investigation.

Given that a max-pooling layer is used after the first convolutional layer, as the receptive field of the models changes, the size of internal representations also changes, which can significantly affect model behaviour. Given that the size of the feature maps is reduced, the spatial features present in the images are lost, possibly explaining the reduction in precision as the receptive field increases. Additional experiments in which the feature map size remained constant across different receptive fields were conducted in Section A.5. These experiments modified the dilation of the first convolutional layer of the ResNet50 and VGG models. Additionally, padding was used to maintain the same feature map size as the receptive field increased. The results in Table A.8 show that, if the receptive field increases with dilation and the feature map size is constant, increasing the receptive field does not increase prunability. This suggests that the information compression performed by the max-pooling layer is responsible for the increased model prunability observed in this chapter. Further experiments should be conducted to measure layer saturation of the models modified with dilation, to see if these models exhibit high saturation and thus, explain the constant prunability across receptive fields.

It would be valuable to explore the interplay between receptive field size and other factors known to affect sparsity, such as L1 regularisation, dropout, and dynamic sparse training (Evci *et al.*, 2020a; Mocanu *et al.*, 2018). Understanding how these techniques interact with receptive field size and redundancy could provide deeper insights into the mechanisms underlying prunability and lead to more effective sparsity-inducing strategies.

Chapter 5

Training Dynamics, Receptive Field, and Pruning

5.1 Introduction

Pruning neural networks is one of the many methodologies for making models efficient and deployable, yet the factors that determine how prunable a trained model is are poorly understood. This chapter explores how both the receptive field of a model and the choice of optimiser influence the geometry of the loss landscape and, ultimately, the model's prunability.

While prior work has extensively focused on pruning techniques and their impact on performance, then the analysis shifts to how design and training choices, particularly those made before pruning, shape the prunability of a model after training. This chapter begins by examining how the receptive field affects the model's optimisation loss landscape. The results show that increasing the receptive field leads to a more rugged loss surface, reflected in a wider Hessian spectrum. This complicates training by introducing more pronounced curvature, making it harder for gradient-based optimisers to converge efficiently.

Interestingly, accuracy can be recovered by increasing the model's width. One interpretation is that widening the model simplifies the traversal of the loss landscape for SGD, thereby enabling improved optimisation. This raises the question, instead of modifying the landscape to help SGD, can more sophisticated optimis-

ers (that use information of the curvature of the landscape) be used to address the challenges posed by this rugged landscape? Next, curvature-aware optimisers such as Adaptive Sharpness-Aware Minimisation (ASAM) (Kwon *et al.*, 2021) and Eigenvalue-corrected Kronecker Factorisation (EKFAC) (George *et al.*, 2018) are considered to answer this question. These optimisers incorporate curvature information and, in theory, should be better equipped to navigate highly curved terrains. However, contrary to expectations, the experiments show that these optimisers do not confer an advantage in final accuracy, particularly in models with large receptive fields. This suggests that curvature-aware methods may not effectively circumvent the optimisation barriers introduced by the receptive field.

One notable observation is that EKFAC is the worst-performing of all the optimisers. One hypothesis is that its poor performance relative to ASAM and SGD stems from EKFAC finding sharper solutions. Results show that indeed, EKFAC actually finds significantly sharper solutions than ASAM and SGD. Furthermore, models trained with EKFAC exhibit a more diverse set of filters than those trained with ASAM or SGD.

This led to the hypothesis that the EKFAC would also perform badly in the pruned setting, for two reasons. First, sharp solutions are vulnerable to weight-space perturbations, such as pruning. Second, a more diverse set of filters reduces network redundancy, leaving the model more vulnerable to pruning.

Experiments show that models trained with EKFAC perform worse in the pruning setting than those trained with ASAM and SGD, confirming the initial hypothesis. At the same time, enlarging the receptive field improves prunability across optimisers, while sharpness and filter diversity remain relatively stable across receptive fields for EKFAC models. This suggests that sharpness or filter diversity alone cannot fully explain pruning behaviour.

Finally, experiments show that similar to results in Chapter 4, this increased prunability arise primarily from changes in saturation, rather than from alterations to sharpness or filter diversity. As such, the findings suggest that while receptive field expansion can improve pruning outcomes, it does so through mechanisms distinct from those typically considered.

Overall, this chapter highlights a nuanced interplay between model architecture and optimisation strategy for model compression. The results challenge assump-

tions about the benefits of curvature-aware optimisation and raise new considerations for practitioners are that training choices can fundamentally determine a model's compressibility, thus requiring the consideration of the training pipeline, and in particular the type of optimiser, for compression of models.

5.2 Large Receptive Field Correlates with Ruggedness of Loss Landscape

Previous research established that width and depth often do not significantly improve accuracy when the receptive field is fixed (Koutini *et al.*, 2019; Wang *et al.*, 2020). This implies that neither increasing the width nor the depth can compensate for the information lost due to a suboptimal receptive field. However, this insight does not explain why the accuracy of the unpruned model decreases as the receptive field expands, and how it affects the training capacity of the network. This section posits that larger receptive fields fundamentally increase the difficulty of training the network itself. This section substantiates this claim by showing how larger receptive fields influence the ruggedness of the loss landscape. Specifically, Figure 5.1 show the 90 largest Hessian eigenvalues for the VGG19 and ResNet50 models on CIFAR10, before training.

Remarkably, at the beginning of training, models with larger receptive fields exhibit a wider Hessian spectrum, encompassing larger eigenvalues. This observation implies that the landscape of these models has steeper descent and ascent directions, making them more challenging to traverse, particularly for first-order optimisers (Bottou, 2010; Bottou & Bousquet, 2007; Sagun *et al.*, 2018). One possible explanation is that large receptive fields, which have smaller feature maps, require larger changes in the magnitude of individual weights to noticeably alter the output of the model, despite having the same number of weights.

Although previous studies have shown that deeper networks can increase the occurrence (Choromanska *et al.*, 2015) and quality of local minima (Kawaguchi, 2016; Kawaguchi & Bengio, 2019; Kawaguchi *et al.*, 2019), some research shows that it is the receptive field that predominantly determines the maximum accuracy achievable by deep neural networks, with depth contributing improvements up to

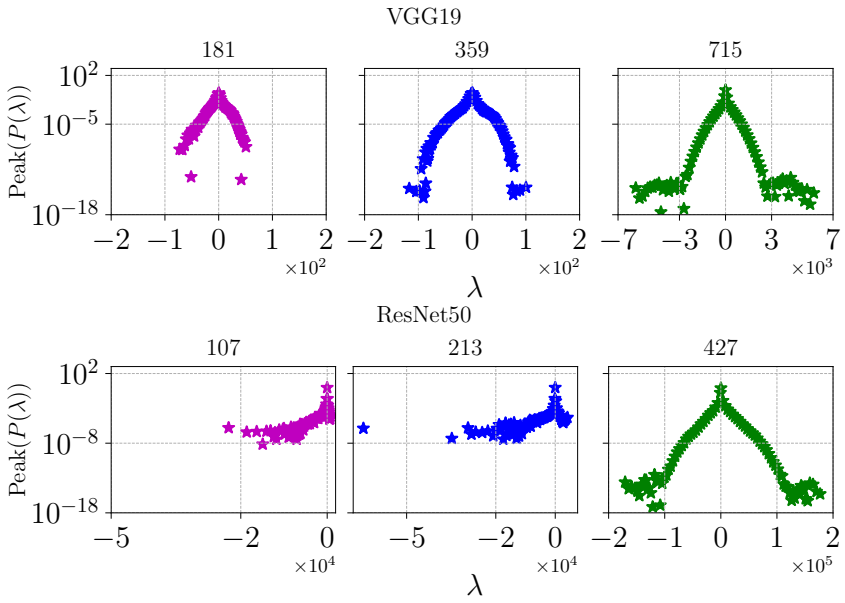


Figure 5.1: **Largest 90 Eigenvalues of the Hessian of VGG19 ResNet50 Models on CIFAR10 Before Training**: For both models, increasing the receptive field widens the hessian spectrum. Only the peaks of the estimated probability distribution of the eigenvalues are presented.

a certain threshold (Wang *et al.*, 2020). Examining the loss landscape, there are two competing forces at play: the increase of high-quality local minima (increase of depth) and the increase of ruggedness in the loss landscape. Modifying the receptive field could potentially make the optimisation process more difficult and impede the convergence to local minima produced by the increased depth.

To test this, the optimisation was run for double the number of epochs for the largest receptive field. For CIFAR10 (Figure 5.2), even with double training time, the model is unable to learn good representations that can match smaller receptive fields. Furthermore, as in Koutini *et al.* (2019), the models with the largest receptive fields in this study appear to be unable to extract relevant characteristics for generalisation, as evident in Figure 5.2. In this case, explicit overfitting does not occur, but the generalisation gap widens. This suggests that while the expressive capacity of the network is not completely impaired, the model struggles to generalise to test data. Furthermore, this section investigates the possibility of modifying the loss landscape associated with a large receptive field so that it

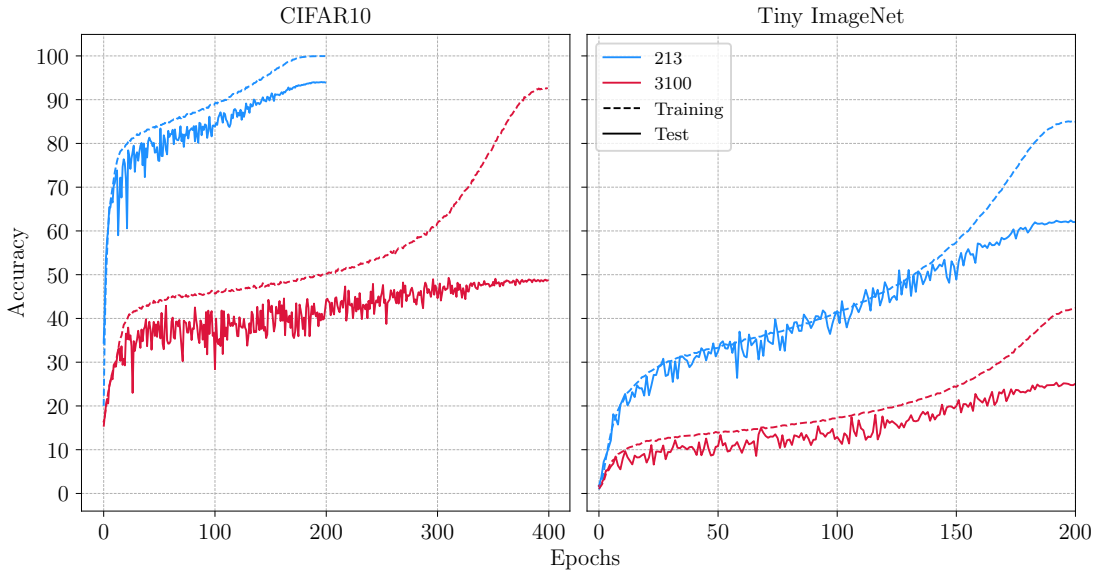


Figure 5.2: **Performance for Different Receptive Fields on CIFAR10 and Tiny ImageNet.** In the case of Tiny ImageNet, the model with a larger receptive field learns more slowly, suggesting that the loss landscape is more difficult to traverse for SGD than the loss landscape of a smaller receptive field. In both cases, CIFAR10 and Tiny ImageNet, a larger receptive field hinders the trainability of the network. Even when trained for double the number of epochs, the network with large receptive field was not able to surpass the network with smaller receptive field on test set.

allows SGD to reach better performance in large receptive fields.

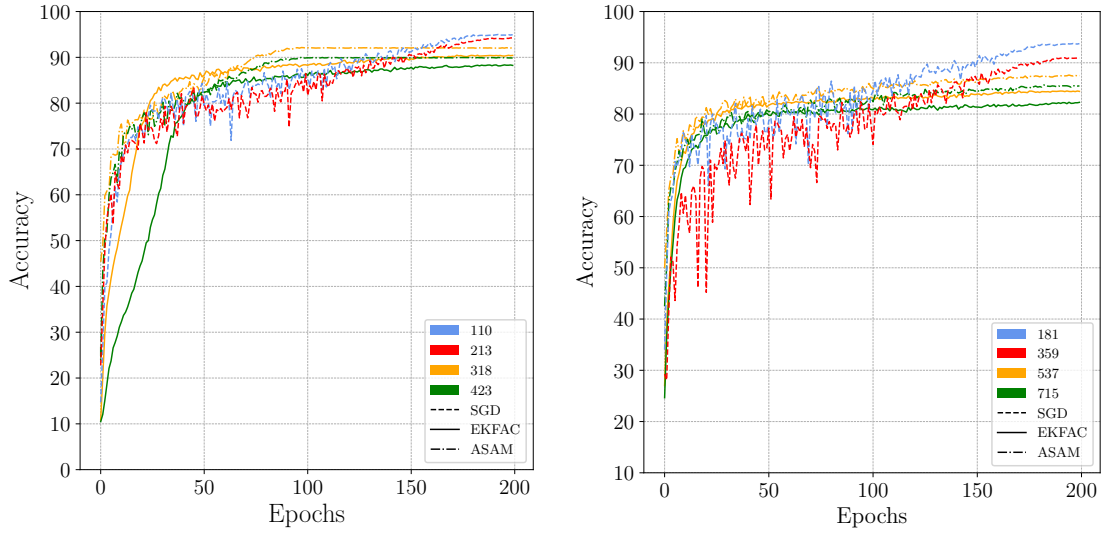
An effective method to evaluate the extent to which the loss landscape constrains the accuracy of specified receptive fields is to increase the width of the model. This approach allows for an increase in the dimensionality of the loss landscape while maintaining the existing receptive field size. The experiments were conducted by training ResNet50 with double and triple its original width in the Tiny ImageNet dataset to examine this hypothesis (Table 5.1). The findings show that augmenting the model's width (i.e., the number of parameters) partially enables SGD to navigate the limitations imposed by the receptive field's impact on the loss landscape. However, for larger receptive fields (1415, 3100), this increase in width does not completely recover the accuracy levels comparable

to those achieved with smaller receptive fields (318, 213).

Receptive Field	Width		
	×1	×2	×3
213	**61.80±0.40**	-	-
318	59.10±0.36	60.97	**61.13**
1415	32.00±0.19	37.36	38.0
3100	22.84±0.37	26.33	26.04

Table 5.1: **Width Experiments:** Accuracy of ResNet50 model on Tiny ImageNet for different widths. For the case of a receptive field of 318, increasing the width 3 fold yields similar performance than a receptive field of 213. For very large receptive field width recovers marginal information.

Increasing the model's width results in an increased number of parameters without modifying the receptive field, as opposed to altering the model's depth. In this case, the loss landscape is modified for a given receptive field, and since manipulating the loss landscape allows a larger receptive field to match the performance of a smaller, yet close, receptive field, this suggests that SGD takes advantage of the modified landscape. Another interpretation is that increasing the width increases the representational capacity by means of redundancy (filter overlap) and recombination capacity of filters. This seems to help recover high-frequency or localised structure even when the receptive field is large. This is supported by the observation that increasing the width seems to recover more accuracy for larger receptive fields. For receptive fields 1415 and 3100, the accuracy gains are $5\% - 6\%$ and $3.2\% - 3.49\%$, respectively, while for receptive field 318 the accuracy gain is between 1.87% and 2.03%. Since a smaller receptive field captures high-frequency details better than a larger receptive field, adding width, and therefore enriched capacity for recombination of filters, does not translate to the same accuracy gains.



(a) ResNet50 (b) VGG19

Figure 5.3: **CIFAR10 Training Traces with Different Receptive Fields:** Despite that curvature-aware optimisers have a smoother climb at the beginning of training, both ASAM and SGD are unable to match the performance of SGD trained on a smaller receptive field.

VGG19				ResNet50			
Level	SGD	ASAM	EKFAC	Level	SGD	ASAM	EKFAC
181	**93.77**	-	-	110	**94.99**	-	-
359	**90.98**	-	-	213	**94.24**	-	-
537	**88.13**	87.67	84.5	318	**92.3**	92.14	90.47
715	**86.12**	85.68	82.3	423	**90.91**	90.04	88.35

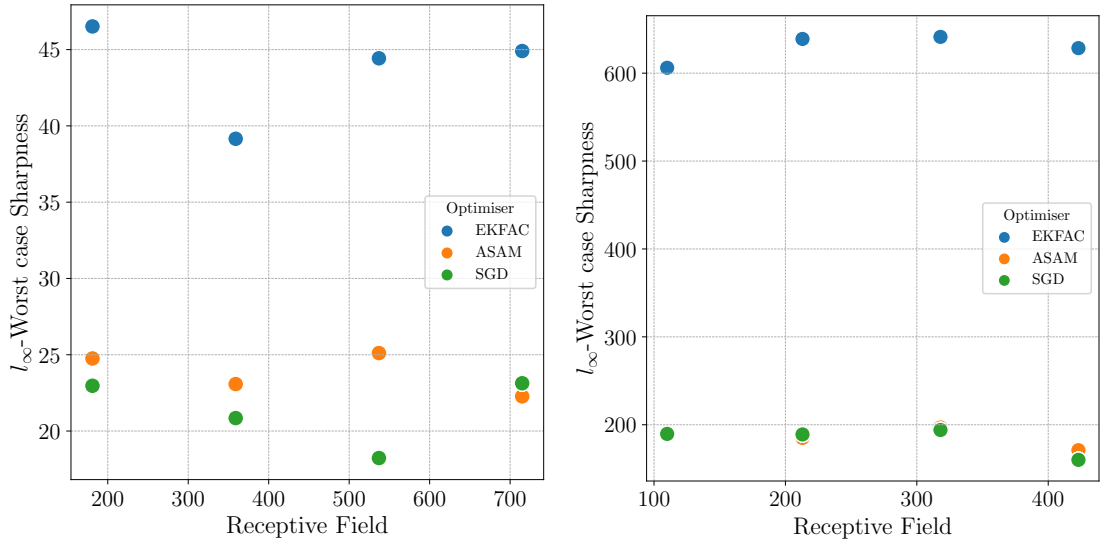
Table 5.2: **Best Accuracy for CIFAR10:** Curvature-aware optimisers that trained on larger receptive fields do not match the performance of SGD trained on smaller receptive fields. Even for the same receptive field, neither curvature-aware optimiser can match SGD's performance. After each training epoch the model's performance was measured in CIFAR10 test set. The highest test performance achieved throughout training is the one shown here.

5.3 Curvature-aware Optimisers vs Rugged Loss Landscape

The preceding section showed that the receptive field alters the loss landscape, hindering SGD training. Furthermore, results showed that when the landscape is manipulated (by increasing the width), SGD is able to partially circumvent the limitations (ruggedenes) imposed by the receptive field. Instead, this section ask: Could more sophisticated optimisers deal with the ruggedenes of loss landscape without increasing the width? Here, these more sophisticated optimisers are called curvature-aware optimisers, because they explicitly incorporate information about the local curvature of the loss landscape to guide their parameter updates. The specific question is: can curvature-aware optimisers trained with a large receptive field match or surpass SGD trained with a smaller receptive field?

To investigate this, EKFAC, a second-order optimiser, and ASAM, an adaptive sharpness minimisation optimiser, were employed. Experiments were conducted with VGG19 and ResNet50 at different receptive field levels, using SGD, EKFAC, and ASAM, respectively. The Optuna package (Akiba *et al.*, 2019) was used to optimise the hyperparameters for the learning rate and momentum. Specifically, the optimal learning rate and momentum for SGD and ASAM were determined to be 0.1 and 0.9, respectively, whereas for EKFAC, the optimal values were 0.01 for the learning rate and 0.5 for the momentum.

Although curvature-aware optimisers initially demonstrate a gradual and consistent improvement in performance (Figure 5.3), they do not surpass the performance of SGD, especially when trained with a smaller receptive field. Furthermore, they fail to outperform SGD even when training in the same receptive field (Table 5.2) despite a more consistent improvement in early epochs. In the context of EKFAC, the inclination to optimise along the most acute directions of change, which is inherent in any second-order optimisation method (Bottou *et al.*, 2018), can adversely affect its efficacy. The presence of a wider Hessian spectrum within an expansive receptive field suggests that $\frac{\lambda_{\max}}{\lambda_{\min}}$ is consequently large. This renders EKFAC susceptible to ill-conditioning, potentially explaining its inferior performance relative to ASAM and SGD. Regarding ASAM, the ratio



(a) VGG19 (b) ResNet50

Figure 5.4: **Sharpness of Models Trained on CIFAR10 for Different Optimisers:** EKFAC learns sharper solutions than ASAM and SGD, which translate to worse performance (Table 5.2).

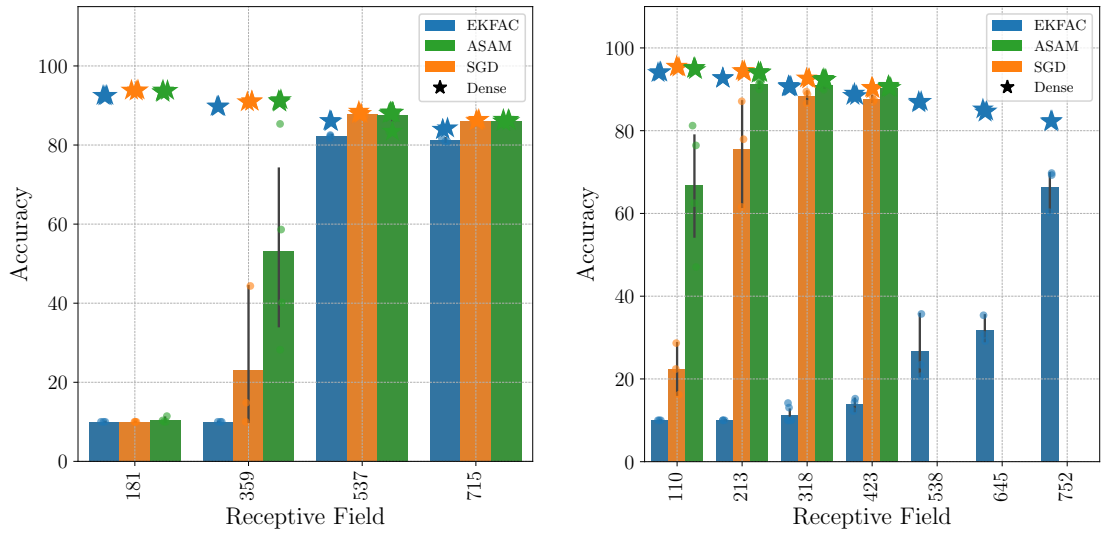
ρ, which determines the local neighbourhood used to calculate the degree of sharpness minimisation, remains excessively high, encompassing several steep directions of change, thereby marginally impairing its performance.

Second-Order Information Optimiser Learns Sharper Solutions and a Richer Set of Filters

This section hypothesises that EKFAC, as the worst performer of the optimisers used, actually learns sharper solutions than ASAM and SGD (Figure 5.4). The sharpness of ResNet50 and VGG19 trained on CIFAR10 is calculated for different receptive fields. The adaptive worst-case ℓ_∞ sharpness is calculated as in (Andriushchenko *et al.*, 2023): For arbitrary $\boldsymbol{w} \in \mathbb{R}^p$ (i.e., not necessarily a minimum), *adaptive worst-case m-sharpness* with radius ρ and with respect to a vector $\boldsymbol{c} \in \mathbb{R}^p$ is defined as:

$$S^{\rho}_{max}(\boldsymbol{w}, \boldsymbol{c}) \triangleq \mathbb{E}_{\mathcal{S}\sim P_m} \max_{\|\boldsymbol{\delta}\odot\boldsymbol{c}^{-1}\|_p \leq \rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\delta}) - L_{\mathcal{S}}(\boldsymbol{w}),$$

where $\odot$ and $\cdot^{-1}$ denote element-wise multiplication and inversion, respectively, and P_m is the data distribution that returns m training pairs $(\boldsymbol{x}, \boldsymbol{y})$.



(a) VGG19 (b) ResNet50

Figure 5.5: **One-shot Accuracy for Models Trained by EKFAC, ASAM and SGD on CIFAR10:** Pruned accuracy is recovered as the receptive field increases across all optimisers. Additionally, EKFAC is the least prone to pruning among all optimisers, consistent with the observation that EKFAC finds sharper minimisers.

Confirming the initial hypothesis, Figure 5.4 shows that EKFAC learns sharper solutions than SGD and ASAM. The models yielded similar sharpness for all receptive fields, but large differences between optimisers.

Furthermore, models trained with EKFAC will display lower prunability (i.g, a large gap between pruned and unpruned models), as sharp solutions are more vulnerable to weight perturbations, such as pruning. Figure 5.5 shows pruned VGG19 and ResNet50 models trained on CIFAR10 with a pruning rate of 90% for different receptive fields. The accuracy of the models trained with EKFAC is lower than that of the models trained with ASAM or SGD. Also, the small variations in sharpness cannot explain the effect on prunability caused by the increase of the receptive field.

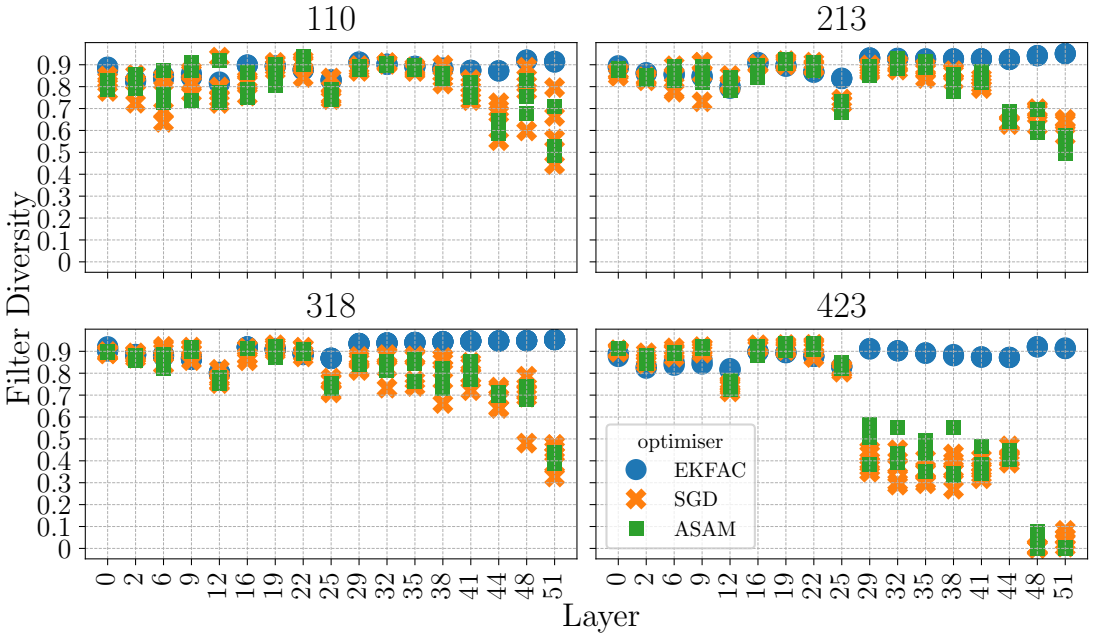


Figure 5.6: **Filter Diversity of ResNet50 Trained by EKFAC, ASAM and SGD on CIFAR10:** EKFAC has the highest filter diversity among all optimisers. Filter diversity seems to be constant as the receptive field increases.

Another possible reason EKFAC is less prunable is that it learns a more diverse set of filters, making solutions less redundant. Given that EKFAC learns a more diverse set of filters, each learnt filter carries important information. Thus, if the network is heavily pruned, the probability that filters meaningfully and uniquely contribute to classification increases. Following the methodology of (Gavrikov & Keuper, 2022), filter diversity was quantified as the Shannon entropy of the explained variance ratios for each principal component derived from a singular value decomposition (SVD) of all filters in a specific layer. As this metric approaches zero, it indicates that most filters can be approximated by a single principal component. Conversely, when this measure approaches 1, it indicates a more uniform distribution of singular values, suggesting that the filters exhibit greater randomness and uniqueness.

Figures 5.6 and 5.7 shows the filter diversity of 5 models and plots the average. Note that only layers with filters of dimensions of 3×3 or higher are included in the

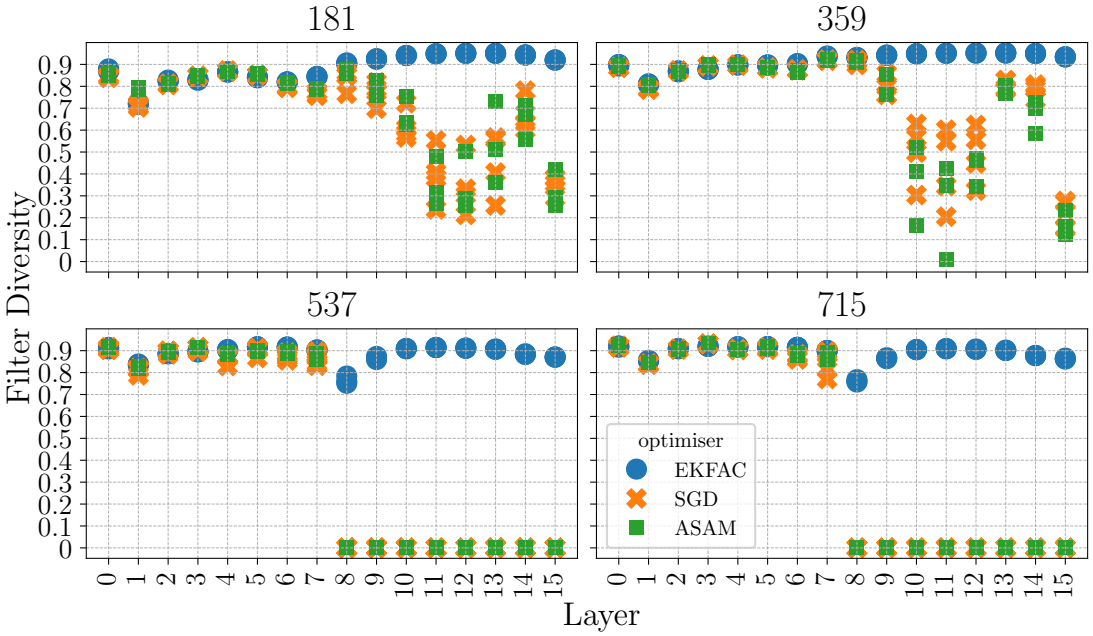


Figure 5.7: **Filter Diversity of VGG19 Trained by EKFAC, ASAM and SGD on CIFAR10:** EKFAC has the highest filter diversity among all optimisers. Filter diversity seems to be constant as the receptive field increases.

calculations. Both models trained with EKFAC have an elevated filter diversity in deeper layers. Additionally, in ASAM and SGD, filter diversity decreases substantially as the receptive field increases, providing an alternative explanation for why increasing the receptive field improves prunability. In contrast, EKFAC maintains nearly constant filter diversity, suggesting that the reduced prunability of EKFAC cannot be attributed solely to filter diversity; otherwise, a similar decrease would be expected in larger receptive fields.

It is important to note that neither sharpness nor filter diversity can explain the increased prunability of EKFAC models as the receptive field increases, as both metrics remain constant. This paradox suggests that a different mechanism underlies EKFAC's improved prunability with larger receptive fields.

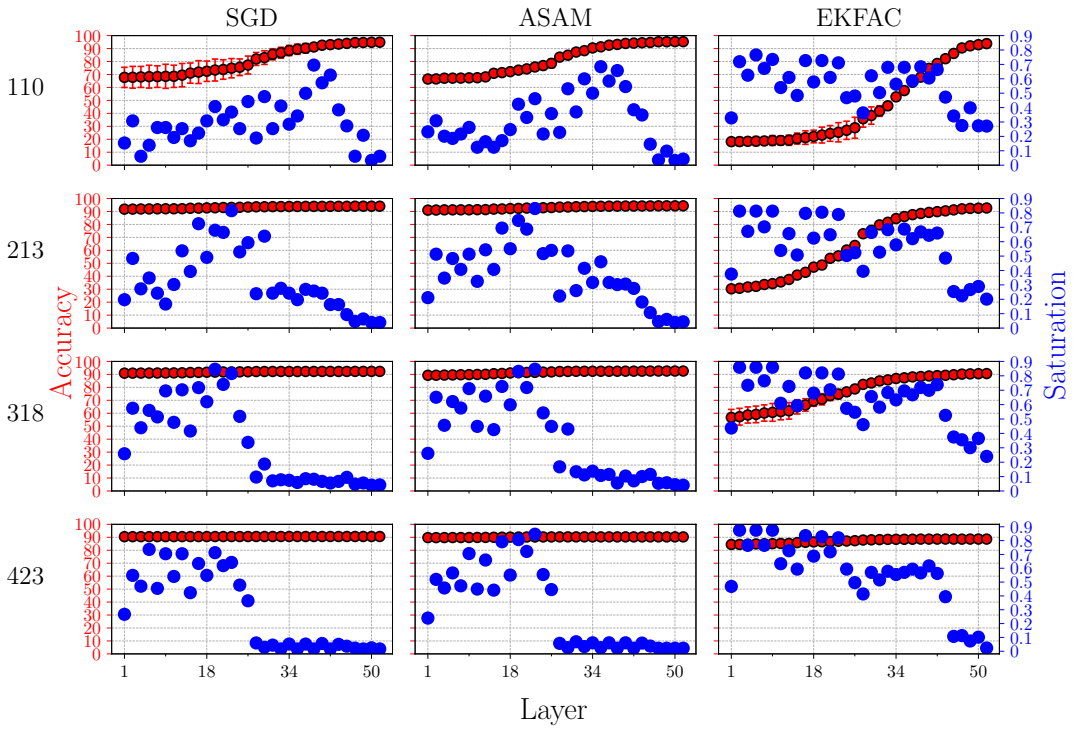


Figure 5.8: **Layer Saturation and Pruned Accuracy for ResNet50 in-block Layers Trained on CIFAR10**: For definition of in-block layers refer to Figure 4.2. Consistent with the results in Chapter 4, as increase the receptive field, saturation of layers across all optimisers drops (after 25^{th} layer) but with the caveat that EKFAC's layers are significantly more saturated than those trained with SGD and ASAM, explaining the gap in pruned accuracy between SGD, ASAM and EKFAC in Figure 5.5b.

Layer Saturation and Second-Order Optimisers

Layer saturation provides a more compelling explanation for this phenomenon. As demonstrated in Figures 5.8 to 5.10, models trained with EKFAC exhibit a higher layer saturation compared to those trained with ASAM and SGD. The accuracy shown in those figures represents the accuracy after pruning 90% of all weights present in the i^{th} layer to the last layer. For ResNet50, the type of layers pruned is consistent across figures (either only in-block layers or only out-of-block layers).

In particular, for the in-block layers of ResNet50 (Figure 5.8), starting around

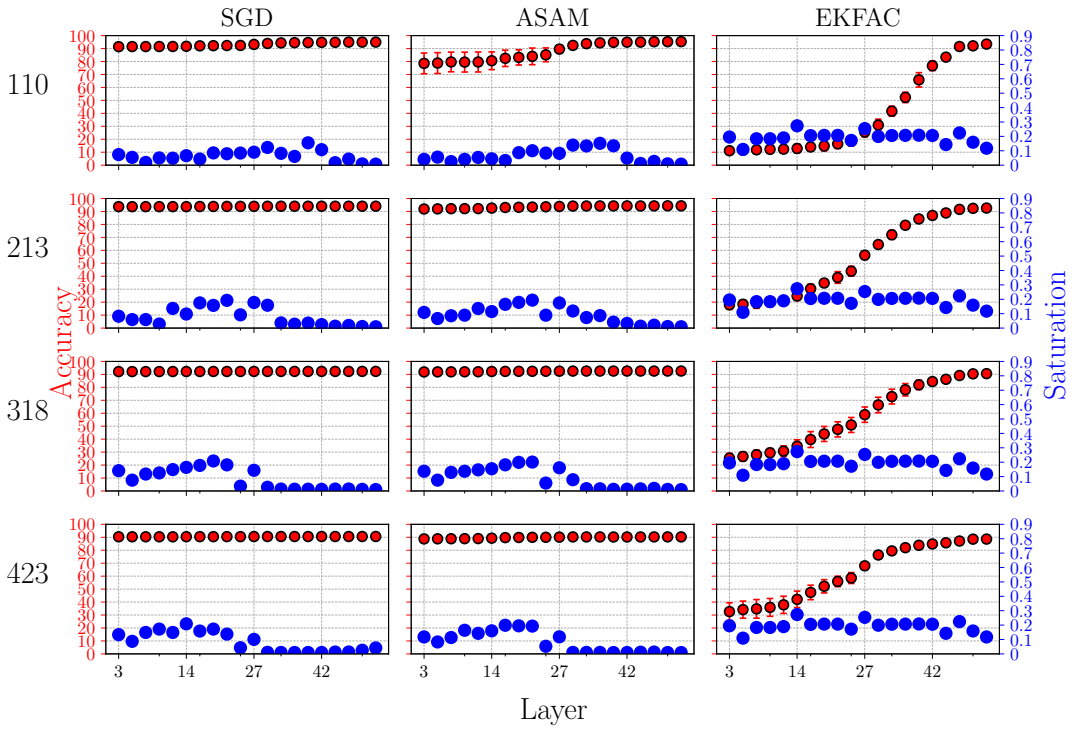


Figure 5.9: **Layer Saturation and Pruned Accuracy for ResNet50 out-of-block Layers Trained on CIFAR10:** For definition of out-of-block layers refer to Figure 4.2.All optimisers, except for EKFAC, show low (< 0.2) saturation on all out-of-block layers. EKFAC has an average saturation of around 0.2, which is enough to influence pruned accuracy.

layer 34, both ASAM and SGD achieve very low saturation (< 0.2) for receptive fields 318 and 423. In contrast, EKFAC maintains high saturation (≈ 0.6) around layer 35, which declines only after layer 50. For out-of-block layers (Figure 5.9), models trained with ASAM and SGD exhibit an average saturation below 0.2 for all layers, while EKFAC maintains an average saturation of 0.2.

Importantly, the behaviour of EKFAC relative to the other optimisers differs substantially between in-block and out-of-block layers of ResNet50. For the 423 receptive field, accuracy does not decline significantly when all in-block layers are pruned; however, because out-of-block layers are relatively saturated (compared to ASAM and SGD), pruning them significantly reduces accuracy. In contrast, ASAM

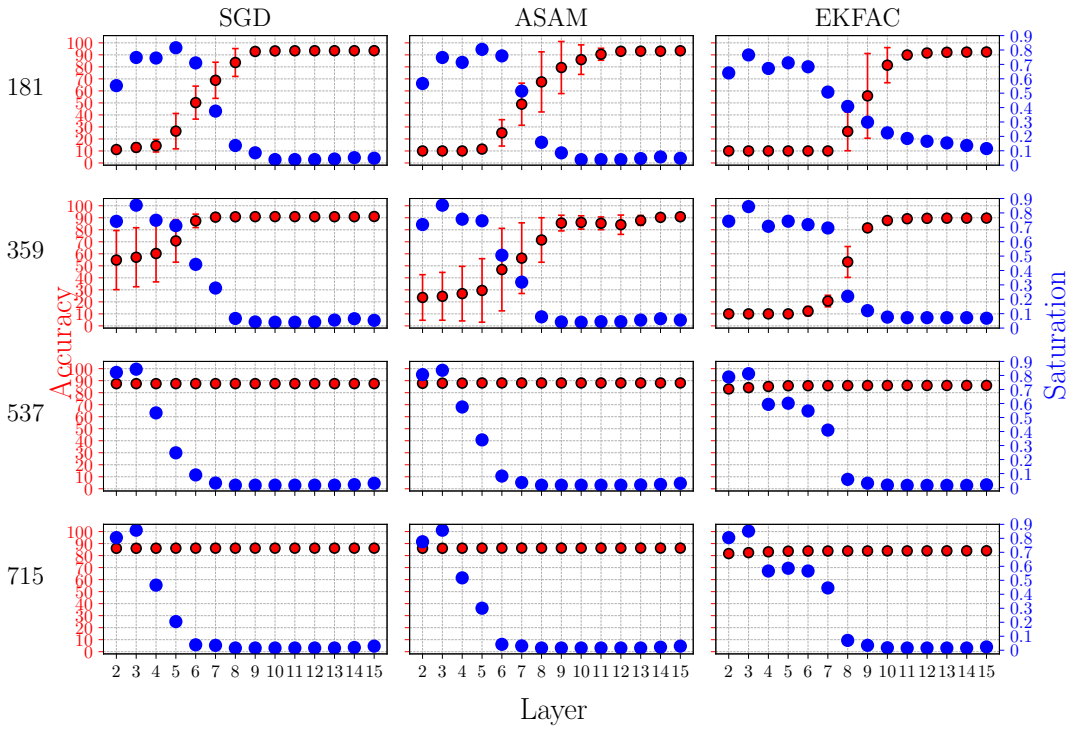


Figure 5.10: **Layer Saturation and Pruned accuracy for VGG19 trained CIFAR10**: Around layer 6, for ASAM and SGD and layer 8, for EKFAC, the saturation of all subsequent layers starts to decline, explaining the increase in pruned accuracy.

and SGD exhibit minimal accuracy drop when all out-of-block layers are pruned (due to their low saturation), and as in-block layers become more prunable with increasing receptive field, the entire model becomes more prunable. For VGG19 (Figure 5.10), the EKFAC layer saturation begins to decline around the 8th layer as the receptive field increases.

Section B.1 shows that the choice of optimiser heavily influences weight distribution, which in turn appears to affect prunability. However, these results were not included in the main analysis because they did not reliably explain increases in prunability when modulating the receptive field, as with filter diversity. Figure B.1 compares the width of the weight magnitude distribution against dense and pruned accuracy using a pruning rate of 90%.

5.4 Conclusion

This chapter explores the effect of the receptive field on the loss landscape, showing that a large receptive field widens the Hessian spectrum of models, making them more challenging to train. Surprisingly, despite intuition to the contrary, curvature-aware optimisers perform worse than SGD; this is because they follow steep directions of change, which leads them to converge to inferior local minima.

Next, models trained with second-order optimisers are less prunable, and the evidence suggests that this is due to the sharpness of the solutions found, the filter diversity, and the saturation of the model's layers. This suggests that any factor, whether architectural or training-related, that directly affects these three attributes also affects prunability. Consequently, training methods that explicitly control the weight distribution can be designed to make models more robust to magnitude-based pruning methods.

Together, these findings highlight three broader implications. First, given that large receptive fields widen the Hessian spectrum, techniques that implicitly narrow the Hessian or stage the receptive field growth (e.g., dilation schedules, hierarchical receptive field growth) may improve conditioning before expanding the receptive field further. Thus, it would be possible to obtain models that are amenable to both training and pruning. Second, the choice of the optimiser should be considered part of the pruning pipeline, and SGD should be used as a baseline, as second-order preconditioning (Gupta *et al.*, 2018) can produce weights less amenable to pruning. Third, controlling activation statistics offers a path toward more prunable models. Methods for avoiding activation saturation could increase the headroom of magnitude-based pruning; on the other hand, if curvature-aware methods are preferred (Gupta *et al.*, 2018) for convergence, pairing them with pruning criteria beyond raw magnitudes, such as Synaptic Flow (Tanaka *et al.*, 2020), Gradient Signal (Wang *et al.*, 2019a) or activation-aware saliency scores (He *et al.*, 2019; Lin *et al.*, 2020; Pham *et al.*, 2024; Wang *et al.*, 2021; Zhang *et al.*, 2023; Zhang & Wang, 2022) may recover accuracy.

By situating the results in this way, the chapter contributes not only with new empirical evidence but also with practical guidance for architectural design, optimiser selection, and pruning methodology. These insights can inform both

the design of training protocols for sparse networks and the development of new pruning strategies that better account for optimisation dynamics.

Chapter 6

Conclusion and Discussion

Modern deep learning operates within an intriguing dichotomy. The models we deploy with remarkable efficiency often originate as extensive, over-parameterised systems. It is well established that these networks can have most of their weights removed and yet maintain nearly equivalent performance (Frankle & Carbin, 2018). However, paradoxically, such extensive redundancy appears to be crucial during the training phase (Frankle *et al.*, 2020b). Expansive parameter spaces provide an enabling ground for optimisation, while the final solution occupies merely a marginal portion of that space. This raises a deeper question: What makes a trained network susceptible to compression, and why are some models amenable to pruning, whereas others are not?

Answering this question requires moving beyond viewing pruning as a mere afterthought. It is not enough to ask how many weights can be removed; it is necessary to ask what structural and dynamical properties make removal possible without destroying function. Throughout this thesis, the evidence points to a common theme: prunability is not an isolated characteristic intrinsic to a weight configuration. Instead, it arises from the complex interaction among architecture, optimisation processes, and the internal statistics that govern inference. These forces shape the topology of the loss landscape, the distribution of redundancy, and the robustness of representations against perturbations.

The work presented here reframes pruning as a design principle rather than a post-hoc operation. By examining how networks behave under stochastic perturbations, how receptive fields influence redundancy, and how optimisation strategies

shape the solutions obtained, this thesis offers three complementary perspectives on engineering prunability.

First perspective: Basin exploration for sparse solutions. Inspired by the evidence that trainable pruned networks found by the Lottery Ticket Hypothesis (Evci *et al.*, 2019) share the same basin of attraction as the dense networks they originate from, and are robust to minor perturbations; I pose the research question, Is it possible to find high-performing sparse solutions by exploring the basin of attraction of a minimiser?

Chapter 3 answered this question affirmatively. Gaussian noise was added to a dense model, and followed by pruning the perturbed networks. Sparse models derived from the original dense model can outperform the original noiseless pruned model (Figure 3.4 and Table 3.1). This added noise reduces the variance of internal features, thereby improving one-shot accuracy (Table 3.2). This advantage is transferred to the fine-tuning regime, although it depends on the pruning method used (Table 3.3). The results show that there is a trade-off between one-shot accuracy and model trainability (Figure 3.8). This trade-off is modulated by the noise level.

Second perspective: Receptive field effects on prunability. Drawing inspiration from neuroscience, where research has shown that optimal computational models for energy-constrained mouse vision systems are shallow architectures with low-resolution inputs that learn general-purpose representations (Nayebi *et al.*, 2023), Chapter 4 investigates how the receptive field (modulated by a max-pooling layer) of CNNs impacts their prunability. Analogously to the visual system of the mouse, a larger receptive field (which corresponds to low-resolution input) increases the prunability of CNNs (which corresponds to a shallow, more compressed architecture). This increase in prunability occurs when the receptive field is both smaller and larger than the input (Table 4.1 and Figure 4.3).

The reason for this increased prunability is a reduction in feature saturation (Richter *et al.*, 2021c) of deeper layers (Figures 4.8 to 4.11). Feature saturation measures the extent to which the feature maps of a given layer are saturated by the data. Finally, results suggest that for ResNet50 architecture, the redundancy in deeper layers caused by increasing the receptive field affects the similarity of network representations (Figure 4.12).

Third perspective: Loss landscape analysis and optimiser effects. Research has shown how different architectural components, such as skip connections, affect the loss landscape (Li *et al.*, 2018b). However, an understanding of how the receptive field affects the loss landscape is still missing. Even though pruning is generally regarded as a post-training compression technique, there is growing evidence that the training process itself is primarily responsible for a model's prunability. Existing work has highlighted the role of initialisation (Frankle & Carbin, 2018) and regularisation (Wang *et al.*, 2019b). However, the role of the type of optimiser, particularly those that shape the geometry of the minima, such as second-order or sharpness-aware methods, remains underexplored.

To address these gaps, two key questions are posed: How does the receptive field affect the loss landscape? And how does the choice of optimiser affect a model's final prunability?

In Chapter 5, the investigation began by measuring the Hessian spectrum of models for different receptive fields. Then the results showed that as the receptive field increases, the width of the Hessian spectrum increases, making the loss landscape more difficult to navigate for SGD (Figures 5.1 and 5.2). Then, the loss landscape is manipulated by increasing the model's width, which improves accuracy, suggesting that width changes benefit SGD's navigation through the loss landscape (Table 5.1).

Rather than modifying the landscape to accommodate SGD, more advanced optimisers, such as second-order and sharpness-aware methods, were investigated. The main question was, can these advanced optimisers better navigate the complex loss landscape?. Contrary to the general intuition, none of these optimisers could outperform SGD (Table 5.2). Additionally, second-order optimisers are less prunable than both SGD and sharpness-aware optimisers (Figure 5.5). This reduced prunability correlates with the sharpness and filter diversity of solutions found by second-order optimisers (Figures 5.4, 5.6 and 5.7).

Finally, the results demonstrate that increasing the receptive field alleviates the prunability decay incurred by second-order and sharpness-aware minimisers. This improvement is related to the phenomenon of layer saturation, where deep layers exhibit low feature saturation (Figures 5.8 to 5.10).

One important observation is that the loss landscape or weight space alone is insufficient to predict a model's prunability. For example, in Chapter 3, the dense noisy models did not show any substantial difference compared to the original solution, not in weight space, loss landscape, or accuracy. It was necessary to examine the internal feature variance (feature variance explosion) of the network to explain the increased prunability of noisy models. Similarly, in Chapter 5, when the receptive field of models trained with the second-order EKFAC optimiser is increased, neither sharpness (loss landscape) nor filter diversity (weight space) could explain the increase in prunability. Then, using the methodology in Chapter 4, the experiments showed that a large receptive field correlates with reduced saturation in the deep layers for second-order information models, thereby increasing their prunability. These two examples show that, in general, the loss landscape or weight space cannot predict the prunability of CNNs. It is necessary to have measures that monitor the network's inference dynamics, such as covariance shifts or feature saturation, to more accurately predict model prunability. This has implications for the types of regularisations that can be devised to increase the prunability of models beyond loss landscape or weight space regularisations; instead, regularisations in the variance or saturation of features could be more beneficial.

6.1 Contribution to Knowledge

Throughout Chapters 3 to 5, this thesis has contributed to the understanding of what makes a Convolutional Neural Network prunable. In Chapter 3, I showed that the internal dynamics of the model are essential not only for dense accuracy but also for pruned performance. This phenomenon has previously been briefly mentioned by (Li *et al.*, 2020), but this work systematically measured it and linked it to pruning performance.

Chapter 4 contributed a systematic analysis of the relationship between the receptive field (mediated by a max-pooling layer), the saturation of the layer, and pruned accuracy. Additionally, this chapter linked the prunability of CNN models to the similarity between their internal representations and their prunability.

Lastly, Chapter 5 sheds light on how the optimisation procedure affects layer saturation, thereby influencing prunability. Here, second-order optimisers oversaturate layers, thereby reducing model prunability.

6.2 Computational Complexity Analysis

The Stochastic Pruning method described in Chapter 3 comprises three main loops to obtain a viable population of pruned models. The first two loops search for the optimal noise level (σ) and pruning rate (γ); the number of iterations depends on how many values for each hyperparameter the researcher is willing to explore. For each parameter combination, n stochastic models must be obtained and compared with the deterministic model. For each model, the average accuracy on the test set is calculated. Thus, the computational complexity of stochastic pruning, including the hyperparameter search, is:

$$\mathcal{O}(n_\sigma n_\gamma n)$$

where n_σ and n_γ are the number of samples to explore for each hyperparameter. The experiments in this work used $n = 10$ for the results shown in Table 3.1.

For the experiments in Chapter 4, finding the best receptive field (in terms of pruned accuracy) for a particular dataset requires full model training. Thus, computational complexity depends on the number of trials allowed by the computational budget, although it is not necessary to use the entire training set to estimate the quality of a particular receptive field.

Lastly, for the results in Chapter 5, selecting the optimiser based on pruned accuracy requires training the models, similar to cross-validation runs typically used to select different training hyperparameters. The work in this thesis did not aim to propose an efficient method for selecting the best optimiser to obtain high-performing pruned networks; instead, it sought to understand how the choice of optimiser affects prunability.

6.3 Reproducibility

6.3.1 Hardware

Three high-performance computing clusters were used for the experiments in this thesis, ARC3/4 and AIRE. Next, I will categorise the results based on the chapter in which they were used.

Feature	ARC3 (Chapter 3)	ARC4 (Chapter 4)	AIRE (Chapter 5)
Server Model	GPU ARC3 Node	GPU ARC4 Node	AIRE (Dell R7615)
CPU	Broadwell E5-2650v4	Intel Xeon Gold 6138	AMD 24-core 9254 Genoa-X
Clock Speed	1.8–2.2 GHz	2.00 GHz - 3.7GHz	2.9 GHz
System Memory	256 GB	192 GB	256 GB DDR5-4800
Memory Bandwidth	800 MHz/core	800 MHz/core	800 MHz/core
GPU	NVIDIA P100	NVIDIA V100	NVIDIA L40S (48 GB)

Table 6.1: Technical specifications of the high-performance computing systems used across experimental chapters.

6.3.2 Software

This work used the Conda package manager to reproduce the same environment across all hardware platforms. The following lists the main packages used to run the experiments in this thesis, along with their versions. In Section A.6 is the exact yaml file used to reproduce the environment.

6.4 Wider Implications

In a broader context, evidence suggests that different solutions in the weight space that appear separated under linear interpolation are actually connected via more flexible curves through the high-dimensional space (Garipov *et al.*, 2018). Crucially, even when there is a nearly constant loss path between two solutions, each solution belongs to a basin that describes a different function on the data, that is, the solutions behave differently at particular data points (Fort *et al.*, 2020; Garipov *et al.*, 2018).

Library	Version	Description / Role
PyTorch (torch)	2.3.0	Deep learning framework
Torchvision	0.18.0	Computer vision utilities
CUDA (cudatoolkit)	10.2.89	GPU acceleration (cuDNN 7.6.5)
Optuna	3.3.0	Hyperparameter optimization
Delve	0.1.50	Layer saturation calculation
Matplotlib	3.4.3	Data visualization
Hydra-core	1.3.1	Experiment configuration management
Omegaconf	2.3.0	Hierarchical configuration support
Torchessian	0.1	Hessian estimation and analysis
SK-learn (scikit-learn)	1.2.1	Logistic probes and statistical modeling

Table 6.2: Main Libraries and Versions for the Experimental Environment

Given that internal network dynamics are necessary for model prunability, it is plausible that different output functions along this interpolation path yield models with varying degrees of prunability. Since each basin represents a distinct function, it is possible to characterise which output functions along the interpolation path produce the most prunable models. This insight opens the possibility of explicitly enforcing these favourable output functions during training to obtain inherently more prunable models.

Inductive Bias and Architecture

Although the experiments in this thesis were performed on CNNs for vision, analogues of the receptive field exist in other architectures and modalities. In Vision Transformers (ViTs), the effective "field" is set by the attention pattern. Pure ViTs employ global self-attention and minimal inductive bias, making them data-hungry; when data or pre-training scale are limited, they tend to underperform CNNs unless additional priors or distillation are introduced (Dosovitskiy *et al.*, 2022; Touvron *et al.*, 2021). In response, modern ViTs introduce locality through windowed or shifted-window attention, yielding a hierarchical backbone whose local windows expand the effective field across depth while preserving efficiency (Liu *et al.*, 2021). Sparse or sliding-window patterns such as Longformer

likewise impose a local field with occasional global tokens, producing an effective field that grows with depth at linear cost (Beltagy *et al.*, 2020). These designs mirror the observation that receptive field design controls where information accumulates: larger receptive fields encourage useful computation to form in shallow stages, so later layers have more representational slack and are easier to prune. In the language of attention, that slack may manifest as redundant heads, tokens, blocks, or even entire layers (Gromov *et al.*, 2024) that can be removed with a small loss in accuracy.

The same picture extends to other modalities. In time-series models such as WaveNet, dilation schedules are the one-dimensional analogue of receptive field design: staged dilation can ensure that early layers capture the most variance, making later dilated stacks more expendable (van den Oord *et al.*, 2016). In graph neural networks, the field is the k-hop neighbourhood exposed by message passing. Increasing k widens context but risks oversmoothing and oversquashing, which can create bottlenecks rather than redundancy (Nikolentzos *et al.*, 2019; Pei *et al.*, 2024). Adaptive schemes such as GeniePath learn breadth and depth of aggregation and provide a direct lever to place the computation where it is most valuable (Liu *et al.*, 2018). Across these cases, the lesson from Chapter 4 generalises: receptive field growth and locality policies can be engineered to concentrate the useful part of the computation in earlier stages, leaving deeper stages more redundant and hence more prunable.

This thesis showed that architectural priors can make models more compressible by shaping internal inference statistics. This aligns with external evidence that stronger locality priors improve data efficiency in vision models and stabilise training when pre-training scale is limited (Dosovitskiy *et al.*, 2022; Touvron *et al.*, 2021). ViT variants that introduce local windows (Swin Transformer) or Gaussian attention biases explicitly re-inject spatial priors into attention, improving performance and robustness across tasks (Kim *et al.*, 2023; Liu *et al.*, 2021). In Transformers, pruning is naturally expressed along tokens, heads, and blocks. Recent work on dynamic token sparsification (DynamicViT) shows that a large fraction of tokens can be pruned progressively and adaptively while preserving accuracy, particularly at high pruning ratios, because many tokens become redundant once sufficient context is integrated (Rao *et al.*, 2021). Hybrid strategies that combine

token pruning and pooling achieve additional savings with minimal loss (Wu *et al.*, 2024).

In addition to the findings of this thesis, this suggests a design principle: Use inductive bias to control where computation takes place. The early locality plus the later global mixing is the analogue of the attention space of "larger receptive fields concentrate useful information early in the network"; both are expected to increase the size of the pruning area because they create redundancy in deeper stages. Then once the architecture concentrates information early, later tokens and channels become dispensable. In practice, early local attention followed by late token pruning appears to be the attention-space analogue of "large receptive field, then prune deeper layers" premise of Chapter 4. The general heuristic is consistent: design for redundancy where you plan to compress, similar to what is observed with large receptive fields in CNNs (Chapter 4).

6.5 Limitations and Challenges

While this thesis provides new insights into the factors shaping prunability, its findings are bounded by practical, methodological, and conceptual limitations that define the scope of the work and point to directions for future research.

Practical Constraints

Two aspects of this work illustrate the practical challenges of applying the proposed methods at scale. First, the main drawback of stochastic pruning is the need to search for the optimal noise level for a given pruning rate. Despite that the objective of this thesis is to increase the efficiency of neural networks, the search of optimal parameters is a necessary step that could be computationally burdening. The search peformed for SP involves only inference on the validation set using n models for specific noise levels and pruning rates (tested with $n = 10$ models). The computational cost in FLOPS to determine the optimal parameters can become significant, depending on n and the number of combinations of parameters. Efficient methods for estimating these optimal parameters are necessary for future research.

Second, selecting a receptive field that improves pruned performance without compromising efficiency is similarly non-trivial. Although effective heuristics, such as aligning the receptive field with the input size (Koutini *et al.*, 2019; Richter *et al.*, 2021b), are available, they may not universally apply across varying models and datasets. In the investigation of the receptive field, empirical training is imperative to assess the quality of the chosen receptive field, in terms of pruned performance. This requirement may render the exploration phase computationally expensive.

The findings in this thesis advance a practical account of what makes convolutional neural networks prunable, yet they rest on design choices and scope conditions that define clear limitations.

Generalisability Across Architectures and Data Regimes

All experiments in this thesis focused on convolutional models for image classification, where mechanisms such as receptive field growth, deep-layer saturation, and structured redundancy are most evident. As noted earlier in the discussion of wider implications, there are conceptual parallels in other architectures. Attention masks and token policies in Vision Transformers, dilation schedules in temporal models, and k-hop neighbourhoods in graph networks all serve as analogues of receptive field design. These parallels suggest that the principles explored here, such as concentrating useful computation early and allowing later stages to become redundant, may extend beyond CNNs. However, whether these mechanisms operate in the same way, or whether new forms of redundancy and saturation emerge under different inductive biases, remains an open question. Models for language, audio, graphs, or multimodal inputs may organise computation differently, shaped by their structural priors and data regimes. Future work should therefore move from conceptual analogy to empirical validation, testing how architectural priors, optimisation dynamics, and context scheduling interact across these diverse settings and whether prunability can be established as a general design principle rather than a CNN-specific phenomenon.

The scope of this thesis is limited. Even within the image classification testbed, the datasets used are small, both in terms of the number of samples and number of classes. I infer that the results observed here can be generalised to a broader

setup, but explicit testing was not performed. Similarly, the selection of models is limited to a small sample of modern convolutional networks, thereby limiting the generalisability of this thesis's findings to other CNN architectures. Finally, although the ultimate goal of understanding pruning is to improve efficiency in training or inference, this thesis does not evaluate efficiency at the hardware level. This limits the extent to which this work can claim improvements in FLOPs, wall time, or throughput, which are, in the end, the measurable objectives of any work claiming to improve efficiency.

The conclusions were drawn from image benchmarks that offer balanced labels. Many real deployments face limited data, long-tailed or imbalanced distributions, and continual or streaming regimes. In such settings, redundancy can be both harmful and helpful, and the way pruning reorganises information flow may differ. The present work does not characterise the stability of saturation and feature variance diagnostics under domain shift or continual adaptation. This is an important open question for compression in the wild.

Training Pipeline and Interaction Effects

This thesis studies post-training pruning. Given that training itself affects prunability, the approach was limited compared to the types of training available today. For example, pruning-aware optimisation (Ding *et al.*, 2019), staged sparsification schedules (Frankle & Carbin, 2018), or curriculum-style adjustments to architectural elements(Yoon *et al.*, 2018). Such interventions may change the balance between flatness, saturation, and redundancy observed here. They may also alter the trade-offs between one-shot gains and fine-tuning performance that emerged for stochastic pruning. A systematic comparison of post-training approaches with training-time approaches was beyond the scope of this thesis and is needed to understand the relationship between learning and compressibility.

Several factors known to influence training and generalisation were kept fixed. Data augmentation, label noise, regularisation techniques such as dropout or weight decay, normalisation strategies, and learning rate schedules can all reshape internal statistics and the loss landscape. Their interaction with receptive field modulation, optimiser choice, and stochastic pruning was not system-

atically mapped. Similarly, the interaction between width, depth, and receptive field was only partially probed; width helped SGD in rugged landscapes, but a full design-space study was beyond scope.

Beyond Accuracy and Towards Practical Efficiency

Prunability in this thesis was primarily assessed through accuracy recovery before and after fine-tuning. This focus reflects common practice in compression research, yet accuracy alone is not sufficient for many real-world applications. Robustness to common corruptions, distribution shift, and adversarial perturbations, as well as calibration, uncertainty estimation, and interpretability, were not evaluated. Pruning alters internal representations and activation statistics, which may affect these properties in ways that accuracy does not capture. A broader evaluation suite is needed to determine when compression is safe to use.

At the same time, pruning is often motivated by efficiency, but this work did not quantify end-to-end system costs. Metrics such as wall-clock time, energy consumption, latency, carbon and memory footprint on specific hardware targets were not measured. Changes to receptive field size influence feature map dimensions and memory traffic, and different sparsity patterns interact very differently with hardware accelerators. Performance portability across CPUs, GPUs, and edge devices remains an unresolved challenge. Future work should therefore combine algorithmic and systems-level perspectives, linking architectural choices and pruning strategies to measurable gains in speed, energy efficiency, and sustainability.

Theoretical and Conceptual Gaps

The analysis is empirical by design. It uses diagnostic tools such as layer saturation, feature variance ratios, representational probes, and Hessian spectrum estimates. These tools show patterns, but do not provide guarantees. For example, reduced saturation in deep layers correlates with increased prunability across architectures and datasets, yet a formal link between saturation, curvature, and recoverable accuracy is still missing. A complementary theoretical account that connects activation statistics, landscape geometry, and sparsity patterns would strengthen the explanatory power of the results.

Although the empirical analysis in this thesis revealed consistent patterns, it also emphasised the lack of formal guarantees and highlighted the need for future research to develop theoretical frameworks that explain these mechanisms and translate them into principled design and training strategies. First, loss landscape descriptors and weight space proximity alone did not predict compressibility. Diagnostics of the inference dynamics, such as feature variance and saturation, aligned more closely with observed prunability. This suggests that future methods should monitor, and perhaps regularise, internal statistics directly. Second, prunability appears to be an outcome that can be engineered. Architectural priors that concentrate useful information in shallow layers, optimiser choices that avoid saturating deep layers, and local basin exploration that mitigates variance pathologies together increase the likelihood of compression. What remains unclear is how to turn these insights into principled training objectives, schedules, or search strategies that produce prunable models by design.

6.6 Final Remarks

This thesis has shown that prunability is not an incidental property but one shaped by architecture, training dynamics, and internal inference statistics. By examining stochastic perturbations, receptive field design, and optimiser choice, is demonstrated throughout this thesis that where and how computation is placed in a network strongly influences its compressibility. These findings suggest a broader principle: models can be designed and trained with pruning in mind, rather than compressed as an afterthought. Future work should extend these ideas beyond CNNs, explore their interaction with inductive bias in architectures such as Vision Transformers and graph networks, and connect empirical patterns to formal theory. Ultimately, the goal is to move from pruning as a reactive step to prunability as an integrated design objective that balances accuracy, efficiency, and interpretability in next-generation learning systems.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 88

Alain, G. & Bengio, Y. (2018). Understanding intermediate layers using linear classifier probes. 17, 67

Andriushchenko, M., Croce, F., Müller, M., Hein, M. & Flammarion, N. (2023). A Modern Look at the Relationship between Sharpness and Generalization. In *Proceedings of the 40th International Conference on Machine Learning*, 840–902, PMLR. 89

Araujo, A., Norris, W. & Sim, J. (2019). Computing receptive fields of convolutional neural networks. *Distill*. 15, 56, 61

Beltagy, I., Peters, M.E. & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*. 105

Bengio, Y. & Delalleau, O. (2011). On the Expressive Power of Deep Architectures. In J. Kivinen, C. Szepesvári, E. Ukkonen & T. Zeugmann, eds., *Algorithmic Learning Theory*, Lecture Notes in Computer Science, 18–36, Springer, Berlin, Heidelberg. 1

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, 177–186, Springer. 83

Bottou, L. & Bousquet, O. (2007). The Tradeoffs of Large Scale Learning. In *Advances in Neural Information Processing Systems*, vol. 20, Curran Associates, Inc. 83

Bottou, L., Curtis, F.E. & Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. *arXiv:1606.04838 [cs, math, stat]*. 2, 88

Cartis, C., Gould, N.I.M. & Toint, P.L. (2011). Adaptive cubic regularisation methods for unconstrained optimization. Part I: Motivation, convergence and numerical results. *Mathematical Programming*, **127**, 245–295. 24

Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B. & LeCun, Y. (2015). The Loss Surfaces of Multilayer Networks. *arXiv:1412.0233 [cs]*. 3, 83

Coates, A. & Ng, A. (2011). Selecting Receptive Fields in Deep Networks. In *Advances in Neural Information Processing Systems*, vol. 24, Curran Associates, Inc. 58

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**, 303–314. 1

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182–197. 43

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. 65

Dettmers, T. & Zettlemoyer, L. (2019). Sparse Networks from Scratch: Faster Training without Losing Performance. *arXiv:1907.04840 [cs, stat]*. 34

Devlin, J., Chang, M.W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long and Short Papers), 4171–4186, Association for Computational Linguistics, Minneapolis, Minnesota. 1

Ding, X., Ding, G., Guo, Y. & Han, J. (2019). Centripetal SGD for Pruning Very Deep Convolutional Networks With Complicated Structure. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4938–4948. 108

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. & Houlsby, N. (2022). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. 5, 51, 104, 105

Duan, Y., Chen, X., Houthooft, R., Schulman, J. & Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, 1329–1338, PMLR. 1

Evci, U., Pedregosa, F., Gomez, A. & Elsen, E. (2019). The Difficulty of Training Sparse Neural Networks. In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*. 12, 29, 32, 99

Evci, U., Gale, T., Menick, J., Castro, P.S. & Elsen, E. (2020a). Rigging the lottery: Making all tickets winners. In H.D. III & A. Singh, eds., *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, 2943–2952, PMLR. 12, 13, 15, 29, 34, 80

Evci, U., Ioannou, Y.A., Keskin, C. & Dauphin, Y. (2020b). Gradient Flow in Sparse Neural Networks and How Lottery Tickets Win. *arXiv:2010.03533 [cs]*. 51

Evci, U., Ioannou, Y., Keskin, C. & Dauphin, Y. (2022). Gradient Flow in Sparse Neural Networks and How Lottery Tickets Win. *Proceedings of the AAAI Conference on Artificial Intelligence*, **36**, 6577–6586. 14, 16, 29, 33, 48

Fedus, W., Zoph, B. & Shazeer, N. (2022). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research*, **23**, 1–39. 2

Foret, P., Kleiner, A., Mobahi, H. & Neyshabur, B. (2021). Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*. 26, 27, 28

Fort, S. & Ganguli, S. (2019). Emergent properties of the local geometry of neural loss landscapes. 22

Fort, S., Hu, H. & Lakshminarayanan, B. (2020). Deep Ensembles: A Loss Landscape Perspective. 103

Frankle, J. & Carbin, M. (2018). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*. 4, 13, 28, 32, 34, 98, 100, 108

Frankle, J., Dziugaite, G.K., Roy, D. & Carbin, M. (2020a). Linear Mode Connectivity and the Lottery Ticket Hypothesis. In *Proceedings of the 37th International Conference on Machine Learning*, 3259–3269, PMLR. 37

Frankle, J., Dziugaite, G.K., Roy, D. & Carbin, M. (2020b). Pruning Neural Networks at Initialization: Why Are We Missing the Mark? In *International Conference on Learning Representations*. 4, 98

Gao, S., Li, Z.Y., Han, Q., Cheng, M.M. & Wang, L. (2023). RF-Next: Efficient Receptive Field Search for Convolutional Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **45**, 2984–3002. 57

Garbin, C., Zhu, X. & Marques, O. (2020). Dropout vs. batch normalization: An empirical study of their impact to deep learning. *Multimedia Tools and Applications*, **79**, 12777–12815. 2

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. & Wilson, A.G. (2018). Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. *arXiv:1802.10026 [cs, stat]*. 12, 103

Gavrikov, P. & Keuper, J. (2022). CNN Filter DB: An Empirical Investigation of Trained Convolutional Filters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19066–19076. 91

George, T., Laurent, C., Bouthillier, X., Ballas, N. & Vincent, P. (2018). Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis. In *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc. 7, 82

Ghorbani, B., Krishnan, S. & Xiao, Y. (2019). An Investigation into Neural Net Optimization via Hessian Eigenvalue Density. In *Proceedings of the 36th International Conference on Machine Learning*, 2232–2241, PMLR. 21, 22, 23, 24, 25

Golomb, J.D. & Kanwisher, N. (2012). Higher Level Visual Cortex Represents Retinotopic, Not Spatiotopic, Object Location. *Cerebral Cortex (New York, NY)*, **22**, 2794–2810. 53

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press. 2

Gromov, A., Tirumala, K., Shapourian, H., Glorioso, P. & Roberts, D.A. (2024). The Unreasonable Ineffectiveness of the Deeper Layers. 105

Gupta, M., Camci, E., Keneta, V.R., Vaidyanathan, A., Kanodia, R., James, A., Foo, C.S., Wu, M. & Lin, J. (2024). Is Complexity Required for Neural Network Pruning? A Case Study on Global Magnitude Pruning. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, 747–754. 35, 60

Gupta, V., Koren, T. & Singer, Y. (2018). Shampoo: Preconditioned Stochastic Tensor Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, 1842–1850, PMLR. 96

Gur-Ari, G., Roberts, D.A. & Dyer, E. (2018). Gradient Descent Happens in a Tiny Subspace. *arXiv:1812.04754 [cs, stat]*. 22, 23

Han, S., Pool, J., Tran, J. & Dally, W. (2015). Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc. 35

Hassibi, B., Stork, D. & Wolff, G. (1993). Optimal Brain Surgeon: Extensions and performance comparisons. In *Advances in Neural Information Processing Systems*, vol. 6, Morgan-Kaufmann. 30

He, F., Wang, B. & Tao, D. (2020). Nonlinearities in activations substantially shape the loss surfaces of neural networks. *undefined*. 3

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*. 7, 61

He, Y., Liu, P., Wang, Z., Hu, Z. & Yang, Y. (2019). Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4335–4344. 96

Hernandez, D. & Brown, T.B. (2020). Measuring the Algorithmic Efficiency of Neural Networks. *arXiv:2005.04305 [cs, stat]*. 3

Hinton, G.E. & Salakhutdinov, R.R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, **313**, 504–507. 26

Hochreiter, S. & Schmidhuber, J. (1997). Flat Minima. *Neural Computation*, **9**, 1–42. 3, 27

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 61

Huang, W.R., Emam, Z., Goldblum, M., Fowl, L., Terry, J.K., Huang, F. & Goldstein, T. (2020). Understanding Generalization through Visualizations. *arXiv:1906.03291 [cs, stat]*. 3, 10, 11, 12

Hubel, D.H. & Wiesel, T.N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, **160**, 106–154.2. 52

Hubel, D.H. & Wiesel, T.N. (1965). Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of Neurophysiology*, **28**, 229–289. 53

Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 22, 51

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D. & Wilson, A.G. (2019). Averaging Weights Leads to Wider Optima and Better Generalization. *arXiv:1803.05407 [cs, stat]*. 51

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D. & Bengio, S. (2019). Fantastic Generalization Measures and Where to Find Them. *arXiv:1912.02178 [cs, stat]*. 20, 27

Karnin, E. (1990). A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, **1**, 239–242. 35

Kawadkar, K. (2025). Comparative Analysis of Vision Transformers and Convolutional Neural Networks for Medical Image Classification. 5

Kawaguchi, K. (2016). Deep Learning without Poor Local Minima. In *NIPS*. 83

Kawaguchi, K. & Bengio, Y. (2019). Depth with nonlinearity creates no bad local minima in ResNets. *Neural Networks*, **118**, 167–174. 10, 83

Kawaguchi, K., Huang, J. & Kaelbling, L.P. (2019). Effect of depth and width on local minima in deep learning. *Neural computation*, **31**, 1462–1498. 10, 83

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. *arXiv:1609.04836 [cs, math]*. 3

Kim, B.J., Choi, H., Jang, H. & Kim, S.W. (2023). Understanding Gaussian attention bias of Vision Transformers using effective receptive fields. *arXiv preprint arXiv:2305.04722*. 105

Kingma, D.P. & Ba, J. (2017). Adam: A Method for Stochastic Optimization. 14

Kobayashi, G. & Shouno, H. (2020). Interpretation of ResNet by Visualization of Preferred Stimulus in Receptive Fields. 57

Kornblith, S., Norouzi, M., Lee, H. & Hinton, G. (2019). Similarity of Neural Network Representations Revisited. 73

Koutini, K., Eghbal-zadeh, H., Dorfer, M. & Widmer, G. (2019). The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification. In *2019 27th European Signal Processing Conference (EUSIPCO)*, 1–5. 15, 53, 58, 83, 84, 107

Krizhevsky, A., Sutskever, I. & Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc. 3

Kwon, J., Kim, J., Park, H. & Choi, I.K. (2021). ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks. 7, 82

Le, Y. & Yang, X. (2015). Tiny imagenet visual recognition challenge. *CS 231N*, **7**, 3. 66

LeCun, Y., Denker, J. & Solla, S. (1989). Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, vol. 2, Morgan-Kaufmann. 30, 35

Lee, J., Park, S., Mo, S., Ahn, S. & Shin, J. (2022). Layer-adaptive Sparsity for the Magnitude-based Pruning. In *International Conference on Learning Representations*. 29, 35, 36, 47

Lee, N., Ajanthan, T. & Torr, P. (2018). SNIP:Single-Shot Network Pruning Based on Connection Sensitivity. In *International Conference on Learning Representations*. 34, 35

Lehky, S.R. & Sejnowski, T.J. (1988). Network model of shape-from-shading: Neural function arises from both receptive and projective fields. *Nature*, **333**, 452–454. 52

Li, B., Wu, B., Su, J. & Wang, G. (2020). EagleEye: Fast Sub-net Evaluation for Efficient Neural Network Pruning. In A. Vedaldi, H. Bischof, T. Brox & J.M. Frahm, eds., *Computer Vision – ECCV 2020*, vol. 12347, 639–654, Springer International Publishing, Cham. 51, 66, 101

Li, D., Ding, T. & Sun, R. (2018a). Over-Parameterized Deep Neural Networks Have No Strict Local Minima For Any Continuous Activations. *ArXiv*. 3

Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. (2018b). Visualizing the Loss Landscape of Neural Nets. 10, 11, 22, 100

Li, Y., Lin, S., Liu, J., Ye, Q., Wang, M., Chao, F., Yang, F., Ma, J., Tian, Q. & Ji, R. (2021). Towards Compact CNNs via Collaborative Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6438–6447. 37

Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y. & Shao, L. (2020). HRank: Filter Pruning Using High-Rank Feature Map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1529–1538. 96

Liu, D.C. & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, **45**, 503–528. 19

Liu, J., Xu, Z., Shi, R., Cheung, R.C.C. & So, H.K.H. (2020). Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers. In *Eighth International Conference on Learning Representations*. 12, 29

Liu, Z., Chen, C., Li, L., Zhou, J. & Sun, M. (2018). GeniePath: Graph neural networks with adaptive receptive paths. In *Proceedings of the ACM SIGKDD Workshop on Mining and Learning with Graphs (MLG)*. 105

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. & Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022. 5, 104, 105

Luo, W., Li, Y., Urtasun, R. & Zemel, R. (2016). Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc. 53, 57

Martens, J. & Grosse, R. (2015). Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, 2408–2417, PMLR. 25

Massed Compute (2025). FAQ answers. 54

Mhaskar, H. & Poggio, T. (2016). Deep vs. shallow networks : An approximation theory perspective. *arXiv:1608.03287 [cs, math]*. 1

Mhaskar, H.N. (1993). Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, **1**, 61–80. 2

Mocanu, D.C., Mocanu, E., Stone, P., Nguyen, P.H., Gibescu, M. & Liotta, A. (2018). Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, **9**, 2383. 12, 15, 29, 34, 80

Mozer, M.C. & Smolensky, P. (1988). Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment. In *Advances in Neural Information Processing Systems*, vol. 1, Morgan-Kaufmann. 35

Nayebi, A., Kong, N.C.L., Zhuang, C., Gardner, J.L., Norcia, A.M. & Yamins, D.L.K. (2023). Mouse visual cortex as a limited resource system that self-learns an ecologically-general representation. 54, 99

Neelakantan, A., Vilnis, L., Le, Q.V., Sutskever, I., Kaiser, L., Kurach, K. & Martens, J. (2015). Adding Gradient Noise Improves Learning for Very Deep Networks. 37

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:1809.10749 [cs, stat]*. 3

Nikolentzos, G., Dasoulas, G. & Vazirgiannis, M. (2019). K-Hop graph neural networks. *arXiv preprint arXiv:1907.06051*. 105

Nocedal, J. & Wright, S.J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, 2nd edn. 19, 23

Okanovic, P., Kwasniewski, G., Labini, P.S., Besta, M., Vella, F. & Hoefler, T. (2024). High Performance Unstructured SpMM Computation Using Tensor Cores. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–14. 51

Olshausen, B.A. & Field, D.J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, **381**, 607–609. 53

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H.W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S.P., Forte, J., Fulford, I.,

Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S.S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Kaiser, Ł., Kamali, A., Kanitscheider, I., Keskar, N.S., Khan, T., Kilpatrick, L., Kim, J.W., Kim, C., Kim, Y., Kirchner, J.H., Kiros, J., Knight, M., Kokotajlo, D., Kondraciuk, Ł., Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C.M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S.M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., Peres, F.d.A.B., Petrov, M., Pinto, H.P.d.O., Michael, Pokorny, Pokrass, M., Pong, V.H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F.P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M.B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J.F.C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright,

C., Wang, J.J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C.J., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W. & Zoph, B. (2024). GPT-4 Technical Report. 1

Pearlmutter, B.A. (1994). Fast Exact Multiplication by the Hessian. *Neural Computation*, **6**, 147–160. 30

Pei, H., Li, Y., Deng, H., Hai, J., Wang, P., Ma, J., Guan, X. *et al.* (2024). Multi-Track Message Passing: Tackling oversmoothing and oversquashing in graph learning via preventing heterophily mixing. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. 105

Pham, V.T., Zniyed, Y. & Nguyen, T.P. (2024). Efficient tensor decomposition-based filter pruning. *Neural Networks*, **178**, 106393. 96

Rao, R.P.N. & Ballard, D.H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, **2**, 79–87. 52

Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J. & Hsieh, C.J. (2021). DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34. 105

Richter, M.L., Byttner, W., Krumnack, U., Wiedenroth, A., Schallner, L. & Shenk, J. (2021a). (Input) Size Matters for CNN Classifiers. In I. Farkaš, P. Masulli, S. Otte & S. Wermter, eds., *Artificial Neural Networks and Machine Learning – ICANN 2021*, 133–144, Springer International Publishing, Cham. 58, 62

Richter, M.L., Schöning, J., Wiedenroth, A. & Krumnack, U. (2021b). Should You Go Deeper? Optimizing Convolutional Neural Network Architectures without Training by Receptive Field Analysis. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 964–971. 18, 76, 107

Richter, M.L., Shenk, J., Byttner, W., Arpteg, A. & Huss, M. (2021c). Feature Space Saturation during Training. 15, 17, 99

Richter, M.L., Schöning, J., Wiedenroth, A. & Krumnack, U. (2022). Receptive field analysis for optimizing convolutional neural network architectures without training. In *Deep Learning Applications, Volume 4*, 235–261, Springer. 58

Rizwan I Haque, I. & Neubert, J. (2020). Deep learning approaches to biomedical image segmentation. *Informatics in Medicine Unlocked*, **18**, 100297. 1

Ro, Y. & Choi, J.Y. (2021). AutoLR: Layer-wise pruning and auto-tuning of learning rates in fine-tuning of deep networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2486–2494. 72

Robbins, H. & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, **22**, 400–407. 3

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*, **323**, 533–536. 3

Sagun, L., Bottou, L. & LeCun, Y. (2016). Eigenvalues of the Hessian in Deep Learning: Singularity and Beyond. 21, 22

Sagun, L., Evci, U., Guney, V.U., Dauphin, Y. & Bottou, L. (2018). Empirical Analysis of the Hessian of Over-Parametrized Neural Networks. *arXiv:1706.04454 [cs]*. 21, 22, 83

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.C. (2019). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv:1801.04381 [cs]*. 61

Schraudolph, N.N. (2002). Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent. *Neural Computation*, **14**, 1723–1738. 30

Schwartz, R., Dodge, J., Smith, N.A. & Etzioni, O. (2019). Green AI. *arXiv:1907.10597 [cs, stat]*. 3

Seif, G. & Androutsos, D. (2018). Large receptive field networks for high-scale image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 763–772. 53

Senior, A.W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A.W.R., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D.T., Silver, D., Kavukcuoglu, K. & Hassabis, D. (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, **577**, 706–710. 1

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, **529**, 484–489. 1

Singh, S.P. & Alistarh, D. (2020). WoodFisher: Efficient Second-Order Approximation for Neural Network Compression. In *Advances in Neural Information Processing Systems*, vol. 33, 18098–18109, Curran Associates, Inc. 35

Sokoloff, L. (1960). The metabolism of the central nervous system in vivo. *Handbook of physiology, section I, neurophysiology*, **3**, 1843–1864. 54

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, **15**, 1929–1958. 2

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. 61

Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. 61

Tan, M. & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, 6105–6114, PMLR. 1, 28, 53

Tanaka, H., Kunin, D., Yamins, D.L. & Ganguli, S. (2020). Pruning neural networks without any data by iteratively conserving synaptic flow. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan & H. Lin, eds., *Advances in Neural Information Processing Systems*, vol. 33, 6377–6389, Curran Associates, Inc. 34, 96

Thompson, N., Greenewald, K., Lee, K. & Manso, G.F. (2023). The Computational Limits of Deep Learning. In *Computing within Limits*, LIMITS. 2, 3, 4

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A. & Jegou, H. (2021). Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, 10347–10357, PMLR. 5, 104, 105

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. & Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. In *Arxiv*. 1, 105

Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., Vezhnevets, A.S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T.L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C. & Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, **575**, 350–354. 1

Wang, C., Zhang, G. & Grosse, R. (2019a). Picking Winning Tickets Before Training by Preserving Gradient Flow. In *International Conference on Learning Representations*. 4, 29, 34, 50, 96

Wang, H., Zhang, Q., Wang, Y., Lu, Y. & Hu, H. (2019b). Structured Pruning for Efficient ConvNets via Incremental Regularization. 100

Wang, R., Gong, M. & Tao, D. (2020). Receptive Field Size Versus Model Depth for Single Image Super-Resolution. *IEEE Transactions on Image Processing*, **29**, 1669–1682. 58, 83, 84

Wang, Z., Li, C. & Wang, X. (2021). Convolutional Neural Network Pruning With Structural Redundancy Reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14913–14922. 96

Wu, X., Zeng, F., Wang, X., Chen, X. & Wang, Y. (2024). PPT: Token pruning and pooling for efficient vision transformers. *arXiv preprint arXiv:2310.01812v2*. 106

Xu, P., Roosta, F. & Mahoney, M.W. (2020). Second-order optimization for non-convex machine learning: An empirical study. In *Proceedings of the 2020 SIAM International Conference on Data Mining, SDM 2020*. 23, 25

Yan, B., Root, A.J., Gale, T., Broman, D. & Kjolstad, F. (2024). Scorch: A Library for Sparse Deep Learning. 51

Yao, Z., Gholami, A., Keutzer, K. & Mahoney, M.W. (2020). PyHessian: Neural Networks Through the Lens of the Hessian. In *2020 IEEE International Conference on Big Data (Big Data)*, 581–590. 21, 22

Yoon, J., Yang, E., Lee, J. & Hwang, S.J. (2018). Lifelong Learning with Dynamically Expandable Networks. In *International Conference on Learning Representations*. 108

Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc. 73, 79

Zhang, S., Gao, M., Ni, Q. & Han, J. (2023). Filter pruning with uniqueness mechanism in the frequency domain for efficient neural networks. *Neurocomputing*, **530**, 116–124. 96

Zhang, W. & Wang, Z. (2022). FPFS: Filter-level pruning via distance weight measuring filter similarity. *Neurocomputing*, **512**, 40–51. 96

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. & Torralba, A. (2015). Object Detectors Emerge in Deep Scene CNNs. 53

Appendix A

Receptive Field

A.1 One-shot Solutions with Multiple Pruning Rates and extra Fine-Tuning results

For each combination, 5 models were trained, pruned, and fine-tuned. The error bars correspond to the standard deviation. For all fine-tuning experiments, we used the SGD optimiser with cosine annealing learning rate schedule with $T_{max} =$ epochs, 0.9 momentum, weight decay of 5×10^{-5} and clipped gradient values to a maximum of 0.1 and fine-tuned for 200 epochs.

	RF	Dense	0.5		0.6		0.7		0.8		0.9	
			Pruned	Δ_{OS}	Pruned	Δ_{OS}	Pruned	Δ_{OS}	Pruned	Δ_{OS}	Pruned	Δ_{OS}
VGG	181	**93.52±0.12**	**93.51±0.13**	0.010±0.03	**93.48±0.11**	**0.048±0.05**	**93.32±0.10**	0.206±0.16	72.22±27.11	21.30±27.18	10.9 ± 2.03	82.6 ± 2.02
	359	91.15±0.23	91.16±0.24	**-0.004±0.01**	91.15±0.24	0.004±0.02	91.10±0.23	0.058±0.05	**90.23± 1.32**	0.92± 1.50	32.4 ± 15.7	58.7 ± 15.8
	537	87.88±0.19	87.88±0.19	0.000±0.00	87.88±0.19	0.000±0.00	87.88±0.19	0.000±0.00	87.87± 0.19	0.00± 0.00	**87.6 ± 0.30**	0.18 ± 0.16
	715	85.88±0.22	85.88±0.22	0.000±0.00	85.88±0.22	0.000±0.00	85.88±0.22	0.000±0.00	85.88± 0.22	0.00± 0.00	85.8 ± 0.22	**0.07 ± 0.04**
ResNet50	110	**94.69±0.21**	**94.71±0.20**	**-0.023±0.01**	**94.61±0.19**	0.080±0.03	**94.24±0.20**	0.453±0.01	92.34± 1.08	2.35± 0.92	56.0 ± 20.7	37.8 ± 20.5
	213	94.03±0.24	94.02±0.24	0.010±0.03	94.01±0.23	0.020±0.02	93.93±0.20	0.097±0.09	**93.81± 0.22**	0.22± 0.23	**91.7 ± 0.98**	2.25 ± 0.90
	318	92.22±0.24	92.22±0.23	-0.003±0.02	92.23±0.23	**-0.010±0.02**	92.24±0.25	**-0.020±0.01**	92.14± 0.23	0.08± 0.04	91.4 ± 0.45	0.85 ± 0.49
	423	90.23±0.17	90.23±0.17	0.000±0.00	90.23±0.17	0.000±0.00	90.23±0.18	-0.007±0.01	90.23± 0.15	**-0.003±0.04**	90.2 ± 0.37	0.18 ± 0.09

Table A.1: One-shot pruning performance for VGG and ResNet50 with pruning rates from 0.5 to 0.9 on CIFAR10.

	RF	Dense	0.5		0.6		0.7		0.8		0.9	
			Pruned	Δ_{OS}	Pruned	Δ_{OS}	Pruned	Δ_{OS}	Pruned	Δ_{OS}	Pruned	Δ_{OS}
VGG	181	**61.58±0.33**	**59.85±0.24**	1.738±0.24	**56.93±0.51**	4.656±0.66	**48.29±1.39**	13.29± 1.46	19.14±4.61	42.44±4.67	0.75±0.09	60.8±0.32
	359	53.25±0.21	51.13±0.67	2.116±0.74	47.94±1.90	5.304±1.96	40.02±2.56	13.23± 2.65	10.09±1.93	43.16±2.09	0.63±0.17	52.6±0.36
	537	41.05±1.92	41.05±1.92	**-0.004±0.02**	41.05±1.91	-0.000±0.07	40.96±1.93	0.092±0.07	40.09±2.13	0.958±0.36	16.7 ±7.50	24.3±5.81
	715	38.57±1.69	38.57±1.69	0.008±0.01	38.58±1.66	**-0.008±0.04**	38.51±1.66	**0.068±0.09**	**37.87±1.57**	**0.700±0.52**	**21.8 ±6.57**	**16.7±7.21**
ResNet50	213	**61.83±0.40**	**60.86±0.46**	0.966±0.40	**59.28±0.47**	2.552±0.49	**55.33±0.99**	6.504±1.19	40.56±3.04	21.27± 3.17	5.91±0.89	55.9±0.99
	318	59.10±0.37	58.46±0.51	0.640±0.40	57.26±0.47	1.840±0.36	54.28±0.40	4.816±0.32	43.09±1.88	16.01± 1.82	8.56±2.66	50.5±2.55
	423	56.53±0.28	56.29±0.41	**0.248±0.18**	55.97±0.46	**0.568±0.21**	54.41±0.45	**2.128±0.22**	**49.27±0.58**	**7.260±0.41**	**21.4 ±2.87**	**35.0±2.99**

Table A.2: One-shot pruning performance for VGG and ResNet50 with pruning rates from 0.5 to 0.9 on Tiny ImageNet.

			No Fine-Tuning		Fine-Tuned (10 epochs)		Fine-Tuned (100 epochs)	
	RF	Dense	Pruned	Δ_{OS}	Pruned	Δ_{FT}	Pruned	Δ_{FT}
VGG	180 *	62.7 ± 0.36	0.81 ± 0.38	61.8 ± 0.16	–	–	–	–
	181	**61.5 ± 0.33**	0.75 ± 0.09	60.8 ± 0.32	30.21±2.60	31.37±2.53	33.93± 0.32	27.66± 0.52
	359	53.2 ± 0.20	0.63 ± 0.17	52.6 ± 0.36	3.56±2.03	49.69±2.19	29.73±16.15	23.52±16.23
	537	41.0 ± 1.91	16.7 ± 7.50	24.3 ± 5.81	**38.21±2.70**	2.84±0.88	**37.50± 1.92**	3.55± 0.16
	715	38.5 ± 1.69	**21.8 ± 6.57**	**16.7 ± 7.21**	36.45±1.55	**2.13±0.76**	35.50± 1.81	**3.08± 0.45**
ResNet50	213	**61.8 ± 0.40**	5.91 ± 0.89	55.9 ± 0.99	44.56±1.21	17.27±1.35	**50.21± 0.43**	11.62± 0.49
	318	59.1 ± 0.36	8.56 ± 2.66	50.5 ± 2.55	44.28±0.65	14.82±0.63	49.83± 0.58	9.27± 0.41
	423	56.5 ± 0.27	21.4 ± 2.87	35.0 ± 2.99	**46.16±0.72**	**10.37±0.58**	49.43± 0.40	**7.11± 0.36**
	1415	32.0 ± 0.19	**32.0 ± 0.19**	0.0 ± 0.0	–	–	–	–
	1920	29.2 ± 0.62	29.2 ± 0.62	0.0 ± 0.0	–	–	–	–
	3100	22.8 ± 0.37	22.8 ± 0.37	0.0 ± 0.0	–	–	–	–

Table A.3: Fine-tuning experiments on Tiny ImageNet

A.2 Similarity for VGG Network

For these experiments, the representations of each layer of all samples from the CIFAR10 test set are compared.

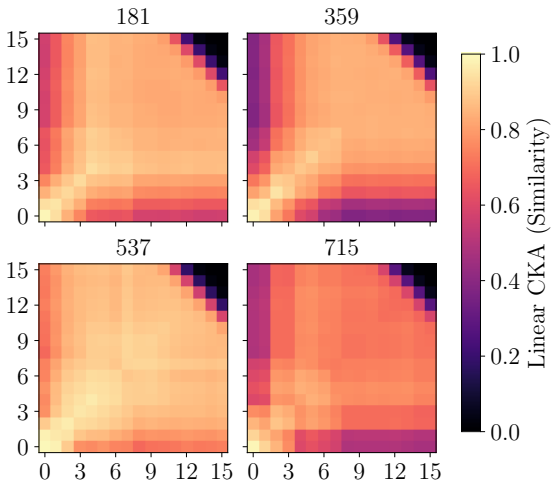


Figure A.1: Representation similarly for all convolutional layers in VGG for different receptive fields. Due to the lack of residual connections, the VGG architecture does not exhibit the same clear pattern as ResNet50. The whole test set of CIFAR10 was used for calculating the similarities.

A.3 Model Implementation details

Table A.4: VGG Implementation

RF 181	RF 359	RF 537	RF 715
conv 3×3, 64			
max pooling 2×2, stride 2	max pooling 3×3, stride 2	max pooling 4×4, stride 3	max pooling 5×5, stride 4
conv 3×3, 64			
max pooling 2×2, stride 2 [conv 3×3, 128] $\times 2$			
max pooling 2×2, stride 2 [conv 3×3, 256] $\times 4$			
max pooling 2×2, stride 2 [conv 3×3, 512] $\times 8$			
average pooling, 10/200-d fc, softmax			

Table A.5: ResNet50 Implementation

RF 110	RF 213	RF 318	RF 423	RF 1415	RF 3100
conv 3×3, 64					
mp=2×2,s=1	mp=3×3,s=2	mp=4×4,s=3	mp=5×5,s=4	mp=15×15,s=14	mp=32×32,s=31
conv 1×1, 64 conv 3×3, 64 conv 1×1, 64				$\times 3$	
conv 1×1, 128 conv 3×3, 128 conv 1×1, 128				$\times 4$	
conv 1×1, 256 conv 3×3, 256 conv 1×1, 256				$\times 6$	
conv 1×1, 518 conv 3×3, 518 conv 1×1, 518				$\times 3$	
average pooling, 10/200-output fc, softmax					

Table A.6: ResNet25 Implementation for Small ImageNet

RF 128	RF 153	RF 178	RF 203	RF 253
conv 3×3, 64				
mp=6×6 stride=5	mp=7×7 stride=6	mp=8×8 stride=7	mp=9×9 stride=8	mp=10×10 stride=9
	conv 1×1, 64 conv 1×1, 64 conv 3×3, 64 conv 1×1, 64 conv 1×1, 64			$\times 1$
	conv 1×1, 128 conv 1×1, 128 conv 3×3, 128 conv 1×1, 128 conv 1×1, 128			$\times 1$
	conv 1×1, 256 conv 1×1, 256 conv 3×3, 256 conv 1×1, 256 conv 1×1, 256			$\times 2$
	conv 1×1, 518 conv 1×1, 518 conv 3×3, 518 conv 1×1, 518 conv 1×1, 518			$\times 1$
average pooling, 200-output fc, softmax				

A.4 Additional width experiments

This section presents the results of the width experiments with ResNet50 on CIFAR10.

Receptive Field	Width		
	×1	×2	×3
110	**94.8 ±0.29**	-	-
213	94.0±0.16	**94.536 ±0.26**	94.516±0.29
318	92.2±0.18	93.14 ±0.31	93.19±0.29
1415	72.8±0.78	75.11 ±0.36	76.58±0.49
3100	50.1±0.25	53.01 ±0.35	54.49±0.45

Table A.7: Accuracy of ResNet50 model on CIFAR10 for different widths. For the case of a receptive field of 213, increasing the width 2-fold yields performance similar to that of a receptive field of 110. For very large receptive field width recovers more accuracy compared to smaller receptive fields, suggesting that the smoothing effect of increasing the width is crucial for such a large receptive field.

A.5 Dilation Experiments

This section shows the same experiments in Table 4.1 but using dilation instead of maxpooling. The same receptive field was matched via dilation, and padding was used to keep the feature map sizes equal across different receptive fields. Only the dilation of the first convolutional layer was used. It can be seen that the manipulation of the receptive field through dilation does not yield the same results seen in Table 4.1. Not only does dense accuracy not diminish as you increase the receptive field, but pruned accuracy also does not exhibit any noticeable pattern. This suggests that the results seen in Chapter 4 could be due to the reduced size of the feature maps in larger receptive fields.

		CIFAR10			Tiny ImageNet		
			No Fine-Tuning			No Fine-Tuning	
Model	**RF**	**Dense**	$A_{os}\uparrow$	$\Delta_{os}\downarrow$	**Dense**	$A_{os}\uparrow$	$\Delta_{os}\downarrow$
VGG	180*	**94.0±0.24**	10.04±0.16	83.9±0.318	**62.5±0.30**	0.92±0.30	61.6±0.26
	359	93.22±0.11	10±0	83.22±0.11	61.03±0.25	**1.1±0.44**	**59.93±0.52**
	537	93.33±0.17	10±0	83.33±0.17	61.21±0.16	1.06±0.28	60.15±0.43
	715	93.30±0.20	**13.72±5.11**	**79.57±5.26**	61.03±0.24	0.74±0.18	60.29±0.13
ResNet50	107*	**95.6±0.076**	**77.5±8.89**	**18.1±8.91**	Training limited by memory constrains		
	213	94.748±0.34	51.796±25.36	42.95±25.66	**63.526±0.78**	1.818±0.24	**61.708±0.81**
	318	94.85±0.12	53.54±11.7	41.3111.68	63.086±0.97	2.296±0.6	60.79±1.30
	423	94.75±0.25	54.812±23.2	39.93±23.12	63.086±1.00	1.812±0.29	61.274±1.02
	1415	94.78±0.15	44.21 ±27.72	50.57 ± 27,67	63.16±0.49	**2.33±0.84**	60.82± 0.56
	1920	94.75±0.089	57.95±24.96	36.80±24.92	63.23±0.49	2.208±0.51	61.02±0.64
	3100	94.68±0.28	55.68±14.21	39±14.21	62.83±0.64	2.244±0.77	60.59±1.01

Table A.8: **Receptive Field experiments using dilation:** Contrary to results in Table 4.1, increasing the receptive field using dilation while maintaining the feature map size constant does not seem to have a noticeable effect, either in dense or pruned accuracy. The receptive field denoted with * is the original receptive field of the model without alterations.

A.6 Reproducibility

Here is a complete list of the packages in the conda environment used in all the experiments.

- **name:** work
- **channels:**
  - pytorch
  - defaults
  - conda-forge
- **dependencies:**
  - _libgcc_mutex=0.1=main
  - bat=0.17.1=h48fdc8c_0

- blas=1.0=mkl
- bottleneck=1.3.4=py39hce1f21e_0
- brotli=1.0.9=he6710b0_2
- bzip2=1.0.8=h7b6447c_0
- ca-certificates=2025.4.26=hbd8a1cb_0
- certifi=2025.1.31=pyhd8ed1ab_0
- cudatoolkit=10.2.89=hfd86e86_1
- cudnn=7.6.5=cuda10.2_0
- cycler=0.10.0=py39h06a4308_0
- dbus=1.13.18=hb2f20db_0
- expat=2.4.1=h2531618_2
- ffmpeg=4.3=hf484d3e_0
- fontconfig=2.13.1=h6c09931_0
- fonttools=4.25.0=pyhd3eb1b0_0
- freetype=2.10.4=h5ab3b9f_0
- giflib=5.2.1=h7b6447c_0
- glib=2.69.1=h5202010_0
- gmp=6.2.1=h2531618_2
- gnutls=3.6.15=he1e5248_0
- gst-plugins-base=1.14.0=h8213a91_2
- gstreamer=1.14.0=h28cd5cc_2
- icu=58.2=he6710b0_3
- ignite=0.4.10=py_0
- intel-openmp=2021.3.0=h06a4308_3350
- jpeg=9d=h7f8727e_0
- kiwisolver=1.3.1=py39h2531618_0

- lame=3.100=h7b6447c_0
- lcms2=2.12=h3be6417_0
- ld_impl_linux-64=2.35.1=h7274673_9
- libffi=3.3=he6710b0_2
- libgcc-ng=9.1.0=hdf63c60_0
- libiconv=1.15=h63c8f33_5
- libidn2=2.3.2=h7f8727e_0
- libpng=1.6.37=hbc83047_0
- libstdcxx-ng=9.1.0=hdf63c60_0
- libtasn1=4.16.0=h27cfd23_0
- libtiff=4.2.0=h85742a9_0
- libunistring=0.9.10=h27cfd23_0
- libuuid=1.0.3=h7f8727e_2
- libuv=1.40.0=h7b6447c_0
- libwebp=1.2.0=h89dd481_0
- libwebp-base=1.2.0=h27cfd23_0
- libxcb=1.14=h7b6447c_0
- libxml2=2.9.10=hb55368b_3
- lz4-c=1.9.3=h295c915_1
- matplotlib=3.4.3=py39h06a4308_0
- matplotlib-base=3.4.3=py39hbbc1b5f_0
- mkl=2021.4.0=h06a4308_640
- mkl-service=2.4.0=py39h7f8727e_0
- mkl_fft=1.3.1=py39hd3c417c_0
- mkl_random=1.2.2=py39h51133e4_0
- munkres=1.1.4=py_0

- ncurses=6.2=he6710b0_1
- nettle=3.7.3=hbbd107a_1
- nnn=3.5=h372f924_0
- numexpr=2.8.1=py39h6abb31d_0
- olefile=0.46=pyhd3eb1b0_0
- openh264=2.1.0=hd408876_0
- openssl=1.1.1q=h7f8727e_0
- packaging=21.3=pyhd3eb1b0_0
- pandas=1.4.2=py39h295c915_0
- pcre=8.45=h295c915_0
- pip=21.2.4=py39h06a4308_0
- pyparsing=2.4.7=pyhd3eb1b0_0
- pyqt=5.9.2=py39h2531618_6
- python=3.9.7=h12debd9_1
- python-dateutil=2.8.2=pyhd3eb1b0_0
- pytorch-mutex=1.0=cuda
- pytz=2022.1=py39h06a4308_0
- qt=5.9.7=h5867ecd_1
- readline=8.1=h27cfd23_0
- setuptools=58.0.4=py39h06a4308_0
- sip=4.19.13=py39h2531618_0
- six=1.16.0=pyhd3eb1b0_1
- sqlite=3.36.0=hc218d9a_0
- tk=8.6.11=h1ccaba5_0
- torchaudio=0.10.0=py39_cu102
- tornado=6.1=py39h27cfd23_0

- tzdata=2021a=h5d7bf9c_0
- wheel=0.37.0=pyhd8ed1ab_1
- xz=5.2.5=h7b6447c_0
- zlib=1.2.11=h7b6447c_3
- zstd=1.4.9=haebb681_0
- **pip:**
  - accelerate==0.20.3
  - alembic==1.8.1
  - antlr4-python3-runtime==4.9.3
  - appdirs==1.4.4
  - attrs==22.1.0
  - autopage==0.5.1
  - charset-normalizer==3.3.2
  - click==8.1.7
  - cliff==4.0.0
  - cmaes==0.11.1
  - cmd2==2.4.2
  - colorlog==6.7.0
  - delve==0.1.50
  - docker-pycreds==0.4.0
  - filelock==3.14.0
  - fsspec==2024.5.0
  - gitdb==4.0.11
  - gitpython==3.1.43
  - greenlet==1.1.3.post0
  - huggingface-hub==0.23.0
  - hydra-core==1.3.1
  - idna==3.7

* imageio==2.36.0
* importlib-metadata==5.0.0
* jinja2==3.1.4
* joblib==1.2.0
* lazy-loader==0.4
* lightning-utilities==0.9.0
* mako==1.2.3
* markupsafe==2.1.1
* mpmath==1.3.0
* networkx==3.2.1
* npy-append-array==0.9.16
* numpy==1.23.0
* nvidia-cublas-cu12==12.1.3.1
* nvidia-cuda-cupti-cu12==12.1.105
* nvidia-cuda-nvrtc-cu12==12.1.105
* nvidia-cuda-runtime-cu12==12.1.105
* nvidia-cudnn-cu12==8.9.2.26
* nvidia-cufft-cu12==11.0.2.54
* nvidia-curand-cu12==10.3.2.106
* nvidia-cusolver-cu12==11.4.5.107
* nvidia-cusparse-cu12==12.1.0.106
* nvidia-nccl-cu12==2.20.5
* nvidia-nvjitlink-cu12==12.4.127
* nvidia-nvtx-cu12==12.1.105
* omegaconf==2.3.0
* optuna==3.3.0
* paretoset==1.2.3
* pbr==5.10.0
* pillow==11.0.0

* prettytable==3.4.1
* protobuf==4.25.3
* psutil==5.9.5
* pyhessian==0.1
* pyperclip==1.8.2
* pyyaml==6.0
* requests==2.31.0
* safetensors==0.4.3
* scikit-image==0.24.0
* scikit-learn==1.2.1
* scipy==1.13.1
* seaborn==0.12.2
* sentry-sdk==2.0.1
* setproctitle==1.3.3
* smmap==5.0.1
* sqlalchemy==1.4.41
* stevedore==4.0.0
* sympy==1.12
* tensorboardx==2.6.2.2
* thop==0.1.1-2209072238
* threadpoolctl==3.1.0
* tifffile==2024.8.30
* timm==1.0.3
* torch==2.3.0
* torch-summary==1.4.5
* torchconvquality==0.3.0
* torchmetrics==1.6.1
* torchvision==0.18.0
* tqdm==4.64.1

    - triton==2.3.0
    - typing-extensions==4.11.0
    - urllib3==2.2.1
    - wandb==0.16.6
    - wcwidth==0.2.5
    - zipp==3.9.0

- **prefix:** /users/sclaam/.conda/envs/work

Appendix B

Training Dynamics, Receptive Field, and Pruning Supplementary material

B.1 Weight Distribution

Figure B.1 shows that the learning rate has a large impact on the weight distribution, which is logical, since a higher pruning rate results in a larger magnitude update. This, combined with the weight decay parameter (which is constant across all models), results in models with a narrower weight distribution than those trained with a smaller learning rate. It is also important to note that, for both learning rates, EKFAC still has a wider distribution than the other two optimisers, and it is also the optimiser that yields the worst prunability. Also, the models that were more prunable (i.e., maintained accuracy after pruning) had a 95^{th} percentile of weight magnitude smaller than 0.01 even across. The average pruned accuracy for models with a 95^{th} percentile of weights smaller than 0.01 is 76.63%. In contrast, for models where this condition is not met, the average pruned accuracy is 18.27%, irrespective of the optimiser, learning rate, or receptive field. It is important to note that differences in the magnitude of the weight distribution do not affect the dense accuracy in the same way, underscoring the importance of this for the pruned regime.

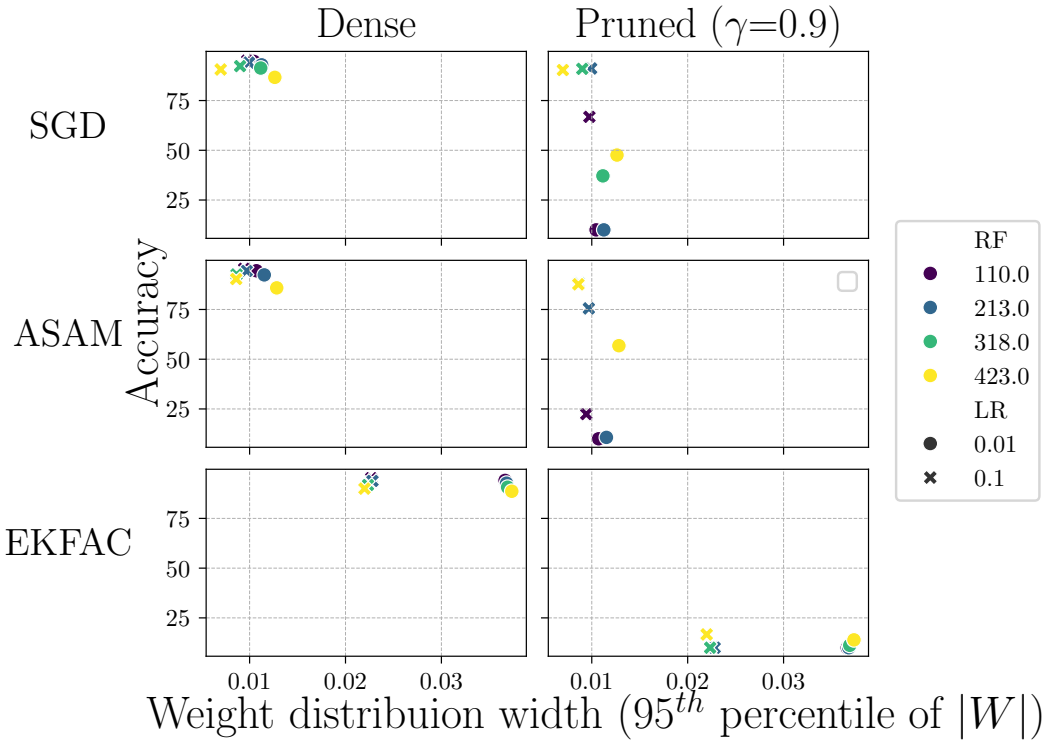


Figure B.1: **Weight distribution vs Accuracy of ResNet50 trained on CIFAR10:** This figure shows the dense and pruned accuracy of ResNet50 for different optimisers, receptive fields and learning rates. The x-axis is the value that corresponds to the 95^{th} percentile of the distribution of the absolute value of the weights of the model. The y-axis shows the accuracy of each model. It can be seen that weight distribution plays a substantial role in prunability: models trained with EKFAC achieve similar dense accuracy to models trained with SAM and SGD, but once pruned, their accuracy collapses.