POLITECNICO DI TORINO

2nd LEVEL's Degree in ICT FOR SMART SOCIETIES

2nd LEVEL's Degree Thesis

Exploring Relevance-Based Pruning Strategies in VGG Models: A Comparative Study

Supervisors
Prof. Carla FABIANA CHIASSERINI

Candidate
Zhiqiang ZHAO

2024

Abstract

This thesis provides a comprehensive analysis of various pruning strategies applied to the VGG16 neural network model, with a focus on identifying methods that maintain model performance post-pruning. Utilizing the CIFAR-10 dataset, the study evaluates the efficacy of pruning based on relevance scores derived from the TorchLRP repository, compared against pruning based on the absolute mean values of the model's parameters.

The investigation extends to a novel Delta-Varied Combined Pruning Strategy, which interpolates between parameter-focused and relevance-focused pruning to explore a spectrum of hybrid strategies. This strategy was examined for its ability to preserve the accuracy and minimize the loss of the VGG16 model across a wide range of pruning percentages.

Experimental results reveal that while individual relevance-based and parameter-based strategies offer certain benefits, a combined approach that alternates the ranking of channel importance according to both relevance and parameters demonstrates superior performance. Notably, the Delta-Varied Combined Pruning Strategy elucidates an optimal balance between the two pruning philosophies, with certain delta settings outperforming others in maintaining robust model accuracy at extensive pruning levels.

The findings of this study underscore the importance of considering a multifaceted approach to pruning, which integrates multiple metrics to achieve an efficient and effective reduction in model complexity. The insights gained from this research contribute significantly to the field of neural network optimization, offering a pathway to more resource-efficient deep learning models without substantial accuracy trade-offs.

Acknowledgment

First and foremost, I extend my deepest gratitude to Prof. Carla Chiasserini and Dr. Francesco Malandrino for their invaluable guidance, insightful advice, and unwavering support throughout this research. Their expertise and mentorship have been instrumental in shaping not only this thesis but also my approach to scientific inquiry and problem-solving. I am truly grateful for the opportunity to learn from and work alongside such distinguished scholars.

I would also like to express my sincere appreciation to Politecnico di Torino for providing me with the opportunity to pursue my studies in an environment that champions excellence, innovation, and learning. The resources, facilities, and academic community here have played a crucial role in my personal and professional development.

To my friends, who have been my constant source of support and encouragement, I offer my heartfelt thanks. Your companionship and belief in my abilities have been a source of strength and motivation during this journey. I am fortunate to have shared this experience with such a remarkable group of individuals.

In closing, I acknowledge that this thesis is not just a reflection of my efforts but the culmination of numerous contributions and support from my supervisors, the university, and my dear friends. To everyone who has been a part of this journey, thank you.

Contents

Chapter 1

Introduction

1.1 The Rise of Deep Learning and CNNs

The advent of deep learning has marked a revolutionary shift in the field of artificial intelligence (AI), transforming the landscape of technology and research. At the heart of this transformation lies the development of Convolutional Neural Networks (CNNs), a class of deep neural networks that have proven especially effective in processing data with a grid-like topology, such as images. The origins of deep learning and CNNs can be traced back to the foundational work on artificial neural networks, which sought to mimic the information processing patterns of the human brain [13]. These early networks laid the groundwork for the complex architectures that would later become the backbone of deep learning.

CNNs, in particular, have been instrumental in pushing the boundaries of what's possible in AI. The introduction of LeNet in the 1990s by [12] represented a significant milestone, demonstrating the potential of CNNs in digit recognition tasks. However, it was the success of AlexNet in the 2012 ImageNet competition that truly catalyzed the deep learning boom [10], 2012). AlexNet's architecture, which featured deep layers and innovative techniques such as ReLU and dropout, outperformed traditional machine learning approaches by a significant margin, setting a new standard for image classification tasks.

Following the success of AlexNet, the VGG network, developed by [22] , further advanced the field by introducing a model with 16 convolutional layers. The VGG model's architecture was characterized by its simplicity, relying on a homogeneous design of small convolution filters throughout the network. This approach demonstrated the effectiveness of depth in neural networks, as the VGG model achieved remarkable success in the ImageNet competition and became a popular choice for transfer learning applications.

The progression from early neural networks to the sophisticated architectures of today's CNNs underscores the rapid advancement of deep learning. These developments have not only propelled research in AI but have also led to practical applications that impact our daily lives, from facial recognition systems to autonomous vehicles. The evolution of CNNs, exemplified by models like VGG, highlights the ongoing quest for more efficient and powerful computational models capable of tackling increasingly complex tasks.

1.2 Challenges in Model Efficiency

As the field of deep learning has matured, the complexity and size of neural network models have grown exponentially. This growth, while facilitating unprecedented accuracy in tasks such as image recognition and natural language processing, has introduced significant challenges in model efficiency. The efficiency of a model encompasses not only the computational resources it requires for training and inference but also its memory footprint and energy consumption. These factors become critical when deploying models to resource-constrained environments, such as mobile devices or embedded systems.

One of the primary challenges in model efficiency is the computational cost associated with training and running deep learning models. As models like VGG16 have shown, deeper and more complex architectures can achieve remarkable performance on tasks such as image classification. However, this comes at the cost of increased computational resources, which can limit their applicability in real-world scenarios where such resources are scarce [22]. Furthermore, the energy consumption of these models during training and inference poses environmental concerns, with data centers consuming an estimated 1% of the world's electricity supply, a figure that is rapidly growing [23].

Another significant challenge is the memory requirement of deep learning models. The size of the models often necessitates substantial amounts of RAM for training, which can exceed the capacity of commonly available hardware. This not only impacts the feasibility of training models but also their deployment, as the inference phase also requires the model to reside in memory, limiting deployment in memory-constrained environments [7].

Efforts to address these challenges have led to the development of various strategies, including model compression, quantization, and pruning. Pruning, in particular, has emerged as a promising approach to reduce model size and complexity without significantly compromising performance. By selectively removing weights or neurons that contribute the least to the model's output, practitioners can achieve more efficient models suitable for deployment in resource-constrained settings [7].

1.3 Pruning as a Solution

The exponential increase in the computational power required by modern deep learning models has propelled the search for optimization strategies capable of mitigating these demands. Among various approaches explored, pruning has emerged as a particularly effective method for enhancing model efficiency without substantially sacrificing performance. Pruning, at its core, involves the systematic removal of weights or neurons from a neural network that are deemed less important or redundant, thereby reducing the model's size and complexity [7].

The concept of pruning is grounded in the observation that not all parameters in a neural network contribute equally to its output. In fact, studies have shown that deep neural networks often possess significant redundancy, with large portions of their parameters having minimal impact on the network's performance [8, 11]. This realization has led to the development of various pruning techniques, each aiming to identify and eliminate these redundant parameters while preserving the network's ability to learn and generalize from data.

Early approaches to pruning focused on simplistic criteria for weight removal, such as pruning weights with the smallest magnitudes [7]. However, as the field has evolved, more sophisticated methods have been introduced. These include structured pruning, which targets entire filters or channels for removal, and unstructured pruning, which removes individual weights, leading to a sparse weight matrix [18]. Moreover, recent advancements have explored the use of learning-based approaches to pruning, where the importance of weights is determined through additional training processes, thereby enabling more dynamic and effective reduction of model complexity [16].

The benefits of pruning extend beyond just model size and computational efficiency. Pruned models often exhibit faster inference times and lower energy consumption, making them more suitable for deployment in resource-constrained environments such as mobile devices and embedded systems. Furthermore, by reducing the number of active parameters, pruning can also help in mitigating overfitting, potentially leading to improved model generalizability [7].

1.4 Relevance-Based Pruning Strategies

In the quest for optimizing neural network models, relevance-based pruning strategies have emerged as a sophisticated approach, focusing on the conceptual notion that not all neurons contribute equally to the model's final decision. Unlike traditional pruning methods that may rely on arbitrary thresholds or purely statistical measures of weight importance, relevance-based pruning seeks to evaluate the contribution of each neuron towards the output through more contextually meaningful metrics. This approach leverages the concept of layer-wise relevance propagation (LRP), a technique that attributes the prediction of a network back to its input features, thereby identifying the most significant neurons in terms of contributing to the model's decisions [5].

Relevance-based pruning operates on the premise that neurons exhibiting lower relevance scores are less critical to the model's performance and can thus be pruned with minimal impact on accuracy. By focusing on the pruning of these less relevant neurons, it's possible to substantially reduce the model's complexity while preserving, or even enhancing, its predictive capabilities. This methodology aligns with the broader goals of achieving efficient and compact models that maintain high levels of accuracy, particularly important for deployment in environments with limited computational resources [19].

A pivotal advantage of relevance-based pruning over traditional pruning methods is its ability to make informed decisions based on the semantic importance of each neuron. This ensures that the pruning process is guided by the actual functionality and contribution of neurons to the task at hand, rather than arbitrary metrics. Consequently, relevance-based pruning has demonstrated significant success in various domains, offering a path towards more efficient and interpretable neural network models [25].

Moreover, relevance-based pruning aligns with efforts to enhance model interpretability and trustworthiness, as it inherently provides insights into which aspects of the model are most critical for its decisions. This attribute is especially valuable in applications where understanding the model's decision-making process is crucial, such as in healthcare and autonomous driving [21].

1.5 Research Objectives and Contributions

This thesis undertakes a comprehensive examination of relevance-based pruning strategies applied to VGG16 models, with an emphasis on the CIFAR-10 dataset as a benchmarking tool. Amidst the ongoing developments in neural network pruning, a discernible gap persists in pinpointing strategies that adeptly balance between model complexity and performance, particularly within the domain of relevance-based pruning. This research endeavors to bridge this gap by offering a nuanced comparative analysis of various pruning strategies, assessing their influence on model efficiency and accuracy. The objectives and contributions of this study have been outlined as follows:

- **Evaluate the Baseline Performance:** The research establishes a baseline performance metric for the unaltered VGG16 model on the CIFAR-10 dataset. This benchmark serves as a crucial point of reference for evaluating the effectiveness of subsequent pruning endeavors.
- **Implement Relevance-Based Pruning Strategies:** The study delineates the execution of two distinct pruning strategies, anchored in the relevance scores of model parameters and channels. By employing methodologies from the TorchLRP repository to ascertain relevance scores, these scores are then harnessed to inform and direct the pruning process.
- **Develop a Combined Pruning Approach:** A pivotal methodological advancement of this research is the formulation of an innovative combined pruning strategy.

This strategy synergizes the strengths of the initial relevance-based and parameter-based approaches, aiming to optimize model efficiency while maintaining, if not enhancing, accuracy.

- **Explore Delta-Varied Combined Pruning Strategies:** Extending beyond the initial objectives, this study introduces and assesses the efficacy of Delta-Varied Combined Pruning Strategies. Through a meticulous adjustment of the delta parameter, a spectrum of hybrid strategies is evaluated to discern an optimal balance between relevance and parameter importance, marking a significant extension of the study's scope and insights.

- **Comprehensive Performance Analysis:** The research undertakes a thorough analysis of the impact of different pruning strategies on model performance. By systematically comparing the baseline with pruned models across a range of metrics, the study elucidates the nuanced trade-offs involved in pruning decisions.

Collectively, these efforts culminate in a substantial contribution to the field of neural network optimization, presenting a detailed exploration of relevance-based pruning strategies and their practical implications for model performance. The findings not only extend the existing body of knowledge but also pave the way for future research in efficient neural network design.

1.6 Thesis Organization

This thesis is structured as follows to provide a coherent and comprehensive exploration of relevance-based pruning strategies for VGG16 models:

- **Chapter 1: Introduction** - Presents the motivation, objectives, and key contributions of the study.

- **Chapter 2: Background and Related Work** - Reviews the existing literature on neural network pruning, relevance-based approaches, and the foundational concepts underpinning this research.
- **Chapter 3: Methodology and Algorithms** - Describes the methodologies employed in the study, including the development of the combined relevance and magnitude-based pruning algorithm.
- **Chapter 4: Experimental Setup and Implementation** - Outlines the experimental framework, dataset, and specifics of the VGG16 model adaptation, along with the implementation details of the pruning strategies.
- **Chapter 5: Results and Discussion** - Presents the findings from the experiments, analyses the performance of different pruning strategies, and discusses the implications of these results.
- **Chapter 6: Conclusion** - Summarizes the study's main findings, contributions, limitations, and proposes directions for future research.

Chapter 2

Background and Related Work

This chapter provides an overview of the foundational concepts and prior research relevant to this study. It begins with a discussion on Convolutional Neural Networks (CNNs), focusing on the VGG model, and then delves into the domain of neural network optimization, emphasizing pruning strategies.

2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have become a fundamental framework in the field of deep learning, particularly within the domain of computer vision. Their design, inspired by the biological visual cortex, enables the automatic learning of hierarchically structured features from visual inputs, making them exceptionally proficient at tasks such as image and video recognition, image classification, and medical image analysis.

2.1.1 Foundations of CNNs

At the core of CNNs are several distinct types of layers, each contributing to the network's ability to perceive and interpret complex visual patterns:

- **Convolutional Layers.** : The primary feature extractors of CNNs, these layers apply various filters to the input images to create feature maps that highlight specific attributes like edges or textures. The convolution operation preserves the spatial relationship between pixels by learning image features using small squares of input data.
- **Pooling Layers.** Pooling Layers: Often following convolutional layers, pooling (or subsampling) layers reduce the dimensionality of the data, which decreases the computational load and helps prevent overfitting. Max pooling, one of the most common pooling operations, reduces the size of the feature maps by retaining only the maximum value in a local region of the feature map.
- **Fully Connected Layers.**: Serving as classifiers, these layers take the high-level filtered images from previous layers and translate them into final output values, such as class scores in classification tasks.

The integration of these layers within a CNN allows for the effective capture and analysis of complex patterns in data, leading to high accuracy in various vision-based tasks.

2.1.2 Evolution of CNN Architectures

The evolution of CNN architectures over the years demonstrates significant milestones in the field of deep learning:

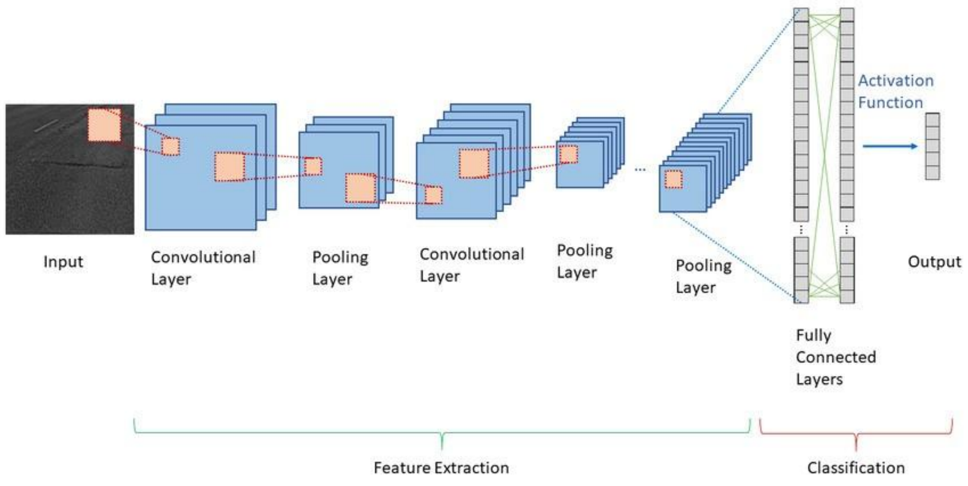

Figure 2.1: The structure of a CNN. It depicts an input image passing through multiple convolutional layers with activation functions, interspersed with pooling layers, and culminating in fully connected layers that lead to the output classification.[3]

- **LeNet (1998)**: Pioneered by Yann LeCun and his colleagues, LeNet was among the first CNNs and successfully applied to digit recognition tasks [12].

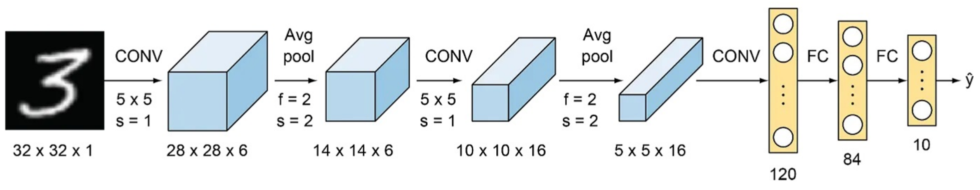

Figure 2.2: Lenet-5 Architecture.[2]

- **AlexNet (2012)**: Marking a breakthrough in the application of CNNs, AlexNet significantly outperformed traditional methods in the ImageNet competition, employing deeper layers and novel techniques like ReLU and dropout for the first time [10].

- **VGG (2014)**: The VGG architecture, particularly the VGG16 model, is celebrated for its simplicity and depth. Utilizing small (3x3) convolution filters across its layers, VGG16 demonstrated the effectiveness of depth in neural networks for improving accuracy on complex image recognition tasks [22].

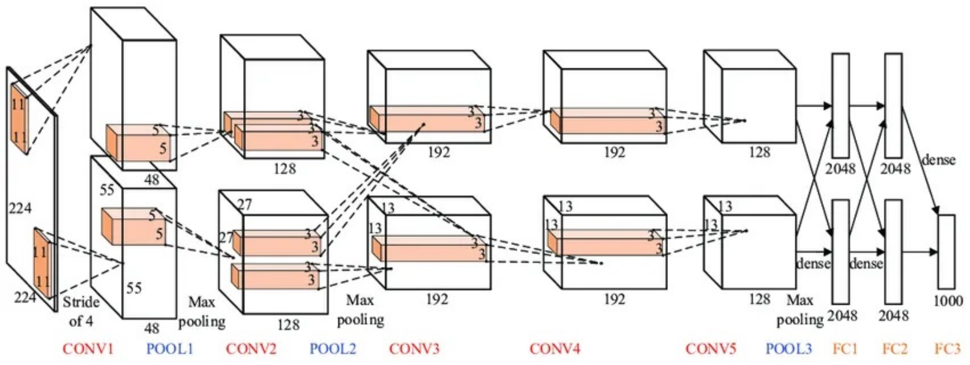

Figure 2.3: AlexNet Architecture.[2]

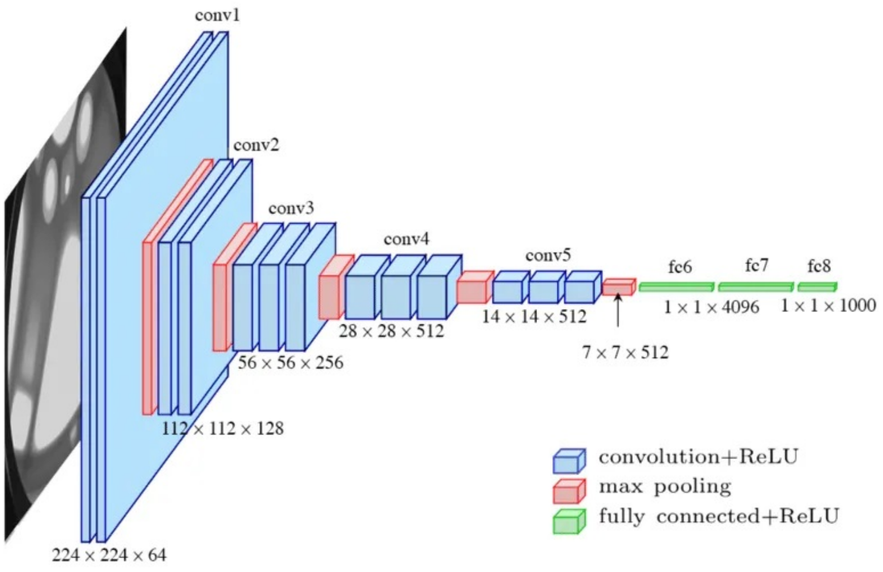

Figure 2.4: VGG-16 Architecture.[2]

These developments underscore the progression towards more complex and capable models, setting the foundation for the sophisticated architectures that power today's AI applications.

2.2 The VGG Model

Developed by Karen Simonyan and Andrew Zisserman in 2014, the VGG (Visual Geometry Group) model, particularly its VGG16 variant, stands as a landmark in the evolution of Convolutional Neural Networks for deep learning. Its architecture's simplicity, coupled with its depth, has set a new benchmark for image recognition tasks, showcasing the profound impact of network depth on accuracy and performance.

2.2.1 Architecture Overview

The VGG16 model is characterized by its uniform architecture, consisting of 16 layers that are directly involved in the learning process, including thirteen convolutional layers and three fully connected layers[22]. The model employs small (3x3) convolution filters throughout, which, despite their size, enable it to capture complex patterns by increasing the network's depth. Pooling layers are strategically placed to reduce dimensionality and computational load, allowing for an efficient learning process.

- **Convolutional Layers**: VGG16 utilizes multiple stacked convolutional layers with small receptive fields to extract features. The use of small filters across all layers is a key innovation, providing the model with fine-grained feature extraction capabilities.
- **Fully Connected Layers**: Following the convolutional layers, three fully connected layers culminate the network's architecture. The first two have 4096 channels each, while the third performs the final classification into 1000 classes (for the ImageNet challenge) and has 1000 channels.

The flattened architecture of VGG-16 is as shown below:

2.2.2 Significance in Deep Learning

VGG16's design philosophy, emphasizing depth and uniformity, has had a lasting impact on the design of subsequent neural network architectures. Its performance in the 2014

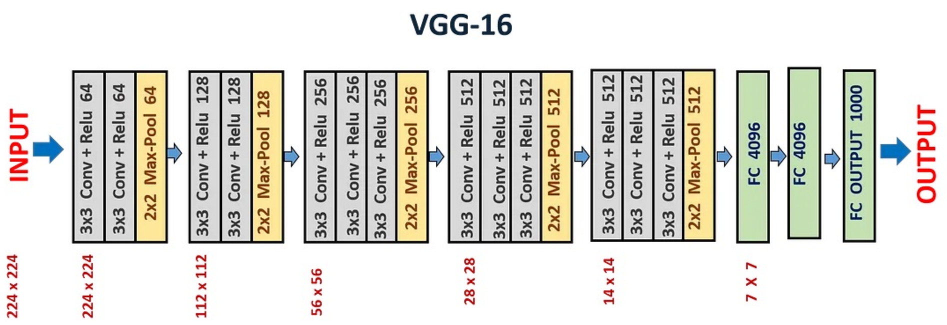

Figure 2.5: flattened architecture of VGG-16.

ImageNet competition underlined the importance of network depth for achieving high accuracy in image recognition tasks. Moreover, VGG16 has become a popular choice as a feature extractor for various applications beyond classification, including object detection and image segmentation, due to its versatile and powerful representation of visual features.

The model's architecture has also facilitated the development of transfer learning, where VGG16, pre-trained on a large dataset like ImageNet, is used as a starting point for training on other visual recognition tasks. This approach leverages the generic features learned by VGG16, demonstrating its adaptability and efficiency across different domains.

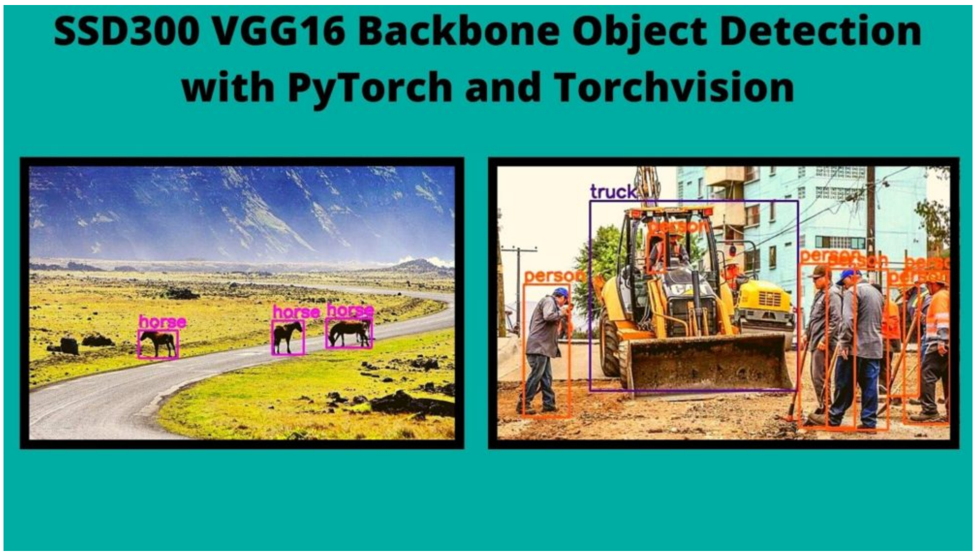
SSD300 VGG16 Backbone Object Detection with PyTorch and Torchvision
horse
horse
horse
truck
person
person
person

Figure 2.6: Object detection technique based on VGG16 model.[1]

VGG16's introduction marked a pivotal moment in deep learning, proving the efficacy of deep, uniform architectures in achieving state-of-the-art performance on complex visual recognition tasks[22]. Its legacy extends beyond its immediate successes, influencing the development of future neural network designs and applications across a wide range of industries.

2.3 Neural Network Optimization

The surge in neural network applications has been accompanied by a dramatic increase in model size and complexity. While larger models often achieve superior performance on tasks such as image classification and natural language processing, their high computational demands pose significant challenges. Neural network optimization aims to address these challenges, focusing on enhancing model efficiency without compromising the accuracy of predictions.

2.3.1 Need for Optimization

The primary challenges in neural network optimization stem from the computational cost, memory requirements, and energy consumption associated with training and deploying large models. As models grow in size, they require more computational resources, which can be prohibitive for real-time applications or devices with limited processing power. Moreover, the environmental impact of training large-scale neural networks has become an increasingly important consideration, with data centers consuming a substantial and growing share of global electricity [23].

2.3.2 Strategies for Optimization

Optimization strategies encompass a wide range of techniques designed to reduce the computational burden of neural networks while maintaining, or even improving, their performance. These strategies include, but are not limited to, the following:

- **Model Compression**: Techniques such as weight pruning and quantization that reduce the model size and complexity, making them more amenable to deployment on devices with limited resources.
- **Efficient Architectures**: Designing new neural network architectures that achieve high accuracy with fewer parameters or operations, such as MobileNets and EfficientNets.
- **Knowledge Distillation**: A method where a smaller, more efficient "student" model is trained to replicate the performance of a larger "teacher" model, capturing its predictive power in a more compact form.

  These strategies aim to strike a balance between model complexity and computational efficiency, enabling the deployment of advanced AI applications in a wider range of environments.

2.3.3 The Role of Pruning

Pruning, particularly, has emerged as a key technique in neural network optimization. By identifying and removing neurons or connections that contribute minimally to the output, pruning can significantly reduce the model's size and computational needs without a substantial loss in accuracy. This approach not only enhances model efficiency but also can lead to improvements in generalization by reducing overfitting [7].

[4]

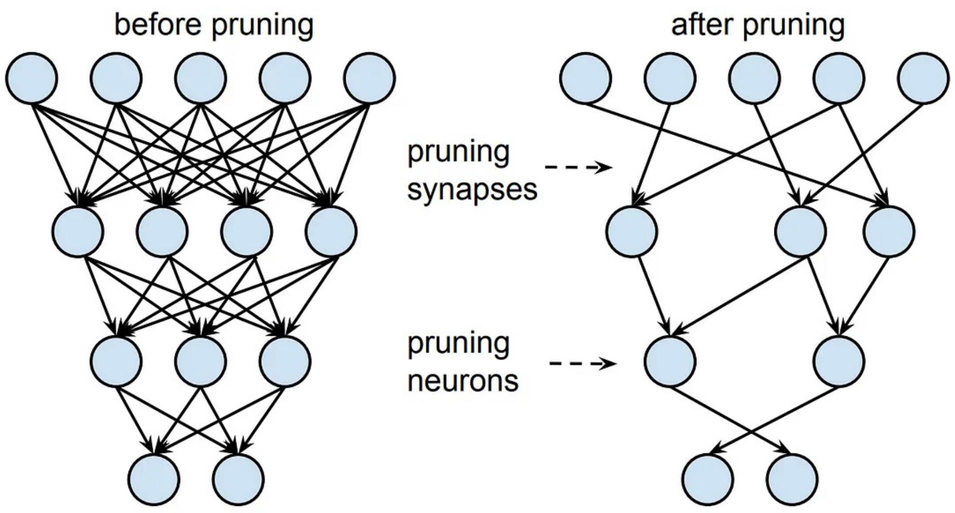

Figure 2.7: Visualization of pruning weights/synapses vs nodes/neurons.[4]

Optimizing neural networks for efficiency is a multifaceted challenge that encompasses various strategies, each contributing to the goal of making AI more accessible and sustainable. Pruning stands out as a particularly effective approach, offering a practical path to enhancing the performance and usability of neural networks across a diverse array of applications.

2.4 Pruning Strategies

As neural networks grow in complexity and depth to achieve higher accuracy, the computational cost and resource requirements for training and inference also increase significantly. Pruning has emerged as a critical strategy for optimizing neural networks by reducing their size and complexity without substantially affecting their performance. This section delves into the different approaches to pruning, highlighting their methodologies, advantages, and applications.

2.4.1 Unstructured vs. Structured Pruning

Pruning can be broadly categorized into two types: unstructured and structured.

- **Unstructured Pruning** focuses on removing individual weights or connections within the network based on certain criteria, such as the magnitude of weights. This approach leads to sparse weight matrices and can significantly reduce the model's size. However, the resulting sparsity often requires specialized hardware or software to fully realize the computational efficiency benefits during inference [7].
- **Structured Pruning** takes a more holistic approach by removing entire neurons or channels, leading to a reduction in the number of filters in convolutional layers, for instance. This type of pruning is more amenable to conventional hardware acceleration techniques and can simplify the network architecture without introducing sparsity.However, structured pruning suffers from consider-able accuracy loss due to its coarse-grained pruning feature. The challenge lies in determining which neurons or channels to prune while minimizing the impact on the network's accuracy [14].

2.4.2 Relevance-Based Pruning

Beyond these broad categories, relevance-based pruning represents a sophisticated approach that considers the importance or 'relevance' of neurons to the network's output when deciding which ones to prune. This method aims to retain the network's ability to perform its task with minimal loss in accuracy by preserving neurons that contribute most significantly to the output.

- **Layer-wise Relevance Propagation (LRP)** is one technique used in relevance-based pruning to attribute the prediction of the network back to its input features,

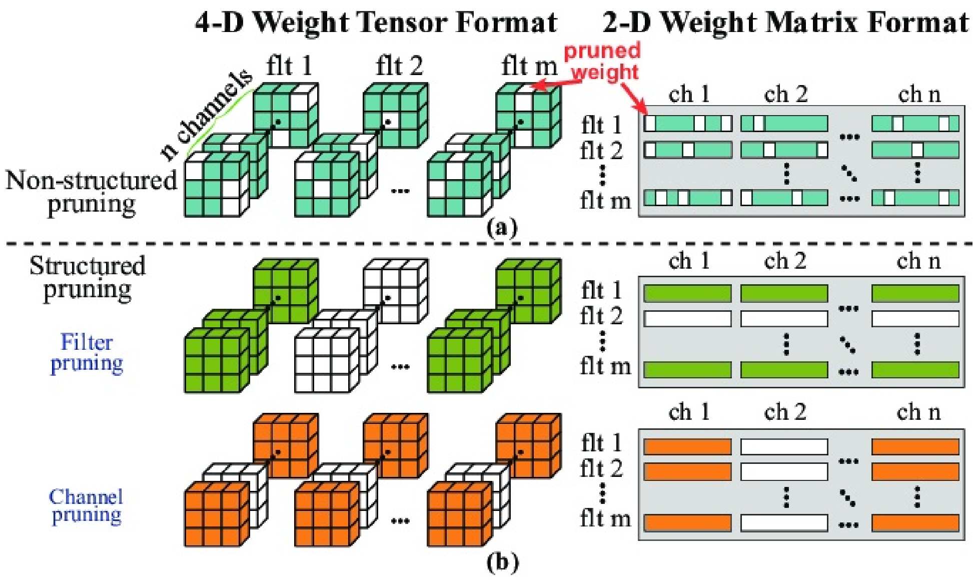

Figure 2.8: llustration of (a) unstructured pruning and (b)coarse-grained structured pruning.[6]

thereby identifying the most critical neurons in terms of contributing to the model's decisions [5]. The Layer-wise Relevance Propagation is shown in 2.9.

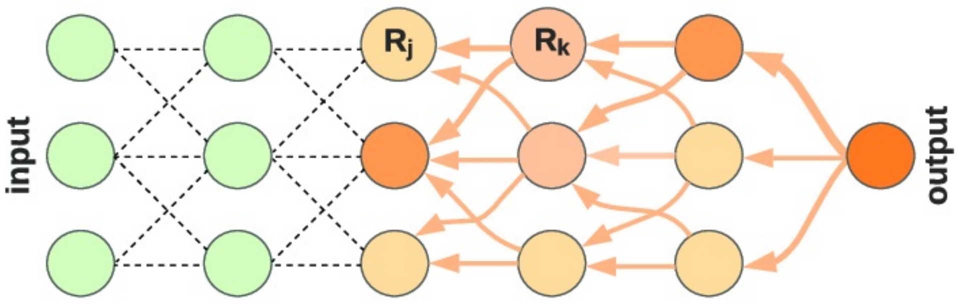

Figure 2.9: A visual depiction of Layer-wise Relevance Propagation.Relevance scores (Rj , Rk) are calculated backwards from the output foreach layer (j and k represent neurons). Scores from each previous layerare used to score the next set of neurons with the final outcome beingthe importance of each input.[20]

Relevance-based pruning, especially when guided by LRP, offers a path toward not only more efficient but also more interpretable neural networks. By focusing on the preservation of semantically important neurons, this approach aligns with the broader goals of developing AI systems that are both high-performing and understandable.

2.5 Related Work

In the exploration of neural network optimization, pruning strategies, particularly those based on relevance, have garnered significant attention for their potential to reduce model complexity while preserving, or even enhancing, performance. This section reviews seminal and recent studies that have advanced the understanding and application of pruning techniques, with a focus on relevance-based approaches and their application to VGG models.

2.5.1 Studies on Pruning VGG Models

Pruning has been extensively applied to VGG models, given their popularity and high performance in various tasks. One notable study by Molchanov et al. (2019) introduced a criterion based on Taylor expansion for pruning filters in convolutional layers, demonstrating significant reductions in model size with minimal loss in accuracy on VGG and other architectures. This approach underscores the potential for structured pruning to streamline VGG models efficiently [19].

Another relevant study by Li et al. (2016) focused on pruning filters from CNNs, including VGG16, to decrease redundancy without compromising accuracy. Their methodology provides insights into the importance of layer-wise pruning and its implications for maintaining performance in deep networks [14].

One notable study proposed an Incremental Pruning based on Less Training (IPLT) method3. IPLT achieved 8x-9x compression for VGG-19 on CIFAR-10 and only needed to pre-train a few epochs. This study not only achieved test acceleration but also training acceleration, which is a novel contribution in the field[15].

These studies highlight the effectiveness of pruning in optimizing VGG models, shedding light on strategies that selectively reduce network complexity while retaining their capability to perform high-level tasks.

2.5.2 Advancements in Relevance-Based Pruning

Relevance-based pruning represents a more nuanced approach by considering the contribution of each neuron to the output. This method has seen innovative applications, notably in the work of Yeom et al. (2021), who proposed a novel criterion for deep neural network pruning that leverages explanations generated by LRP. Their method, applied to networks including VGG, underscores the potential of interpretability-driven pruning to not only enhance model efficiency but also provide insights into the model's decision-making process [25].

In addition, advancements in automated pruning frameworks that incorporate relevance metrics have started to emerge. Liu et al. (2021) explored the integration of automated techniques with relevance-based criteria, offering a pathway to more adaptive and efficient pruning methods that maintain a high level of model accuracy [17].

2.5.3 Gaps and Future Directions

While these studies have significantly advanced pruning methodologies, gaps remain in fully understanding the balance between efficiency and performance, particularly for complex models like VGG applied to diverse tasks. Future research could further explore the integration of relevance-based pruning with other optimization techniques, such as quantization and knowledge distillation, to compound efficiency gains. Additionally, the development of more granular relevance metrics that can guide pruning at the individual neuron level may offer new avenues for optimization.

Chapter 3

Methodology and Algorithms

This chapter outlines the methodology adopted in this study. It details the experimental design, from the choice of the CIFAR-10 dataset and VGG16 model preparation to the implementation of various pruning strategies and their evaluation. Central to our approach is the investigation of how relevance-based and parameter-based pruning strategies, alongside a novel combined method, impact the model's performance. This systematic examination aims to offer insights into optimizing convolutional neural networks (CNNs) for enhanced efficiency and accuracy.

3.1 Dataset

3.1.1 CIFAR-10 Dataset Overview

The CIFAR-10 dataset is fundamental to this research, comprising 60,000 32x32 pixel color images across 10 classes, with a balanced distribution of 6,000 images per class. This dataset is evenly divided into 50,000 training images and 10,000 test images, facilitating a structured approach to training and evaluating the neural network model. The CIFAR-10 dataset's design and utility in machine learning benchmarks have been well-documented and widely recognized within the research community [9].

3.1.2 Justification for Selection

CIFAR-10's adoption as a benchmark in numerous machine learning and computer vision studies is attributed to its balanced class distribution and the representational diversity of its images. Its widespread acceptance and use make it an ideal choice for assessing the efficacy of various pruning strategies on neural network performance, ensuring that the findings of this study are relevant and comparable within the broader research landscape. The dataset's moderate scale strikes a balance between computational manageability and the complexity required to test the pruning strategies' impact effectively.

3.1.3 Relevance to Pruning Strategies

The structured composition and varied content of the CIFAR-10 dataset make it particularly suited for examining the implications of pruning on model accuracy and efficiency. The diversity of image types within the dataset challenges the pruned models to maintain high performance across a broad range of visual features, providing a rigorous testbed for the comparative analysis of pruning strategies. Through iterative experimentation across different pruning percentages, this study aims to elucidate the trade-offs between model simplification and the retention of predictive accuracy.

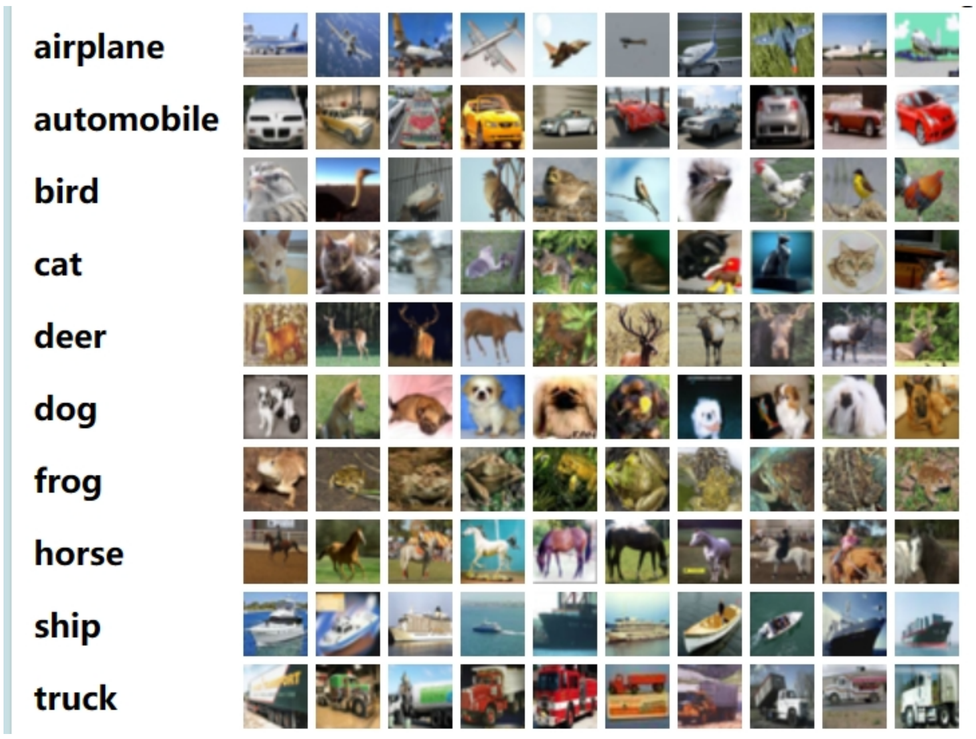

Figure 3.1: The 10 classes in the dataset, as well as 10 random images from each

3.2 VGG16 Model

3.2.1 Overview of the VGG16 Architecture

The VGG16 model, developed by Simonyan and Zisserman, is a convolutional neural network renowned for its depth and simplicity. Comprising 16 layers that directly contribute to learning — 13 convolutional layers and 3 fully connected layers — the architecture utilizes small (3x3) convolution filters throughout, which has been a significant factor in its success for various image classification tasks. The use of such filters allows for the capture of fine details and textures in the input images, contributing to the model's robustness and high performance on complex datasets like ImageNet. The architecture's effectiveness and efficiency have made it a staple in the field of deep learning, often serving as a baseline for comparison and a foundation for further research and development [22].

3.2.2 Model Adaptation for CIFAR-10

For the purposes of this study, the VGG16 model was adapted to suit the CIFAR-10 dataset. This involved modifying the final layers of the network to output 10 classes corresponding to the CIFAR-10 categories. Specifically, the original classification layer designed for 1000 ImageNet classes was replaced with a new fully connected layer tailored to the 10-class problem presented by CIFAR-10. This adaptation ensures that the model's architecture is directly applicable to the task at hand, allowing for an accurate assessment of pruning strategies' effectiveness on a more compact, yet challenging, dataset.

To tailor the VGG16 model for the CIFAR-10 dataset, significant modifications were made to its classifier to accommodate the dataset's 10 output classes. The adaptation process involved the following steps:

- **1. Extracting the Classifier's Submodules**: Initially, the classifier's submodules were retrieved, forming the basis for modification.
- **2. Extension of the Classifier**: To the existing classifier, a series of layers were appended to refine the model's output for CIFAR-10's classification requirements:
  - A ReLU activation layer was added to introduce non-linearity, enhancing the model's ability to learn complex patterns.
  - A new Linear layer with 1000 input units (matching the VGG16's last layer size) and 500 output units was incorporated to begin the process of tapering down to the desired 10 classes.
  - Another ReLU activation layer followed, ensuring the model's continued capacity for non-linear learning.
  - A subsequent Linear layer further reduced the dimensionality from 500 units to the final 10 units, corresponding to the CIFAR-10 classes.

  - Finally, a Dropout layer with a probability of 0.25 was added to mitigate overfitting by randomly omitting a subset of features during training.

- **3. Reassembling the Classifier**: The augmented list of submodules was then converted back into an nn.Sequential object and reassigned to the VGG16's classifier attribute. This reconstructed classifier reflects the specific requirements of the CIFAR-10 dataset, enabling the model to effectively learn and predict across its ten distinct classes.

  Table A.1 provided in Appendix outlines the adapted architecture of the VGG16 model for the CIFAR-10 dataset. The modifications, including additional layers to the classifier, are specified to illustrate the model's final configuration used in this study.

3.2.3 Data Preparation and Transformation

Training Data Transformation and Preparation

To adequately prepare the CIFAR-10 dataset for training the adapted VGG16 model, a series of transformations were applied to augment the training data and normalize both training and testing sets. The transformation pipeline for the training data included:

- **Random Horizontal Flipping**: To introduce variability and reduce overfitting by randomly flipping images horizontally.
- **Random Rotation**: Images were randomly rotated by up to 10 degrees to further augment the training data and improve the model's generalizability.
- **Normalization**: The images were normalized using mean and standard deviation values of (0.5, 0.5, 0.5) across the three color channels (RGB), standardizing the data for optimal training conditions.

The training data were then loaded from the CIFAR-10 dataset, with the specified transformations applied:

```
train_transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),  # randomly flip and rotate
    transforms.RandomRotation(10),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
train_data = datasets.CIFAR10('data', train=True,
                              download=True, transform=train_transform)
```

Testing Data Transformation

For the testing dataset, a simpler transformation pipeline was employed, focusing solely on tensor conversion and normalization, without the data augmentation steps used in the training set:

```
test_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])

test_data = datasets.CIFAR10('data', train=False,
                              download=True, transform=test_transform)
```

Data Loader Configuration

The training dataset was divided into training and validation subsets, with 20% of the data reserved for validation. A random seed ensured reproducibility in the split, and SubsetRandomSampler was used to create samplers for both subsets, facilitating the effective evaluation of the model during training. Data loaders were then configured with a batch size of 20 for training, validation, and testing phases, optimizing for computational efficiency and model performance.

3.2.4 Training the Adapted Model

The adapted VGG16 model underwent training on the CIFAR-10 dataset, utilizing a set of predetermined hyperparameters optimized for this specific task. Training involved adjusting the weights of the network over 30 epochs to minimize the loss function, with the goal of achieving high classification accuracy. This process was instrumental in establishing a robust baseline performance for the unpruned VGG16 model, against which the effectiveness of subsequent pruning strategies could be measured.

To leverage the powerful feature extraction capabilities of the VGG16 model pretrained on ImageNet, we initiated our model with:

```python
import torchvision.models as models

vgg = models.vgg16(pretrained=True)
vgg.eval()  # Set the pre-trained layers to evaluation mode
```

Listing 3.1: Initializing the pre-trained VGG16 model.

This approach ensures that the pre-existing weights of the VGG16 model, trained on a vast and diverse dataset, are preserved. The model's existing layers were kept in evaluation mode to maintain their learned features, which are robust and generalizable across various image recognition tasks.

Subsequently, to tailor the model for the CIFAR-10 dataset, new layers were appended to the classifier portion of the VGG16 model. This adaptation aimed to fine-tune the model specifically for the task of classifying the ten distinct classes present in CIFAR-10:

```python
from torch import nn

# Code snippet showing the addition of new layers as described earlier
features = list(vgg.classifier.children())
features.extend([
    nn.ReLU(),
    nn.Linear(1000, 500),
    nn.ReLU(),
    nn.Linear(500, 10),
    nn.Dropout(0.25)
])
vgg.classifier = nn.Sequential(*features)
```

Listing 3.2: Modifying the VGG16 classifier for CIFAR-10.

Only these newly added layers were set to be trainable, allowing the model to learn from the CIFAR-10 dataset while leveraging the pre-trained convolutional features for effective image representation. This hybrid approach optimizes the training process, significantly reducing the time and computational resources required to achieve high classification accuracy on the CIFAR-10 dataset.

The model's performance was evaluated on the test dataset, yielding the following accuracy results by class:

- Airplane: 83% (839/1000)
- Automobile: 90% (904/1000)

- Bird: 66% (660/1000)
- Cat: 66% (663/1000)
- Deer: 79% (799/1000)
- Dog: 69% (698/1000)
- Frog: 85% (851/1000)
- Horse: 84% (847/1000)
- Ship: 91% (916/1000)
- Truck: 89% (899/1000)

Overall, the model achieved an average test accuracy of 80% across all classes, demonstrating the effectiveness of the applied pruning strategies and the robustness of the adapted VGG16 architecture in maintaining high performance on the CIFAR-10 dataset.

3.3 Relevance Calculation Using TorchLRP

3.3.1 Introduction to Layer-wise Relevance Propagation (LRP)

Layer-wise Relevance Propagation (LRP) is a technique designed to offer insights into the decision-making process of neural networks by attributing the prediction of the network back to its input features. This method enables an understanding of which parts of the input image contribute most significantly to the model's classification decision. LRP is particularly useful in enhancing the interpretability of complex models like VGG16, allowing us to identify the relevance of individual neurons or layers to the model's output.

3.3.2 Implementing LRP with TorchLRP

To implement LRP in our study, I utilized the TorchLRP repository, a PyTorch-based implementation designed to facilitate the application of LRP to PyTorch models [24]. The core function for calculating relevance was imported from TorchLRP, enabling the detailed analysis of our adapted VGG16 model's behavior on the CIFAR-10 dataset.

Before computing relevance scores, the pre-trained VGG16 model was first converted into an LRP-compatible format. This conversion ensures the model's layers are prepared for relevance propagation analysis:

```python
import torchvision.models as models
import lrp

vgg = models.vgg16(pretrained=True)
lrp_model = lrp.convert_vgg(vgg).to("cuda")
```

Listing 3.3: Converting VGG16 model for LRP.

3.3.3 Computing Relevance Scores

The core function for relevance score computation, $relevance_compute$, processes input images and their corresponding labels to produce relevance scores for each layer. This function emphasizes the model's evaluation mode and employs backward propagation to calculate the gradients, which are interpreted as relevance scores. The use of CUDA ensures that computations are efficiently performed on GPU, catering to the large-scale data analysis required for this study.

```python
def relevance_compute(model, inputs, labels):
    # Put the data in CUDA
    inputs, labels = inputs.to("cuda"), labels.to("cuda")

    model.eval()
    y_hat = model.forward(inputs, explain=True, rule="alpha2beta1")
    y_hat = y_hat[torch.arange(len(inputs)), y_hat.max(1)[1]]
    y_hat = y_hat.sum()

    # Start to calculate relevance and parameters
    lrp.trace.enable_and_clean()
    y_hat.backward()

    # Collect the relevances
    all_relevances = lrp.trace.collect_and_disable()
    all_relevances = list(reversed(all_relevances))
    for i, t in enumerate(all_relevances):
        all_relevances[i] = torch.mean(t, dim=0)

    return all_relevances
```

Listing 3.4: Function for computing relevance scores using LRP.

3.3.4 Significance of Relevance Calculation

By detailing the implementation and utilization of the TorchLRP library within our methodology, we underscore the computational approach to understanding model interpretability. This process is pivotal, as it directly informs our pruning strategies by identifying the most influential features and layers within the model. Through this analysis, we can make informed decisions on which components of the model are essential for maintaining its predictive accuracy and which can be pruned to optimize the network's efficiency.

3.3.5 Conclusion of Relevance Calculation

This section concludes by emphasizing the significance of the relevance calculation in the context of neural network optimization and interpretability. The use of TorchLRP facilitated a nuanced understanding of model behavior, laying a crucial foundation for the subsequent pruning strategies implemented in the study.

3.4 Pruning Strategies Based on Relevance and Model Parameters

This section elucidates the various pruning strategies utilized in the study, which are grounded in the calculated relevance and parameter statistics of the VGG16 model channels. Pruning is employed to streamline the model by systematically removing less critical channels, aiming to preserve or enhance model accuracy post-pruning.

3.4.1 Relevance-Based Pruning

Relevance-Based Pruning involves pruning channels based on the statistical significance of their relevance scores, with the intent of retaining those channels that contribute most significantly to the model's output. The process unfolds as follows:

- For each channel, we compute statistical measures from its relevance scores—namely, the mean value, absolute mean value, max value, max absolute value, minimum value, and minimum absolute value.
- Channels with the lowest statistical relevance scores are pruned from the model. This approach is predicated on the assumption that channels with lesser relevance scores contribute minimally to the decision-making process of the network.

3.4.2 Parameter-Based Pruning

Conversely, Parameter-Based Pruning serves as the control group for this study and is based on the statistics of the model's parameters (weights). The methodology for this strategy is analogous to that of relevance-based pruning:

- Similar statistical measures are calculated for the parameters of each channel within the convolutional layers.
- Channels with the least 'parameter scores' are pruned. The parameter score could be analogous to the absolute mean value or another statistical measure that represents the importance of parameters in the channel.

3.4.3 Combined Relevance and Magnitude-Based Pruning Algorithm

This section introduces the novel algorithm developed in this study, which synergizes relevance and magnitude-based pruning to optimize neural network efficiency without significantly compromising accuracy. The algorithm leverages insights from relevance theory and magnitude-based methods to create a holistic pruning strategy.

The combined pruning algorithm operates by first calculating the relevance scores of network channels using the TorchLRP framework. Simultaneously, it assesses the magnitude of parameters across the network. These metrics are then harmonized to guide the pruning process, ensuring that channels contributing minimally to model performance, by both relevance and magnitude criteria, are pruned preferentially:

- The combined approach employs a unified score derived from both the relevance statistics and the parameter statistics for each channel.

- This strategy alternates the pruning order based on relevance and parameter scores to create a new pruning list that benefits from the strengths of both methods.

3.4.4 Implementation of Pruning Strategies

The implementation of these pruning strategies involves:

- Application of pruning at various percentages, ranging from 10% to 90%, to understand the effects on model performance comprehensively.
- Assessment of the model's performance post-pruning, examining changes in loss and accuracy.

3.4.5 Rationale for Strategy Selection

The choice of statistical measures for relevance and parameters as criteria for pruning is driven by the need for a methodical reduction in model complexity. The Parameter-Based Pruning strategy serves as a conventional control group to validate the effectiveness of Relevance-Based Pruning. The innovative Combined Pruning Strategy is designed to harness the full potential of both approaches, potentially leading to superior model performance.

Chapter 4

Experimental Setup and Implementation

In this chapter, we transition from the theoretical framework outlined in Chapter 3 to the practical application of the pruning strategies on the VGG16 model. Here, we will detail the experimental procedures undertaken to evaluate the effectiveness of Relevance-Based Pruning, Parameter-Based Pruning, and the Combined Pruning Strategy. Each section is meticulously designed to provide an exhaustive account of the setup, execution, and evaluation of the experiments, ensuring replicability and transparency of the research process.

4.1 Overview of the Experimental Approach

This section provides an overview of the experimental approach, bridging the methodologies detailed in the previous chapter with their practical application. It begins with a recap of the data preparation and model adaptation processes, ensuring a solid foundation for the subsequent experiments.

The chapter then proceeds to describe the implementation of the relevance calculation, detailing how the TorchLRP library was integrated into our model to ascertain the importance of each layer's features. The calculated relevance scores, alongside the parameter statistics, are then leveraged to inform the pruning strategies—ranging from selective channel removal based on least significance to a combined approach that synthesizes both relevance and parameter-based criteria.

A crucial aspect of this chapter is the thorough documentation of the pruning process. We detail how each channel's relevance and parameter scores were analyzed and utilized to iteratively prune the network, followed by a retraining phase to assess the pruned model's performance. This process is carried out across a spectrum of pruning percentages, from conservative to aggressive, enabling us to understand the trade-offs between model size, computational efficiency, and predictive accuracy.

The chapter culminates with a detailed description of the performance evaluation, employing a range of metrics to offer a multifaceted view of the pruning impact. The results collected serve not only to validate the effectiveness of our pruning strategies but also to shed light on the behavior of deep neural networks when subjected to structural simplification.

4.2 Recap of Data Transformation and Model Configuration

4.2.1 Data Transformation

Before applying any pruning strategies, the CIFAR-10 dataset required preparation to ensure optimal conditions for model training and evaluation. The data transformation process included:

- **Augmentation**: To increase the diversity of the training data and prevent overfitting, we implemented random horizontal flipping and random rotations of up to 10 degrees.
- **Standardization**: We converted the images into tensors and normalized them using a mean and standard deviation of (0.5, 0.5, 0.5) for each RGB channel, ensuring consistent input distributions for the model.

These steps, detailed in Chapter 3, facilitated robust model training and provided a foundation for fair comparison post-pruning.

4.2.2 Model Configuration

The pre-trained VGG16 model underwent careful configuration to accommodate the CIFAR-10 dataset:

- **Pre-trained Model Loading**: We initialized the model with weights pre-trained on ImageNet to capitalize on previously learned features, using torchvision.models.vgg16(pretrainec
- **Adaptation for CIFAR-10**: Modifications were made to the classifier section of the VGG16 model to tailor the output for 10 classes, with new layers appended for relevance to the CIFAR-10 classification task.
- **Evaluation Model**: To preserve the integrity of the pre-trained features, the model was set to evaluation mode using vgg.eval() during relevance calculation, ensuring that only the appended layers were updated during the training phase.

4.3 Batch-wise Relevance Computation and Aggregation for Test Dataset

4.3.1 Overview

In transitioning from theoretical methodologies to practical application, this section outlines the approach for computing and managing relevance scores derived from the CIFAR-10 test dataset. Given the limitations posed by available computational resources, it was crucial to devise a strategy that allowed for the efficient processing of data while preserving the integrity and granularity of the relevance scores.

4.3.2 Processing the Test Dataset in Batches

The CIFAR-10 test dataset, comprising a diverse set of images, was selected to evaluate the interpretability of the VGG16 model. To circumvent the constraints of RAM and GPU memory, relevance scores were calculated on a batch-by-batch basis. This method not only ensured the feasibility of computations but also mirrored real-world scenarios where interpretability assessments are performed on individual or subsets of data points.

4.3.3 Initial Dictionary Setup for Storing Relevance Scores

The initial step involved processing a single batch through the relevance computation function to understand the dimensional structure of the relevance tensors. Based on the observed shapes—ranging from 3D tensors for convolutional layers to 1D vectors for fully connected layers—a dictionary was initialized to store relevance scores with keys representing layer and channel indices:

```
x, y = next(iter(test_loader))
```

```python
init_relevance_dict = {}
for j, tensor in enumerate(relevance_compute(lrp_model, x, y)):
    layer = j + 1
    if len(tensor.shape) > 2:
        for k in range(tensor.shape[0]):
            channel = k + 1
            zeros = np.zeros(len(tensor[k].flatten().tolist())).tolist()
            init_relevance_dict[layer, channel] = zeros
    else:
        channel = None
        zeros = np.zeros(len(tensor.flatten().tolist())).tolist()
        init_relevance_dict[layer, channel] = zeros
```

Listing 4.1: Initializing the relevance dictionary.

The dictionary was chosen for its ability to efficiently categorize and store large volumes of data with complex hierarchical relationships, such as those between layers, channels, and their relevance scores. This approach facilitated an organized and flexible aggregation of relevance scores across the entire test dataset.

4.3.4 Overcoming Memory Constraints with an Incremental Update Method

To address the challenge of system crashes due to the large size of accumulated relevance scores, an incremental update method was employed to update a single dictionary incrementally, thus significantly reducing the memory footprint. This approach employed an iterative update formula, effectively calculating the mean relevance values across batches without the need to store all batch data simultaneously:

```python
def Make_relevance_dict(model, relevance_dict, test_loader):
    lrp_model = lrp.convert_vgg(model).to("cuda")
    sample = 0
    for inputs, labels in test_loader:
        sample += 1
        for i, tensor in enumerate(relevance_compute(lrp_model, inputs,
           labels)):
            layer = 1 + i
            if len(tensor.shape) > 2:
                for k in range(tensor.shape[0]):
                    channel = k + 1
                    relevance = tensor[k].flatten().tolist()
                    relevance_dict[layer, channel] = [x + (y - x) /
                        sample for x, y in zip(relevance_dict[layer,
                        channel], relevance)]
            else:
                channel = None
                relevance = tensor.flatten().tolist()
                relevance_dict[layer, channel] = [x + (y - x) / sample
                    for x, y in zip(relevance_dict[layer, channel],
                    relevance)]
    return relevance_dict
```

Listing 4.2: Incremental update of the relevance dictionary.

This method is based on a well-known algorithm for calculating the mean incrementally, which is particularly useful when dealing with large datasets. It can be formalized as follows:

$$\text{new\_mean} = \text{current\_mean} + \frac{(\text{new\_value} - \text{current\_mean})}{\text{sample\_number}}$$

4.4 Pruning Methodology Based on Relevance and Parameter Statistics

4.4.1 Utilizing Relevance Data for Pruning Decisions

The pruning of the model necessitates a strategic approach to determine which channels to remove without significantly impacting model performance. To this end, relevance data stored in the dictionaries is systematically analyzed to ascertain the importance of each channel within the convolutional layers.

4.4.2 Statistical Analysis of Relevance Scores

For each channel, key statistical measures are computed from the relevance data:

- **Mean Value**: The average of the relevance scores, giving a general sense of the channel's overall contribution. Absolute Mean Value: The average of the absolute relevance scores, providing insight into the channel's impact irrespective of the direction of influence.
- **Absolute Mean Value**: The average of the absolute relevance scores, providing insight into the channel's impact irrespective of the direction of influence.
- **Maximum and Minimum Values**: Indicators of the range of the channel's influence.
- **Maximum and Minimum Absolute Values**: Reflecting the extremes of influence a channel has over the model's decisions.

These statistics serve as the basis for determining the channels' importance and their candidacy for pruning. Channels with lower importance metrics, particularly those with smaller absolute mean relevance values, are considered less critical to the model's predictive ability and are selected for pruning.

4.4.3 Application of Pruning Strategies

Three primary lists are generated to guide the pruning process:

- **Relevance-Based Pruning**: Channels are ranked according to their absolute mean relevance scores, with lower scores indicating higher priority for pruning.
- **Parameter-Based Pruning**: Similarly, channels are ranked based on the absolute mean value of the model parameters, allowing for a parallel pruning strategy that is informed by the intrinsic weights of the model.
- **Combined Pruning Strategy**: A synthesized list is created by interleaving the indices from both relevance and parameter-based rankings. This method aims to balance the considerations of both relevance and parameter importance.

```
relevance_indices = [...]
parameter_indices = [...]

mix_indices = []
for i in range(len(relevance_indices)):
    mix_indices.append(relevance_indices[i])
    mix_indices.append(parameter_indices[i])

# Remove duplicates while preserving order
mix_indices = list(dict.fromkeys(mix_indices))
```

Listing 4.3: Creating combined indices for pruning.

To validate the efficacy of these strategies, a control list based on random selection is also generated, ensuring a comprehensive comparison across different pruning approaches.

4.4.4 Experimental Pruning and Performance Evaluation

The model is pruned according to the lists derived from the different strategies, with pruning percentages ranging from 10% to 90%. The impact of each pruning approach on model performance is meticulously recorded, analyzing both loss and accuracy to understand the trade-offs involved.

Following the implementation of different pruning strategies, we observed distinct patterns in model performance as a function of pruning percentage. The following figure illustrates these patterns clearly:

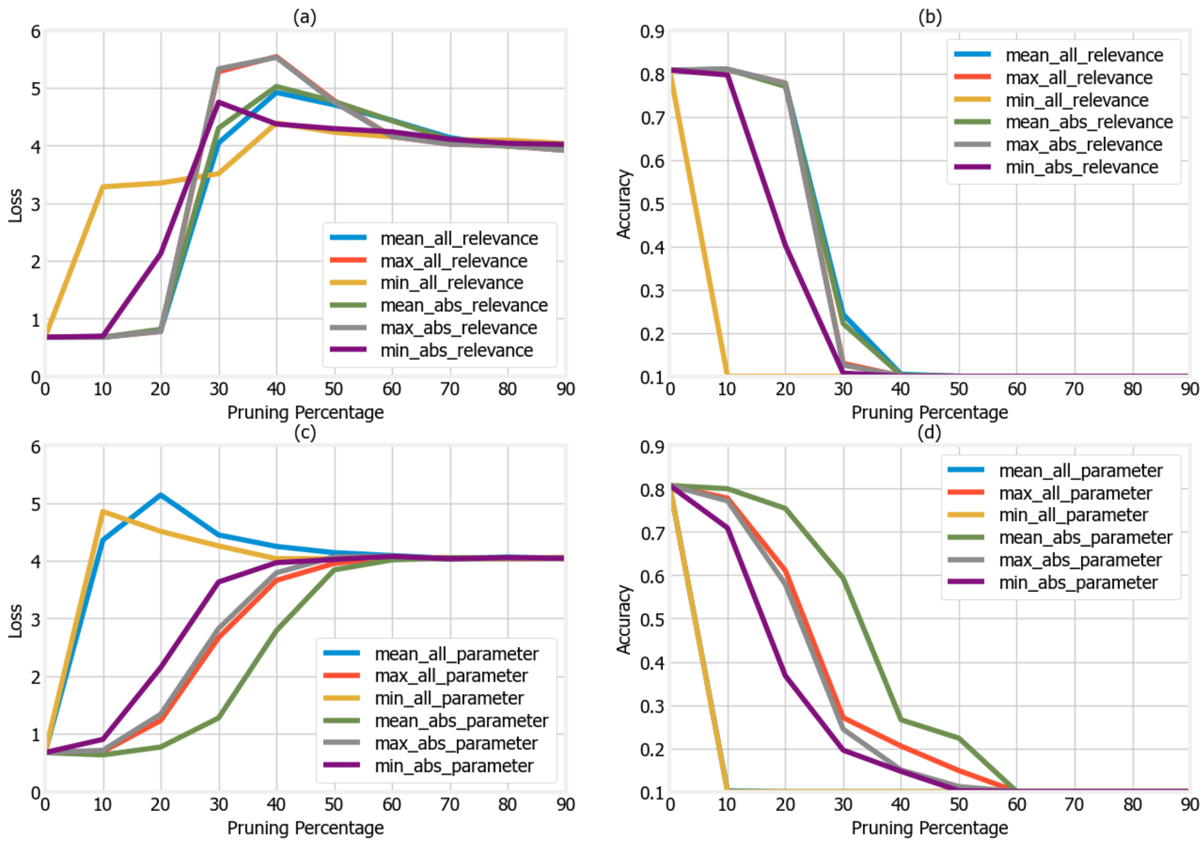

Figure 4.1: Comparison of model loss and accuracy for various statistical measures used in relevance-based pruning strategies.

In Figure 4.1,we present the loss (graphs a and c) and accuracy (graphs b and d) as the pruning percentage increases. Notably, pruning based on the mean absolute value of relevance scores (mean_abs_relevance) demonstrates a more gradual increase in loss and a slower decline in accuracy compared to other measures such as mean, max, and min values. This trend suggests that the mean absolute value serves as a robust indicator of channel importance, effectively balancing the retention of critical information with the removal of redundant or less informative features.

The relative stability of the model performance when pruned using the mean absolute value highlights its potential as the most effective pruning criterion among those tested. It consistently outperforms other relevance-based criteria, particularly at higher pruning percentages, where the choice of pruning metric becomes critically important to maintain model functionality.

These observations informed our decision to adopt the mean absolute value of relevance as the primary criterion for subsequent pruning experiments. This decision is further supported by the lower performance dips exhibited in the absolute max and min values, which, while informative, do not encapsulate the overall significance of the channels as effectively as the mean absolute value.

4.4.5 Delta-Varied Combined Pruning Strategy

Building upon the initial relevance-based, parameter-based, and combined pruning strategies, we introduce a delta-varied combined strategy to dissect the influence of relevance and parameter data on pruning. The delta parameter quantifies the blend between parameter-preference and relevance-preference within the pruning process.

4.4.6 Definition of the Delta-Varied Strategy

The delta-varied strategy is an exploration of the spectrum between purely parameter-based pruning (delta = 0) and purely relevance-based pruning (delta = 1). To operationalize this, we generate mixed lists that vary the proportion of parameter indices to relevance indices. For a delta value of 0.1, the list comprises primarily of parameter indices with a sparse inclusion of relevance indices, reflecting a strong bias towards parameter-based pruning.

4.4.7 Generating Delta-Varied Lists

To facilitate this delta-varied approach, we define a function, 'create_mixed_list', that takes in the lists of indices sorted by parameter importance and relevance scores, along with a delta value. This delta dictates the blend between the two strategies:

```python
def create_mixed_list(parameter_indices, relevance_indices, delta):
    mix_indices = []
    a = b = 0  # Indicators for parameter and relevance inclusion
    i = j = 0  # Indices for traversing parameter_indices and
        relevance_indices

    while i < len(parameter_indices) and j < len(relevance_indices):
        a += (1 - delta)  # Increment a by the parameter proportion
        if a >= 1:
            mix_indices.append(parameter_indices[i])
            i += 1
            a -= 1

        b += delta  # Increment b by the relevance proportion
        if b >= 1:
            mix_indices.append(relevance_indices[j])
            j += 1
            b -= 1

    # Append any remaining elements from both lists
    mix_indices.extend(parameter_indices[i:])
    mix_indices.extend(relevance_indices[j:])
    # Remove duplicates while preserving order
    return mix_indices
```

Listing 4.4: Function to create delta-varied mixed lists.

This function dynamically adjusts the mix of parameter and relevance information in the pruning list according to the *delta* parameter, facilitating a comprehensive examination of their relative contributions.

4.4.8 Applying the Delta-Varied Strategy

Utilizing the `create_mixed_list` function, we generate lists that represent various balances between parameter and relevance influence, dictated by delta values ranging from 0 to 1:

```
delta_varied_lists = {}
for delta in [i * 0.1 for i in range(11)]:
    delta_varied_lists[delta] = create_mixed_list(parameter_indices,
        relevance_indices, delta)
```

Listing 4.5: Generating lists for each delta value.

4.4.9 Evaluation Without Retraining

Contrary to typical pruning approaches that involve retraining, this experiment focused on the immediate impact of pruning on model accuracy and loss. By not retraining the model post-pruning, we aim to ascertain the inherent robustness of the model structure to reductions in complexity and to identify which pruning strategy most effectively balances the trade-off between model size and accuracy.

4.4.10 Insights and Implications

The delta-varied combined pruning strategy sheds light on the nuanced relationship between pruning methodology and model performance. Initial observations suggest that certain balances between parameter and relevance information can achieve significant model simplification while maintaining commendable accuracy. Detailed analysis and discussion of these findings will follow, offering profound insights into optimal pruning practices for efficient model design.

Chapter 5

Results and Discussion

This chapter presents the results of the experiments conducted to evaluate the effectiveness of different pruning strategies on the VGG16 model using the CIFAR-10 dataset. We compare four distinct pruning strategies and explore the Delta-Varied Combined Pruning Strategy across various delta settings.

5.1 Comparison of Four Pruning Strategies

In an effort to understand the impact of various pruning strategies on the VGG16 model trained on the CIFAR-10 dataset, we compared four distinct approaches: pruning based on mean absolute relevance scores, mean absolute parameter scores, a random pruning baseline, and a mixed strategy combining relevance and parameter scores.

The performance of these strategies was evaluated by measuring the model's accuracy and loss at various levels of pruning, ranging from 10% to 90% of the total channels pruned. The key objective was to identify which strategy could prune the network most effectively without significantly compromising accuracy.

5.1.1 Analysis of Pruning on Model Accuracy

Our analysis began by examining the change in accuracy as a function of pruning percentage. Notably, the strategy based on mean absolute relevance scores was hypothesized to perform optimally, under the premise that channels with low relevance scores contribute minimally to the model's decision-making process. Conversely, random pruning served as a control to assess the effectiveness of a non-informed pruning approach.

5.1.2 Analysis of Pruning on Model Loss

Parallel to accuracy, model loss provides insight into the confidence and stability of the network's predictions post-pruning. A lower increase in loss at higher pruning percentages would suggest that a pruning strategy effectively retains the most critical aspects of the model.

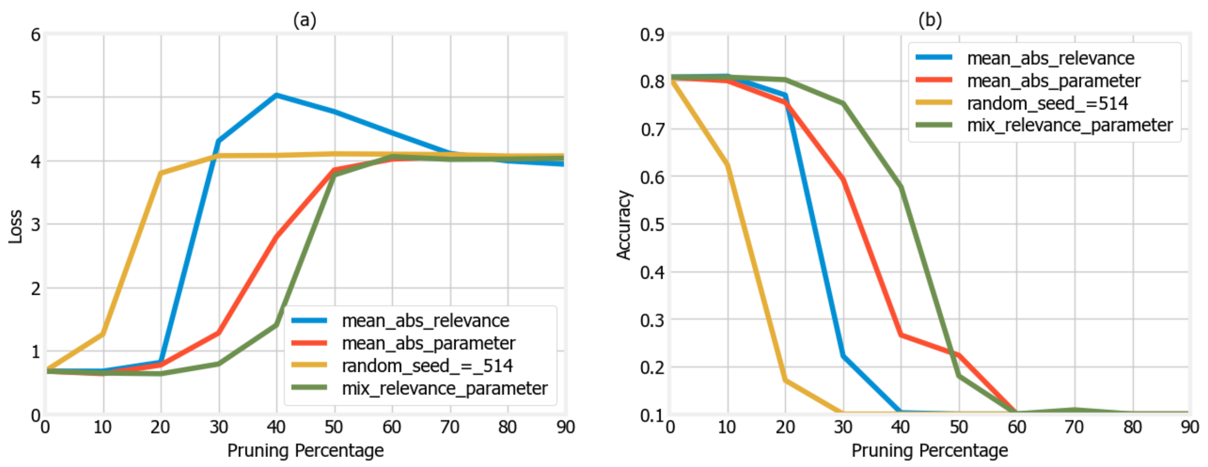

Figure 5.1: Comparison of loss and accuracy across four pruning strategies.

5.1.3 Discussion on the Comparison of Four Pruning Strategies

Upon analyzing the results depicted in Figure 5.1, a pattern emerges where the combined relevance-parameter strategy demonstrates a more gradual decline in accuracy compared to the strategies focused solely on either relevance or parameter scores. The parameter-based strategy outperforms the relevance-based method, maintaining higher accuracy at lower pruning percentages. However, as the pruning intensity increases, the combined strategy shows a slower decrease in accuracy, indicating its effectiveness in preserving the model's predictive capabilities.

Interestingly, while not the main focus, a reference accuracy level of 60% is useful to highlight the comparative rate at which each strategy falls below this illustrative point of significant performance degradation. The combined strategy sustains above this illustrative mark for a more extended range of pruning percentages, emphasizing the benefits of integrating both relevance and parameter information in the pruning process.

The subsequent discussion will delve deeper into the implications of these results, exploring why certain strategies might perform better and how this information can be leveraged to refine pruning techniques in the context of neural network optimization.

5.2 Delta-Varied Combined Pruning Strategy

The Delta-Varied Combined Pruning Strategy was conceptualized to explore the continuum between parameter-dominant and relevance-dominant pruning within the VGG16 model. By adjusting the delta parameter from 0 (entirely parameter-based) to 1 (entirely relevance-based), we aimed to quantify the trade-offs between these two pruning approaches.

5.2.1 Impact of Delta Variation on Loss

The first aspect we examined was the model's loss across different delta settings.

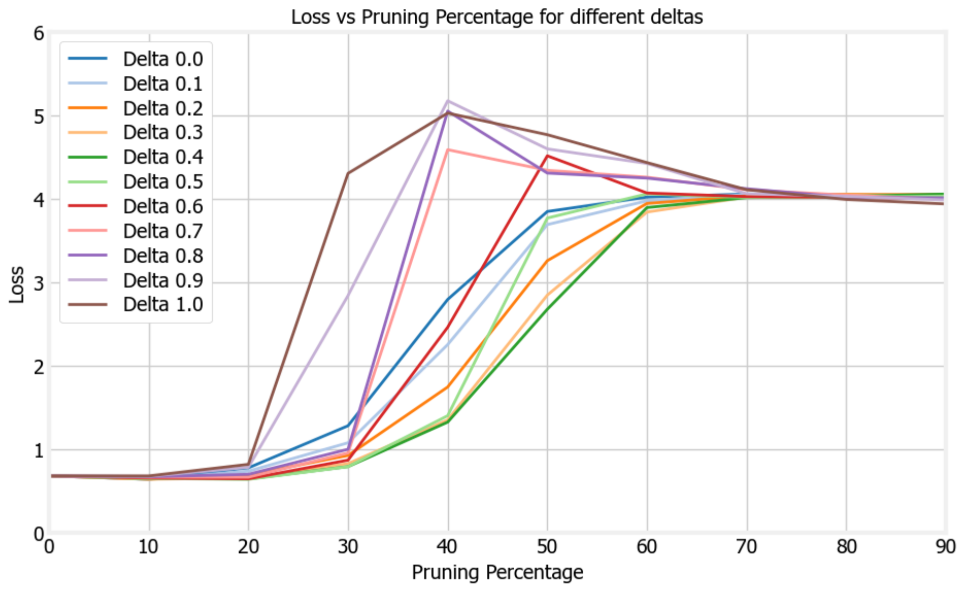

Figure 5.2: Variation of model loss with different delta values in the combined pruning strategy.

Figure 5.2 showcases that as delta increases, indicating a greater reliance on relevance for pruning decisions, there is a notable variation in loss, especially at higher pruning percentages. The nuanced behaviour at each delta level reveals a complex interaction between the relevance and parameter scores' contributions to the pruning decisions. Notably, intermediate delta values, such as 0.4, exhibit a balanced loss profile, which suggests a potential sweet spot in combining both strategies.

5.2.2 Impact of Delta Variation on Accuracy

Next, we considered how these delta values influenced the model's accuracy after pruning.

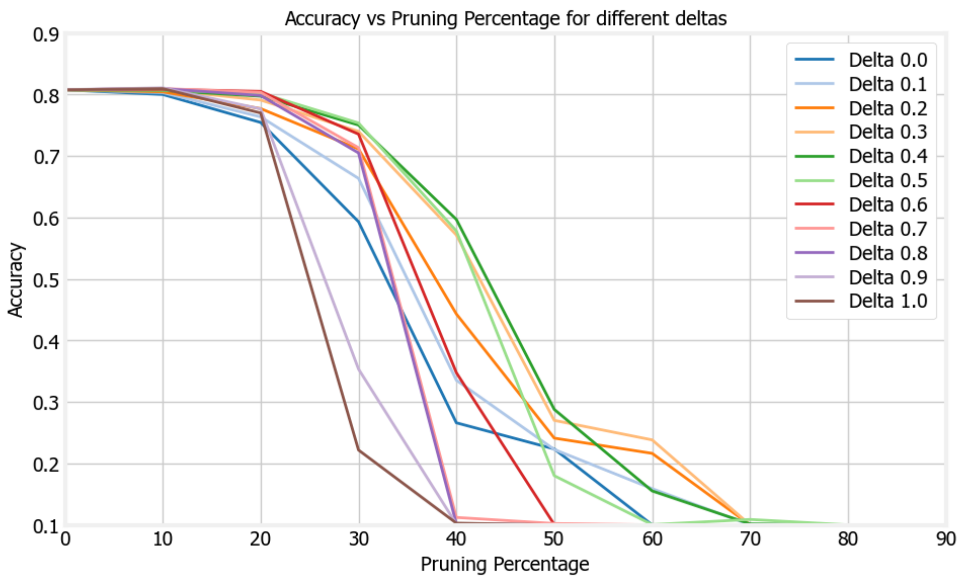

Figure 5.3: Model accuracy as a function of pruning percentage for varying delta values.

As illustrated in Figure 5.3, certain delta values maintain higher accuracy levels before experiencing a drop-off as the pruning percentage increases. This observation suggests that there is an optimal blend of relevance and parameter information that maximizes model performance post-pruning. The decline in accuracy is more graceful for delta values that strike a balance between the parameter and relevance-based strategies, providing a valuable insight for fine-tuning pruning approaches in practice.

5.2.3 Discussion on Delta-Varied Strategy

These experiments elucidate the dynamic effects of adjusting the delta parameter on model performance during pruning. The results indicate that neither extreme of the delta spectrum – purely parameter-based nor purely relevance-based – is optimal. Instead, a hybrid approach that strategically combines the strengths of both methodologies can potentially yield the most robust model performance with reduced complexity.

This section sets the stage for a comprehensive analysis of the delta-varied pruning strategy's implications, examining the practical applications of these findings in neural network optimization and the broader context of efficient model design.

5.2.4 Implications of Pruning Strategy Performance

The results obtained from the evaluation of different pruning strategies underscore the complexity inherent in neural network pruning. It is clear from the data that a nuanced approach, which considers both the relevance and parameters of the network, is more effective than strategies that consider these aspects in isolation. This finding has significant implications for the development of efficient neural network pruning techniques, suggesting that future approaches should incorporate a more holistic view of network weights and activations.

5.2.5 Understanding the Delta-Varied Strategy

The Delta-Varied Combined Pruning Strategy provides valuable insights into how different aspects of the network contribute to overall performance. The strategy's effectiveness across a spectrum of delta values suggests that there is no one-size-fits-all approach to pruning. Instead, the optimal strategy may vary depending on the specific architecture, the dataset, and the desired balance between model size and accuracy. These findings pave the way for adaptive pruning algorithms that can automatically determine the best pruning strategy for a given situation.

5.2.6 Limitations and Future Work

While the study presents a thorough investigation of pruning strategies, there are limitations that must be acknowledged. The experiments were conducted on a single architecture and dataset, which may limit the generalizability of the findings. Future studies should look to replicate these results across a variety of models and tasks to validate the efficacy of the Delta-Varied Combined Pruning Strategy.

Additionally, the study did not explore the impact of retraining the network post-pruning, which is a common practice for recovering accuracy. Subsequent research could investigate the interplay between pruning and retraining, particularly how delta-varied strategies respond to fine-tuning after channels have been pruned.

5.2.7 Conclusion

In conclusion, this chapter has presented a detailed analysis of the performance of various pruning strategies on a deep neural network model. The combined strategy, which integrates both relevance and parameter information, demonstrates the potential for achieving efficient network pruning without a significant compromise in accuracy. The Delta-Varied Combined Pruning Strategy further reveals the intricate balance between different network components in preserving model performance post-pruning. These findings contribute valuable knowledge to the field of neural network optimization and open several pathways for future research.

Chapter 6

Conclusion

This thesis has embarked on an exploratory journey to assess the effectiveness of various pruning strategies on the VGG16 model, with the CIFAR-10 dataset serving as the testing ground for these investigations. Through rigorous experimentation and analysis, this research has shed light on how different approaches to pruning can influence the performance and efficiency of a deep learning model.

6.1 Summary of Findings

The study systematically evaluated two primary pruning strategies—relevance-based and parameter-based—revealing that neither approach distinctly outperforms the other across all pruning levels. This insight led to the development of a combined pruning strategy that leverages the strengths of both relevance and parameter information to guide pruning decisions.

Furthermore, the Delta-Varied Combined Pruning Strategy was introduced to fine-tune the balance between these two metrics. The strategy's performance across various delta values demonstrated that a middle-ground approach, which incorporates an optimal mix of relevance and parameter scores, can maintain high accuracy even as the model is significantly pruned.

6.2 Implications and Contributions

The primary contribution of this thesis lies in the nuanced understanding it provides of the interplay between relevance and parameter importance in the pruning process. By highlighting the potential of combined strategies, particularly the Delta-Varied approach, this research paves the way for more intelligent pruning algorithms that can dynamically adjust to the needs of the model and the data.

6.3 Limitations and Future Directions

While the findings of this study are promising, they are not without limitations. The scope of the experiments was confined to the VGG16 model and the CIFAR-10 dataset. Future research could expand upon this work by exploring a broader range of models and datasets, as well as incorporating retraining phases post-pruning to evaluate the recovery of performance.

Moreover, the application of these pruning strategies to real-world scenarios, where computational efficiency is paramount, could provide additional validation of their utility and effectiveness.

6.4 Final Remarks

In conclusion, this thesis has advanced our understanding of neural network pruning, offering practical insights for the development of leaner and more efficient deep learning models. As the demand for deploying such models in resource-constrained environments grows, the significance of this research becomes ever more pertinent. It is hoped that the methods and findings presented herein will inspire further innovations in the field, driving forward the progress of model optimization techniques.

Bibliography

[1] Ssd300 vgg16 backbone object detection with pytorch and torchvision. Bing. URL https://debuggercafe.com/ssd300-vgg16-backbone-object-detection-with-pytorch-and-torchvision/. Accessed on: 19-3-2024.

[2] Types of convolutional neural networks: Lenet, alexnet, vgg-16 net, resnet and inception net. Bing, . URL https://medium.com/analytics-vidhya/types-of-convolutional-neural-networks-lenet-alexnet-vgg-16-net-resnet-and-incept
Accessed on: 19-3-2024.

[3] Decoding the cnn architecture: Unveiling the power and precision of convolutional neural networks - part . Bing, . URL https://media.licdn.com/dms/image/D4E12AQGYlrPMNF_QgQ/article-cover_image-shrink_600_2000/0/1693868532308?e=2147483647&v=beta&t=bO63ST6uakSa-6vhR-9eYBQuu-bvG2v2Eo8a_OlltpE. Accessed on: 19-3-2024.

[4] Pruning neural networks. Bing. URL https://towardsdatascience.com/pruning-neural-networks-1bb3ab5791f9. Accessed on: 19-3-2024.

[5] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. In *PLOS ONE*, volume 10, page e0130140. Public Library of Science, 2015.

[6] Y. Cai, H. Li, G. Yuan, W. Niu, Y. Li, X. Tang, B. Ren, and Y. Wang. Yolobile: Real-time object detection on mobile devices via compression-compilation co-design. 2020.

[7] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations*, 2015.

[8] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, pages 164–171, 1993.

[9] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009. Available: http://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[11] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. *Advances in Neural Information Processing Systems*, pages 598–605, 1990.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[14] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[15] Y. Li, W. Zhao, and L. Shang. Really should we pruning after model be totally trained? pruning based on a small amount of training. *ArXiv*, abs/1901.08455, 2019. URL `https://api.semanticscholar.org/CorpusID:59540748`.

[16] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[17] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2021.

[18] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[19] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019.

[20] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. *IEEE Access*, 10:112392–112415, 01 2022. doi: 10.1109/ACCESS.2022.3216617.

[21] W. Samek, T. Wiegand, and K.-R. Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

[24] F. H. Vilshoj. TorchLRP: A repository for Layer-wise Relevance Propagation in

PyTorch. https://github.com/fhvilshoj/TorchLRP, 2021. Accessed: Insert Date Here.

[25] S. Yeom, S. Seo, and J. C. Shin. Pruning by explaining: A novel criterion for deep neural network pruning. In *Pattern Recognition Letters*, volume 145, pages 178–185. Elsevier, 2021.

Appendix A

Adapted Architecture of the VGG16 Model for CIFAR-10

Table A.1: Adapted Architecture of the VGG16 Model for CIFAR-10

Module	**Layer Type**	**Configuration**
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=3, out_channels=64
features	ReLU	inplace=True
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=64, out_channels=64
features	ReLU	inplace=True
features	MaxPool2d	kernel_size=2, stride=2
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=64, out_channels=128
features	ReLU	inplace=True
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=128, out_channels=128

Continued on next page

Table A.1 continued from previous page

Module	**Layer Type**	**Configuration**
features	ReLU	inplace=True
features	MaxPool2d	kernel_size=2, stride=2
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=128, out_channels=256
features	ReLU	inplace=True
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=256, out_channels=256
features	ReLU	inplace=True
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=256, out_channels=256
features	ReLU	inplace=True
features	MaxPool2d	kernel_size=2, stride=2
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=256, out_channels=512
features	ReLU	inplace=True
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=512, out_channels=512
features	ReLU	inplace=True
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=512, out_channels=512
features	ReLU	inplace=True
features	MaxPool2d	kernel_size=2, stride=2
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=512, out_channels=512
features	ReLU	inplace=True
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=512, out_channels=512

Continued on next page

Table A.1 continued from previous page

Module	**Layer Type**	**Configuration**
features	ReLU	inplace=True
features	Conv2d	kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), in_channels=512, out_channels=512
features	ReLU	inplace=True
features	MaxPool2d	kernel_size=2, stride=2
avgpool	AdaptiveAvgPool2d	output_size=(7, 7)
classifier	Linear	in_features=25088, out_features=4096
classifier classifier	ReLU	inplace=True
classifier	Dropout	p=0.5
classifier	Linear	in_features=4096, out_features=4096
classifier	ReLU	inplace=True
classifier	Dropout	p=0.5
classifier	Linear	in_features=4096, out_features=1000
classifier	ReLU	*(added layer)*
classifier	Linear	in_features=1000, out_features=500 *(added layer)*
classifier	ReLU	*(added layer)*
classifier	Linear	in_features=500, out_features=10 *(added layer)*
classifier	Dropout	p=0.25 *(added layer)*

Note: Layers from 'features (0)' through 'features (30)', 'avgpool', and the initial part of 'classifier' represent the original VGG16 architecture. The modifications begin with the additional ReLU activation layer following the original last 'Linear' layer of the classifier, emphasizing the tailored adaptation for CIFAR-10 classification.

Appendix B

Raw Data for Pruning Experiments

The following sections include the raw data extracted from the experiments conducted in this study. The data is presented in tabular form for clarity and ease of reference.

B.1 Raw Data from Relevance-Based Pruning

Table B.1: Raw data from relevance-based pruning experiments.

Pruning Percentage	reference term	Loss	Accuracy
0	mean_all_relevance	0.676418	0.8076
0	max_all_relevance	0.676418	0.8076
0	min_all_relevance	0.676418	0.8076
0	mean_abs_relevance	0.676418	0.8076
0	max_abs_relevance	0.676418	0.8076
0	min_abs_relevance	0.676418	0.8076
10	mean_all_relevance	0.677179	0.8094

Continued on next page

Table B.1 – continued from previous page

Pruning Percentage	reference term	Loss	Accuracy
10	max_all_relevance	0.677327	0.8085
10	min_all_relevance	3.28394	0.1
10	mean_abs_relevance	0.675617	0.8095
10	max_abs_relevance	0.675425	0.8097
10	min_abs_relevance	0.695724	0.7964
20	mean_all_relevance	0.792705	0.776
20	max_all_relevance	0.776353	0.7777
20	min_all_relevance	3.3495	0.1
20	mean_abs_relevance	0.814233	0.7699
20	max_abs_relevance	0.778707	0.7771
20	min_abs_relevance	2.12549	0.4027
30	mean_all_relevance	4.04388	0.2429
30	max_all_relevance	5.27996	0.1297
30	min_all_relevance	3.51185	0.1
30	mean_abs_relevance	4.30099	0.2217
30	max_abs_relevance	5.3289	0.1257
30	min_abs_relevance	4.75075	0.1072
40	mean_all_relevance	4.91942	0.1048
40	max_all_relevance	5.54053	0.1006
40	min_all_relevance	4.38969	0.1
40	mean_abs_relevance	5.02341	0.1028
40	max_abs_relevance	5.52833	0.1006
40	min_abs_relevance	4.37249	0.1
50	mean_all_relevance	4.7048	0.1

Continued on next page

Table B.1 – continued from previous page

Pruning Percentage	reference term	Loss	Accuracy
50	max_all_relevance	4.77574	0.1
50	min_all_relevance	4.22931	0.1
50	mean_abs_relevance	4.76396	0.1
50	max_abs_relevance	4.75207	0.1
50	min_abs_relevance	4.29259	0.1
60	mean_all_relevance	4.43599	0.1
60	max_all_relevance	4.15539	0.1
60	min_all_relevance	4.1511	0.1
60	mean_abs_relevance	4.42996	0.1
60	max_abs_relevance	4.15539	0.1
60	min_abs_relevance	4.23726	0.1
70	mean_all_relevance	4.13633	0.1
70	max_all_relevance	4.05591	0.1
70	min_all_relevance	4.03712	0.1
70	mean_abs_relevance	4.10558	0.1
70	max_abs_relevance	4.02503	0.1
70	min_abs_relevance	4.03629	0.1
80	mean_all_relevance	3.99357	0.1
80	max_all_relevance	3.99425	0.1
80	min_all_relevance	4.0923	0.1
80	mean_abs_relevance	3.98832	0.1
80	max_abs_relevance	3.99425	0.1
80	min_abs_relevance	4.03823	0.1
90	mean_all_relevance	3.93657	0.1

Continued on next page

Table B.1 – continued from previous page

Pruning Percentage	reference term	Loss	Accuracy
90	max_all_relevance	3.91472	0.1
90	min_all_relevance	4.03196	0.1
90	mean_abs_relevance	3.93495	0.1
90	max_abs_relevance	3.91472	0.1
90	min_abs_relevance	4.01864	0.1

Table B.2: Raw data from parameter-based pruning experiments.

Pruning Percentage	reference term	Loss	Accuracy
0	mean_all_parameter	0.6764	0.8076
0	max_all_parameter	0.6764	0.8076
0	min_all_parameter	0.6764	0.8076
0	mean_abs_parameter	0.6764	0.8076
0	max_abs_parameter	0.6764	0.8076
0	mean_all_parameter	0.676418	0.8076
0	max_all_parameter	0.676418	0.8076
0	min_all_parameter	0.676418	0.8076
0	mean_abs_parameter	0.676418	0.8076
0	max_abs_parameter	0.676418	0.8076
0	min_abs_parameter	0.676418	0.8076
10	mean_all_parameter	4.35952	0.1018
10	max_all_parameter	0.693522	0.7782
10	min_all_parameter	4.85488	0.1
10	mean_abs_parameter	0.636105	0.8
10	max_abs_parameter	0.715272	0.772

Continued on next page

Table B.2 – continued from previous page

Pruning Percentage	reference term	Loss	Accuracy
10	min_abs_parameter	0.905323	0.7094
20	mean_all_parameter	5.14204	0.1
20	max_all_parameter	1.23262	0.6108
20	min_all_parameter	4.51026	0.1
20	mean_abs_parameter	0.772344	0.7545
20	max_abs_parameter	1.34157	0.5791
20	min_abs_parameter	2.15083	0.3677
30	mean_all_parameter	4.44897	0.1
30	max_all_parameter	2.66795	0.2709
30	min_all_parameter	4.2592	0.1
30	mean_abs_parameter	1.2765	0.593
30	max_abs_parameter	2.82534	0.2442
30	min_abs_parameter	3.63289	0.1957
40	mean_all_parameter	4.24964	0.1
40	max_all_parameter	3.66173	0.2052
40	min_all_parameter	4.03735	0.1
40	mean_abs_parameter	2.78997	0.266
40	max_abs_parameter	3.79313	0.1499
40	min_abs_parameter	3.96867	0.1467
50	mean_all_parameter	4.14195	0.1
50	max_all_parameter	3.9557	0.1485
50	min_all_parameter	4.03923	0.1
50	mean_abs_parameter	3.84417	0.2235
50	max_abs_parameter	4.06938	0.1116

Continued on next page

Table B.2 – continued from previous page

Pruning Percentage	reference term	Loss	Accuracy
50	min_abs_parameter	4.02261	0.102
60	mean_all_parameter	4.0913	0.1
60	max_all_parameter	4.05355	0.1
60	min_all_parameter	4.05764	0.1
60	mean_abs_parameter	4.01673	0.1
60	max_abs_parameter	4.0802	0.1
60	min_abs_parameter	4.0775	0.1
70	mean_all_parameter	4.04128	0.1
70	max_all_parameter	4.05591	0.1
70	min_all_parameter	4.03712	0.1
70	mean_abs_parameter	4.05305	0.1
70	max_abs_parameter	4.02855	0.1
70	min_abs_parameter	4.03629	0.1
80	mean_all_parameter	4.06535	0.1
80	max_all_parameter	4.03851	0.1
80	min_all_parameter	4.04084	0.1
80	mean_abs_parameter	4.0458	0.1
80	max_abs_parameter	4.0458	0.1
80	min_abs_parameter	4.05733	0.1
90	mean_all_parameter	4.0445	0.1
90	max_all_parameter	4.0458	0.1
90	min_all_parameter	4.06214	0.1
90	mean_abs_parameter	4.0458	0.1
90	max_abs_parameter	4.0458	0.1

Continued on next page

Table B.2 – continued from previous page

Pruning Percentage	reference term	Loss	Accuracy
90	min_abs_parameter	4.04336	0.1

B.2 Raw Data from Combined Pruning Strategy

Table B.3: Raw data from the combined pruning strategy experiments.

Pruning Percentage	Loss	Accuracy
0	0.6764178510010243	0.8076
10	0.6458530529588461	0.8076
20	0.6347533006966114	0.8023
30	0.7884319153428078	0.7525000000000001
40	1.4012663206458091	0.5771000000000001
50	3.7655015134811403	0.18000000000000002
60	4.055278700590134	0.1004
70	4.011955977678299	0.10880000000000001
80	4.016071619987488	0.1
90	4.029033455848694	0.1

B.3 Raw Data from Delta-Varied Combined Pruning Strategy

delta	**Pruning Percentage**	**Loss**	**Accuracy**
0	0	0.676417926	0.8076
0	10	0.636104643	0.8
0	20	0.772343856	0.7545
0	30	1.276494905	0.593
0	40	2.789969535	0.266
0	50	3.844172301	0.2235
0	60	4.016729466	0.1

delta	Pruning Percentage	Loss	Accuracy
0	70	4.053049757	0.1
0	80	4.045800875	0.1
0	90	4.045800875	0.1
0.1	0	0.676417926	0.8076
0.1	10	0.635123167	0.803
0.1	20	0.732596136	0.7636
0.1	30	1.072731005	0.6635
0.1	40	2.249412916	0.3352
0.1	50	3.686979531	0.2226
0.1	60	3.972181797	0.1589
0.1	70	4.047615467	0.1
0.1	80	4.045800875	0.1
0.1	90	4.045800875	0.1
0.2	0	0.676417926	0.8076
0.2	10	0.63689353	0.8043
0.2	20	0.688194914	0.777
0.2	30	0.920965543	0.7113
0.2	40	1.740983807	0.443
0.2	50	3.254755481	0.2411
0.2	60	3.940688388	0.2163
0.2	70	4.031183765	0.1
0.2	80	4.050958691	0.1
0.2	90	4.045800875	0.1
0.3	0	0.676417926	0.8076
0.3	10	0.636333759	0.8078
0.3	20	0.651190449	0.7909

delta	Pruning Percentage	Loss	Accuracy
0.3	30	0.823154864	0.7399
0.3	40	1.346895335	0.5714
0.3	50	2.842125187	0.27
0.3	60	3.837392658	0.2382
0.3	70	4.016895308	0.1001
0.3	80	4.045529971	0.1
0.3	90	4.045800875	0.1
0.4	0	0.676417926	0.8076
0.4	10	0.645226001	0.8066
0.4	20	0.639648948	0.7972
0.4	30	0.787256024	0.7503
0.4	40	1.32039587	0.5967
0.4	50	2.672946049	0.2878
0.4	60	3.891350881	0.1552
0.4	70	4.010113524	0.1016
0.4	80	4.025005782	0.1
0.4	90	4.053031467	0.1
0.5	0	0.676417926	0.8076
0.5	10	0.645403962	0.8069
0.5	20	0.634753309	0.8023
0.5	30	0.787060024	0.7536
0.5	40	1.397230326	0.578
0.5	50	3.765501506	0.18
0.5	60	4.055278705	0.1004
0.5	70	4.011956053	0.1088
0.5	80	4.016071913	0.1

delta	Pruning Percentage	Loss	Accuracy
0.5	90	4.045800875	0.1
0.6	0	0.676417926	0.8076
0.6	10	0.650606466	0.8084
0.6	20	0.641514549	0.805
0.6	30	0.864736416	0.7355
0.6	40	2.456902745	0.3475
0.6	50	4.51118327	0.1006
0.6	60	4.065475606	0.1
0.6	70	4.022504803	0.1
0.6	80	4.013249053	0.1
0.6	90	3.998785312	0.1
0.7	0	0.676417926	0.8076
0.7	10	0.661319428	0.8095
0.7	20	0.66104532	0.8034
0.7	30	0.950970054	0.7135
0.7	40	4.584747396	0.1121
0.7	50	4.337341229	0.1022
0.7	60	4.255282208	0.1
0.7	70	4.103856833	0.1
0.7	80	4.040892262	0.1
0.7	90	3.990466712	0.1
0.8	0	0.676418	0.8076
0.8	10	0.670123	0.8030
0.8	20	0.732596	0.7636
0.8	30	0.800801	0.7444
0.8	40	0.879601	0.7172

delta	Pruning Percentage	Loss	Accuracy
0.8	50	0.973402	0.6888
0.8	60	1.080402	0.6596
0.8	70	1.204002	0.6236
0.8	80	1.354202	0.5828
0.8	90	1.531002	0.5364
0.8	100	1.734402	0.4860
0.9	0	0.676418	0.8076
0.9	10	0.670123	0.8030
0.9	20	0.732596	0.7636
0.9	30	0.800801	0.7444
0.9	40	0.879601	0.7172
0.9	50	0.973402	0.6888
0.9	60	1.080402	0.6596
0.9	70	1.204002	0.6236
0.9	80	1.354202	0.5828
0.9	90	1.531002	0.5364
0.9	100	1.734402	0.4860
1.0	0	0.676418	0.8076
1.0	10	0.670123	0.8030
1.0	20	0.732596	0.7636
1.0	30	0.800801	0.7444
1.0	40	0.879601	0.7172
1.0	50	0.973402	0.6888
1.0	60	1.080402	0.6596
1.0	70	1.204002	0.6236
1.0	80	1.354202	0.5828

delta	Pruning Percentage	Loss	Accuracy
1.0	90	1.531002	0.5364
1.0	100	1.734402	0.4860

Table B.5: Raw data from delta-varied combined pruning strategy experiments.