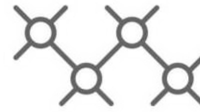


A MASTER THESIS ON

BPLS

Back Propagation Layer Scheduling

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

Diplom-Ingenieur

(Equivalent to Master of Science)

in

Embedded Systems (066 504)

by

Stefan Dangl, BSc

Matr.Nr.: 01631845

Supervisor(s):

Univ.Prof. Dipl.-Ing. Dr.techn. Jantsch Axel

Projektass. Dipl.-Ing. Daniel Schnöll

Ao.Univ.Prof Shinya Takamaeda-Yamazaki, Ph.D. (University of Tokyo)

Vienna, Austria, May 2024

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

In recent years, machine learning has advanced rapidly. Nowadays, neural networks are not only used in large data centers but also on tiny embedded devices, which often have power and performance constraints. Machine learning tasks are highly computationally- and power intensive. Accordingly, concepts for efficient training are in high demand. This work presents the Back Propagation Layer Scheduling (BPLS) approach. BPLS can reduce power consumption and shorten training time by skipping less critical training operations or, depending on the chosen configuration, reduce memory requirements. We investigated BPLS in numerous fine-tuning experiments, featuring the Cifar10, Cifar100 and a custom created key-word dataset. By comparing the weights of various training approaches using Euclidean Distance and Cosine Similarity, we discovered that BPLS and its Per-Layer Learning Rate counterpart approach similar optima. Furthermore, UMAP indicates that they follow nearly identical training paths, towards those optima. We required up to 51.3% less operations for training with improved accuracy. In another test, a reduction in peak memory of up to 49.6%, with only a slight decrease in accuracy (1.9%) was achieved. On MCUs, this reduction in number of operations directly correlates with training time and energy consumption. This allows to train larger networks on smaller and weaker devices. The expected optimizations were validated on an actual MCU.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Kurzfassung

Machine Learning nimmt zunehmend Platz in unsere Gesellschaft ein. Während neuronale Netze nicht mehr nur in großen Rechenzentren anzutreffen sind, findet man sie immer öfters auch auf kleinen Embedded Devices. Machine Learning ist generell sehr energie- und ressourcenintensiv. Gerade Embedded Devices unterliegen oft Ressourcenbeschränkungen, was das Ausführen, aber vor allem auch das Training auf ihnen erschwert. Dementsprechend werden neue Konzepte für effizientes Training benötigt. Diese Arbeit stellt das sogenannte Back Propagation Layer Scheduling (BPLS) vor. BPLS überspringt weniger kritische Trainingsschritte und reduziert dadurch den Stromverbrauch und die Trainingszeit. Abhängig von der Konfiguration kann BPLS auch den Spitzenspeicherbedarf (peak-memory) senken. Im Rahmen dieser Arbeit wurde BPLS in zahlreichen Fine-Tuning-Experimenten, basierend auf Cifar10, Cifar100 und einem speziell dafür erstellten Keyord-Datensatz untersucht. In den Experimenten wurden die Netzwerkkonfigurationen verschiedener Trainingsansätze mittels Euklidischer Distanz und Kosinus-Ähnlichkeit verglichen. Dabei zeigte sich, dass BPLS sich ähnlichen Optima annähert wie das Training mit entsprechenden Layer-spezifischen Learning Rates. Weiters wurde mittles UMAP gezeigt, dass beide Ansätze nahezu identische Pfade zu dem jeweiligen Optimum aufweisen. Des Weiteren haben wir eine Reduktion von 51.3% der Operationen bei verbesserter Genauigkeit erreicht. Bei einem anderen Test wurde der Spitzenspeicherbedarf um 49.6% reduziert, bei leicht verringerter Genauigkeit (1.9%). Auf MCUs korreliert die Reduktion der Operationen mit der Trainingszeit und dem Energiebedarf. Dadurch können größere neuronale Netze auf kleineren und weniger performanten Geräten trainiert werden. Die erwarteten Optimierungen wurden mittels MCU validiert.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Copyright Statement

I, Stefan Dangl, BSc, hereby declare that this thesis is my own original work and, to the best of my knowledge and belief, it does not:

- Breach copyright or other intellectual property rights of a third party.
- Contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
- Contain material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.
- Contain substantial portions of third party copyright material, including but not limited to charts, diagrams, graphs, photographs or maps, or in instances where it does, I have obtained permission to use such material and allow it to be made accessible worldwide via the Internet.

Signature: ______________________

Vienna, Austria, May 2024 Stefan Dangl, BSc

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgment

I would like to thank several people, especially, Daniel Schnöll, MSc. at TU Wien. He made significant contributions to evaluating timing and memory estimations and ensured quality standards.

I am also very grateful to Professor Shinya Takamaeda-Yamazaki for allowing me to execute parts of this thesis at his laboratory at the University of Tokyo. Professor Shinya Takamaeda-Yamazaki is the head of the CASYS (Computer Architecture and Systems) laboratory which focuses on computer architecture, algorithm/hardware co-design for machine learning, and high-level synthesis compilers for productive hardware design.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Abstract **iii**

Kurzfassung **iv**

1 Introduction **1**

2 Background **5**

2.1 Basics of Machine Learning 5
2.2 Structure of Neural Networks 9
2.3 On-device Machine Learning 15
2.4 Overview of Machine Learning Platforms 18
2.5 Training on Embedded Devices 21
2.6 Analyzing and Comparing Neural Networks 29

3 Related Work **35**

3.1 Embedded Training 35
3.2 Static Layer Freezing 37
3.3 Dynamic Layer Freezing 38

4 BPLS **41**

4.1 Base Idea 41
4.2 Definition 42
4.3 Schedules 44
4.4 Notation 45

5 Methodology **47**

5.1 Training Conditions 47
5.2 Selecting BPLS-schedules of Interest 48

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

5.3 Performance Evaluation . . . 48
5.4 Time and Memory Estimation . . . 50
5.5 Similarity Evaluation . . . 51

6 Network Architectures and Scenarios 53
6.1 CIFAR10 - Scenario . . . 53
6.2 CIFAR100 - Scenario . . . 55
6.3 Key-Word - Scenario . . . 55
6.4 MCU - Scenario . . . 57
6.5 Network Architectures . . . 58

7 Results 65
7.1 Terms . . . 65
7.2 Similarity of different Training Approaches . . . 67
7.3 Performance Evaluation . . . 84
7.4 Evaluation of Estimations . . . 100

8 Conclusion and Future Work 107
8.1 Summary . . . 107
8.2 Conclusion . . . 108
8.3 Future Work . . . 110

Bibliography 114

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Tables

2.1 Number of MAC operations - Forward Path 32
2.2 Number of MAC operations - Backward Path 32
2.3 Memory requirements of different layers 33

4.1 Example of Per-Layer Learning Rate training configuration 43
4.2 Example of BPLS-schedule 43
4.3 Example of memory optimizing BPLS-schedule (peak memory reduction = 6.38%, peak operations reduction = 12.32%) 45
4.4 Example of BPLS-schedules 46

6.1 Key-words used in the experiment 56
6.2 Statistics of utilized networks 59

7.1 Cifar10 - Training Performance Summary 86
7.2 Cifar100 - Training Performance Summary - top-1 91
7.3 Cifar100 - Training Performance Summary - top-5 92
7.4 Key Words - Training Performance Summary 95
7.5 Server measurement: bw / fw 101
7.6 Number of Operations required for different layers 103
7.7 MCU measurement: bw / fw 103
7.8 Peak Memory Reduction - with and without Memory Optimization 105

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

2.1 VGG16 Architecture [1] 10
2.2 2D - Convolution Example [2] 11
2.3 Sigmoid and ReLU activation functions 13
2.4 Max-Pool with 2x2 kernel and stride of 2 [3] 13
2.5 EMA with different β-values [4] 22
2.6 Comparison of different learning rate schedules [5] 24
2.7 Cyclical Learning Rate Adjustment [6] 24
2.8 Network before and after pruning [7] 25
2.9 *Energy consumption and prediction accuracy of a DNN as a function of the Arithmetic Precision adopted for a single MAC unit in a 45 nm CMOS.* [8] 27
2.10 UMAP - clustering of image classification [9] 31

5.1 Learning Rate Grid Search 50

6.1 CIFAR10 - original for pre-training (top) and modified for fine-tuning (bottom) 54
6.2 CIFAR100 - original for pre-training (top) and modified for fine-tuning (bottom) 55
6.3 Key-words - Go! for pre-training (left), 行け (ike!) for fine-tuning (right) 57
6.4 SmoothSubspace - UCR Time Series Classification Archive [10] 58
6.5 Network of the Cifar10 - Scenario (Pre-Training: left, Fine-Tuning: right) 60
6.6 Network of the Cifar100 - Scenario (Pre-Training) 61
6.7 Network of the Cifar100 - Scenario (Fine-Tuning) 62
6.8 Network of the Key Word - Scenario (Pre-Training and Fine-Tuning) 63
6.9 Network used for time and memory measurements on MCU and Desktop 64

7.1 Cifar10 Network - Distance Polar Plots 69
7.2 Cifar10 Network - Angle Polar Plots 71
7.3 Cifar10 Network - Distance Color Maps 74

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

7.4 Cifar10 Network - Angle Color Maps 76
7.5 Cifar10 Network - UMAPs 78
7.6 Cifar100 Network - UMAPs 79
7.7 Cifar100 Network - Distance Polar Plots (Baseline: Per-Layer Learning Rate) 80
7.8 Cifar100 Network - Distance Polar Plots (Baseline: BPLS) 82
7.9 Cifar10 Network - Training Performance 87
7.10 Cifar10 Network - Training Performance (best accuracy) 89
7.11 Cifar10 Network - Epochs per Operations 90
7.12 Cifar100 Network - Training Performance 93
7.13 Cifar10 Network - Epochs per Operations 94
7.14 KW Network - Training Performance 96
7.15 KW Network - Epochs per Operations 97
7.16 Comparison between Server measurement and estimated number of operations (bw + fw) 101
7.17 Server measurements - Histogram 102
7.18 Comparison between MCU measurements and estimated number of operations 104
7.19 Estimated peak memory requirements and static stack information 104

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Acronyms

AI Artificial Intelligence. 5, 21

ASIC Application Specific Integrated Circuit. 19, 20, 21

BPLS Back Propagation Layer Scheduling. xi, xiv, 2, 3, 18, 26, 35, 37, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 65, 67, 68, 69, 70, 71, 72, 73, 75, 77, 79, 81, 82, 83, 84, 85, 86, 88, 89, 90, 91, 92, 95, 96, 97, 98, 99, 104, 105, 107, 108, 109, 110, 111, 112, 113

CNN Convolutional Neural Networks. 10, 12, 13, 20, 31, 35, 37, 41, 53, 55

CPP C Plus Plus (C++). 33, 50, 51, 103

CPU Central Processing Unit. 19, 20, 50, 51, 100, 101, 102, 103

DKL Kullback – Leibler Divergence. 38

EMA Exponential Moving Average. 14, 22, 23

FPGA Field Programmable Gate Array. 1, 15, 18, 19, 20, 21

FPU Floating-Point Unit. 50, 51

GPU Graphics Processing Unit. 19, 20, 21, 28, 38

I/O Input / Output. 20

LLM Large Language Model. 5

MAC Multiplication and Accumulation. xi, 26, 27, 31, 32, 43, 45, 50, 84, 85, 88, 89, 92, 96, 97, 109, 110

MCU Microcontroller Unit. 1, 2, 15, 17, 18, 19, 20, 32, 33, 47, 48, 50, 51, 58, 88, 100, 101, 102, 103, 104, 105, 109, 112

MSE Mean Squared Error. 6

OS Operating System. 28, 33

QAT Quantization-Aware Training. 26

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

RAM Random-Access Memory. 20

ReLU Rectified Linear Unit. 12

SGD Stochastic Gradient Descent. 48

SRAM Static Random-Access Memory. 20, 50

TMAC Tera-MAC (10^{12} MACs). 87, 89, 93, 96

TPU Tensor Processing Unit. 19, 21, 28

UMAP Uniform Manifold Approximation and Projection. xiii, 30, 31, 52

VGG Visual Geometry Group. 53

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 1

Introduction

The field of machine learning has experienced substantial advancements in recent years. Interesting papers in this area are published almost monthly. Beyond academia, machine learning has led to significant improvements in solving real-world problems, particularly in the fields of natural language processing [11], automotive industries [12], healthcare [13], and many others.

On-device machine learning is a subcategory of machine learning. It involves performing inference and training of neural networks on standalone devices such as laptops and smartphones. Additionally, machine learning on embedded devices such as Microcontroller Units (MCUs) and Field Programmable Gate Arrays (FPGAs) has become increasingly popular recently. While on-device inference simply involves executing previously trained neural networks, on-device training refers to training networks locally using data samples available on the device.

On-device inference has been common for a while now. However, training has typically been performed on powerful servers rather than directly on local devices, primarily due to computational limitations of embedded platforms. In recent years, embedded platforms have become increasingly powerful and energy-efficient, making them more suitable for on-device training. Training on these devices is also referred to as embedded training, tiny training, or training on high-constraint devices.

On-device training offers certain benefits. It not only enables the optimization of neural networks based on the local environment but also ensures independence from internet connectivity. Consequently, this leads to a significant improvement in security by allowing users to keep their data completely private. Moreover, on-device training conserves network bandwidth, reduces latency, and saves energy because no data needs to be uploaded to the cloud and no updated model needs to be sent back to the device.

This enables the implementation of smart and secure embedded solutions. For example, smart sensors can be installed near technical equipment inside factories. These sensors enable the detection of anomalies, such as data drifts, help to reduce communication traffic by abstracting data, and ensure timely execution of maintenance tasks. These sensors often contain neural networks running on an integrated MCU. The MCU is not only responsible for analyzing data by performing on-device inference but can also retrain the deployed neural network to adapt to the local environment over time. Another example could be a medical device that learns to provide personalized predictions for a specific patient. This allows medical situations to be classified with very high accuracy while maintaining the privacy of sensitive data.

Despite the advancements in embedded platforms mentioned earlier, they still encounter limitations when dealing with large neural networks or require additional optimizations to enhance energy efficiency and reduce latency. Consequently, various methods have been introduced to accelerate training, improve energy efficiency, and minimize memory usage on embedded device. The simplest way to optimize the training of neural networks on embedded platforms involves selecting appropriate training optimizers and hyperparameters, such as batch size. Additionally, there are several fundamental model-side optimization techniques:

- **Quantization**: Utilizing reduced-precision representations, such as 16-bit, 8-bit, or even lower integer values instead of the commonly used 32-bit full-precision floating-point format, can reduce memory requirements for both inference and training. However, this approach may also result in a reduction in model accuracy.
- **Pruning**: Removing less important nodes or connections from a neural network or ignoring less important activations.
- **Sparse Updates**: Instead of updating the entire neural network, only a subset of parameters is updated while the rest are frozen. For example, updates can be restricted to biases while weights remain unchanged.

There are further advanced techniques published recently, such as MiniLearn [14], TinyOL [15], and POET [16]. A more detailed introduction to efficient training on resource-limited devices can be found in the Background and Related Work chapters of this thesis.

Independent of the mentioned techniques, this thesis introduces a different approach for efficient fine-tuning of neural networks on resource-constraint devices. This approach is called Back Propagation Layer Scheduling, abbreviated as BPLS.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Different from the previously mentioned approaches, BPLS focuses on reducing power consumption and memory requirements, as well as speeding up training by applying a predefined and well-suited training schedule that skips certain layer updates and corresponding computations, while minimizing the impact on network accuracy. Furthermore, BPLS is designed to be compatible with other approaches for efficiently fine-tuning neural networks on resource-constrained devices.

The following chapters, Background and Related Work, contain an overview of the theory behind this thesis and corresponding state-of-the-art concepts. They provide all the required information to understand the upcoming explanations. This is followed by a detailed description of BPLS, which includes its benefits, restrictions, and important aspects to note. Subsequently, the Methodology chapter describes how the experiments were executed and which datasets and networks were used. Finally, the experimental results and a final conclusion are presented.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Chapter 2

Background

This chapter presents the theoretical background necessary to understand this thesis. It begins with fundamental concepts of machine learning, with a specific focus on the types of neural network elements used in the experiments featured in this work. Additionally, it covers the basic concepts of on-device training and training under high constraints. The chapter concludes with models and metrics for analyzing neural networks.

2.1 Basics of Machine Learning

Even when machine learning started booming in the early 2020s, with impressive Large Language Models (LLM) [17] such as GPT [18] or AI for image synthesis [19] such as DALL-E [20], the actual basics of machine learning date back to the 1990s. Basically, machine learning algorithms autonomously improve mathematical models using sample (training) data to make decisions without explicit programming. This allows to solve problems that are challenging to address using conventional methods, such as speech recognition. However, despite its seemingly high complexity, the underlying mathematical principles of machine learning are relatively simple. The subsequent pages provide an overview of the basics of training neural networks.

2.1.1 Categories of Machine Learning

Machine learning is about using known data to make predictions about unknown data. The training of neural networks can be performed in various ways, categorized into Supervised, Unsupervised, and Reinforcement Learning.

In Supervised Learning, labels attached to the training data allow the neural network to correlate features. Supervised Learning is used to solve tasks like regression and classification. Regression is used to predict values based on the learned patterns and classification to recognize data such as images and audio.

The other categories are Unsupervised Learning and Reinforcement Learning. In Unsupervised Learning, the network is trained using unlabeled data. It is commonly employed to solve clustering problems or perform anomaly detection. On the other hand, in Reinforcement Learning, the network interacts directly with a task and receives rewards based on its actions. This enables networks to maximize performance by autonomously determining optimal behavior within a specific context and improving decision-making over time. Reinforcement Learning is often utilized in robotics and game AI to enhance specific abilities. A notable example of reinforcement learning is AlphaStar, *the first artificial intelligence (AI) system to beat a professional player at the game of StarCraft II* [21]. In addition, Reinforcement Learning also shows potential in other industries. For example, NVIDIA's AutoDMP [22] uses Reinforcement Learning to optimize macro placement in chip design.

Additionally, training neural networks can be categorized as Online [23] and Offline Learning. Offline Learning is far more common and refers to cases where a training dataset is prepared in advance before training the model. On the other hand, Online Learning involves training during inference, meaning that the actual real-world application data used for inference is simultaneously used to update the network.

2.1.2 Concept of Supervised Machine Learning

Deep neural networks consist of several layers. The term 'deep' refers to the depth of the network, measured by the number of layers. [24] These layers contain weights and biases. The goal of the training process is to adjust these parameters in a specific way. Unfortunately, these adjustments cannot be computed using analytical methods. Instead, they must be discovered empirically.

In Supervised Learning, input data is labeled, meaning that the actual output can be compared to its ground truth. For example, the input data could be a collection of handwritten digits, where the labels represent the numbers from 0 to 9, just as in the well-known MNIST dataset [25].

To compare the network output with its ground truth, an error function, also known as cost or loss function is used. There are several types. For example, the cross-entropy error function is often used for image classification. However, one of the simplest is the Mean Squared Error (MSE), where $\widehat{y}$ represents the ground truth, y represents the output of the network for the corresponding input data,

TU Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

and i is the index of the output class.

$$L_{\text{MSE}} = \frac{1}{2} \sum_i (\widehat{y}_i - y_i)^2 \tag{2.1}$$

Usually, the gradient of the error function with respect to the network parameters is used to adjust the weights and biases in the opposite direction of the gradient, employing the widely known Gradient Descent Algorithm. Ideally, this process reduces the error step by step until a minimum is reached. Deep neural networks consist of several layers. To adjust the weights and biases of layers further away from the output, the error information must be propagated through the network, a process known as Back Propagation. Back Propagation essentially involves applying derivation' s chain rule. While there are alternative algorithms, Gradient Descent is by far the most common method used to train neural networks. More detailed explanations about the algorithm can be found here [26].

The data used to train and validate neural networks is organized into datasets. Typically, datasets are split into Training (60-80%), Test (10-20%), and Validation (10-20%) subsets. The Training Set is usually the largest and is used to update the model parameters (weights and biases) during the training process. The Validation Set, on the other hand, is used to evaluate the model's performance during training. It is also employed for tuning Hyperparameters and observing overfitting, which are terms explained later on. The network state that yields the best results on the Validation Set is typically chosen as the final network configuration, making this final configuration biased by the Validation Set. To evaluate the network under real-life conditions, the Test Set is used. The Test Set is not utilized during training or Hyperparameter tuning. Instead, it is employed after training is complete to assess the final performance and generalization ability of the trained network on unseen data. [24]

The specific division into subsets depends on the dataset size, task complexity, and the availability of data. In many cases, the data is simply split into two parts instead of three: Training Set (80%) and Validation Set (20%), while the Validation Set is often referred to as Test Set in this case [27], just as in this thesis.

The training process can be monitored using four primary metrics: train and test - error/accuracy. The error metric quantifies the difference between ground truth and predicted outputs averaged over all output classes. Accuracy measures the proportion of data samples correctly classified.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.1.3 Hyper Parameters

To achieve satisfactory training behavior, the so-called hyperparameters must be configured correctly. Hyperparameters differ from network parameters like weights and biases in that they do not define the network but rather the training process. They impact how quickly the network learns, the quality of the resulting network, and the resources required for training. Some important hyperparameters include Learning Rate, Batch Size, and the Number of Training Epochs.

Starting with the most basic one, the Number of Training Epochs defines how many times the complete training data is applied to the network. One epoch corresponds to one iteration through the training data. While several epochs are usually required to achieve satisfactory results, training for too many epochs can have negative effects. Overfitting can occur when the network becomes overly specialized on the training data, losing its ability to deal with general, yet unknown data. Decreasing test-accuracy by high train-accuracy indicates Overfitting.

The Training Batch Size determines how many training data samples are processed before the network is updated. The computed weight and bias changes, also called gradients are computed for a batch of input data, averaged, and finally applied, resulting in an updated network. In general, larger batches lead to improved training speed but also increase memory requirements because gradients must be cached. The size of a batch can be as small as one data sample but can also contain the complete training data. Batches containing less than the complete training data are referred to as Mini-Batches. Besides Training Batch Size, the Inference Batch Size defines how many samples are simultaneously applied for validating the network, affecting memory requirements and validation speed.

Another significant hyperparameter that affects training performance is the learning rate. The learning rate simply represents a multiplication factor determining how intensely the weights and biases are modified when updating. It can be difficult to determine the right learning rate for a specific network, data set, and batch size. Low learning rates often lead to good network quality but require many epochs for training. Also, low learning rates pose the risk that training converges at a local optimal network configuration which likely does not perform as well as a global optimum. Contrary, high learning rates can reduce the number of epochs required to find good network configurations but may result in overshooting optimal configurations. Learning rates are often adjusted over time using learning rate schedules. Those will be explained later.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.1.4 Data sets

Training neural networks requires a lot of data. Fortunately, relatively extensive datasets are available online for free use. Some of the most famous datasets for image recognition include the already mentioned MNIST [25], CIFAR-10/100 [28], and ImageNet [29]. Others, for example, consist of audio keyword samples like Google's Speech Commands dataset [30].

Even though common datasets consist of several thousand samples, the earlier mentioned overfitting phenomenon can still occur. There are several ways to prevent the network from overfitting. For example, using smaller networks or even larger datasets. A simple way to increase the amount of available training data is through data augmentation. Simple augmentation methods for image data include flips around the vertical axis, rotations, cropping, scaling, or color modifications. These modifications can be applied to the images with a certain probability, leading to slightly different training data for each epoch. Most machine learning frameworks provide predefined functions to simplify data augmentation [31].

Besides augmentation, normalization is another important data preprocessing step. Non-normalized data samples with wide ranges can cause instability during training. For example, one feature of a dataset ranges from 1 to 10 and another from 1 to 999. This could lead to large updates applied to one weight due to its large gradient, preventing the Gradient Decent algorithm from converging. On the other side, smaller updates could be applied to another weight due to its smaller gradient, resulting in smaller steps and accordingly longer training time. Therefore, datasets are often normalized before training. There are different normalization approaches, such as scaling the dataset to a range from 0 to 1 or modifying the dataset to have a mean of 0 and a standard deviation of 1. In other words, normalization brings the different data samples into a consistent range, which can improve learning speed and network quality.

2.2 Structure of Neural Networks

As previously mentioned, deep neural networks are composed of a sequence of layers. These are among others Convolutional Layers, Linear Layers, and Max Pooling Layers. Layers closer to the input of the network are responsible for recognizing basic structures, while layers closer to the output are responsible for recognizing details. [32]

Diagram 2.1 shows the VGG16 network as an example. Convolutional Layers extract features from the input data while Pooling Layers summarize these features. Extracted features are stored in so-called feature maps. In this example, the input image has dimensions of 224 x 224 x 3 (width, height,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

channels), whereas subsequent feature maps are of dimensions such as 56 x 56 x 256 or 7 x 7 x 512. Accordingly, when propagating through the network, the size of the feature maps typically decreases while the number of channels increases.

The network's output is usually a Linear Layer which allows to classify the extracted features. To confine the output values within a range of 0 to 1, a Softmax Layer can be attached. Typically, after each Convolutional and Linear Layer an activation function is applied. In addition to the depicted layers, it is common to incorporate Dropout and Batch Normalization Layers to enhance the training process.

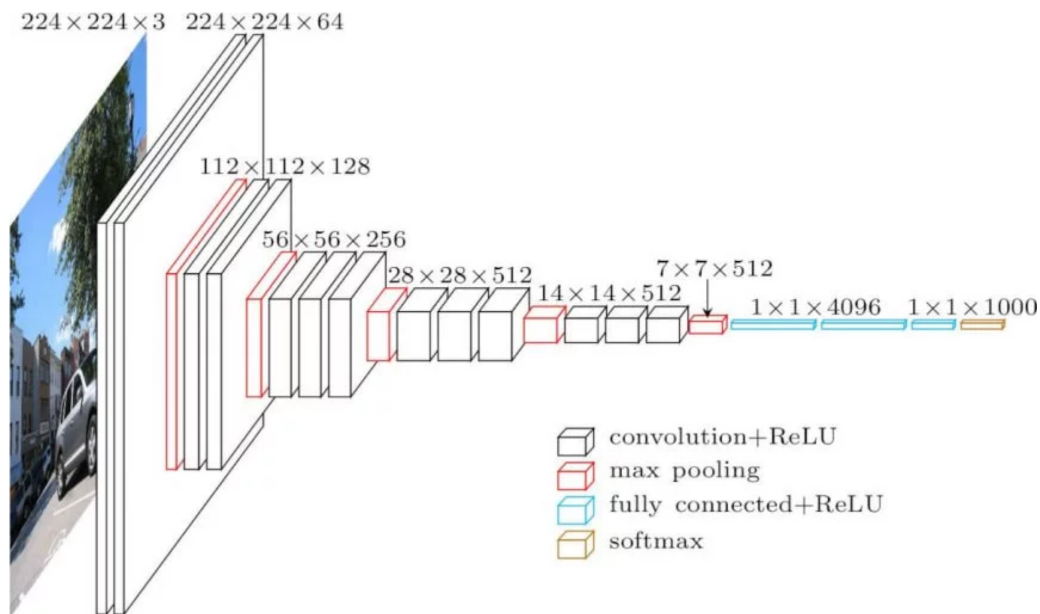


Figure 2.1: VGG16 Architecture [1]

2.2.1 Convolutional Layers

Convolutional Layers form the backbone of Convolutional Neural Networks (CNN), which are specialized for computer vision tasks. As the name suggests, these layers are based on mathematical convolution operations. In general, a filter is systematically applied to the input to produce a feature map. The filter consists of trainable weights.

The convolution operation involves computing a dot product between the filter and a patch of the input, resulting in a single value. Common filter sizes include 3x3, 5x5, and 7x7. Computing this dot product for every filter-sized patch of the input by shifting the filter from left to right and top to bottom results in a feature map.

In practice, Convolutional Layers often receive not just a single 2D image or feature map but multiple ones. For instance, a color image in RGB format consists of three channels: red, green, and blue. The

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

number of input channels is often referred to as input depth. An input of depth 3 requires a filter of depth 3. Consequently, a 3x3x3 (width, height, depth) filter comprises 27 trainable weights. The number of different filters applied to the input determines the number of output channels, which represents the input depth of the subsequent layer. In other words, every filter produces its own feature map.

Convolutional Layers positioned at various depths within a network extract different types of information. Layers near the input tend to learn to extract low-level features like lines, whereas those closer to the output focuses on features such as cars, faces, or animals. Those in between learn to extract simple shapes. Thus, stacking multiple Convolutional Layers sequentially leads to a hierarchical decomposition of the input.

Figure 2.2 illustrates the principle of 2D convolution. In practice, various configuration options exist, such as how to handle edges (padding) and how to move the filter across the input data (stride). Apart from 2D convolutions used for image data, 1D convolutions are common for processing one-dimensional data like sensor signals.

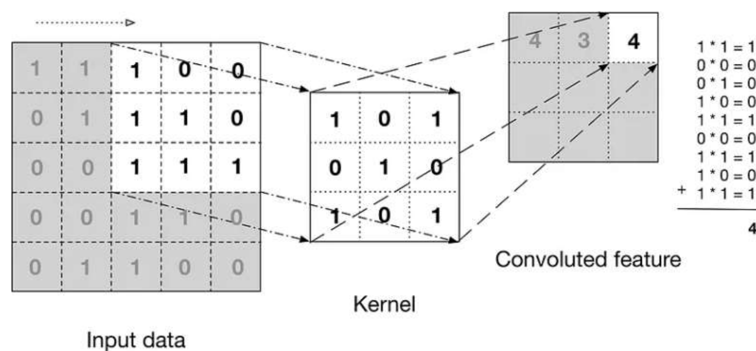


Figure 2.2: 2D - Convolution Example [2]

2.2.2 Linear Layers

Linear Layers, often referred to as Dense Layers or Fully Connected Layers, are relatively simple in structure. The output value is computed by multiplying each input to a specific output by the corresponding weight, summing up the results, and then adding a bias. In other words, Linear Layers represent a vector-matrix multiplication that can be described using the following formula:

$$y_j = b_j + \sum_i x_i \cdot w_{i,j} \tag{2.2}$$

Here, y_j defines the value of output node j, b_j represents the corresponding bias, x_i is the value of the input feature i, and $w_{i,j}$ is the corresponding weight.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Linear Layers are frequently used to reshape or transform the dimensions of the data within neural networks. When placed at the output of networks, Linear Layers are often referred to as Classifier Layers, particularly in classification tasks.

2.2.3 Activation Functions

As mentioned earlier, activation functions are typically applied after each Convolutional and Linear Layer. They play a crucial role in solving complex problems and enable the propagation of data through deep networks.

There are several types of activation functions, broadly categorized into two commonly used groups. The first group includes the Rectified Linear Unit (ReLU), defined as $f(x) = \max(0, x)$. ReLU is especially popular in CNNs, computationally efficient, and typically performs well during training. However, due to mapping negative values to zero, it causes the "Dying ReLU" problem [33], a subcategory of the Vanishing Gradient Problem [34]. This causes some nodes to die and stop learning. ReLU does not saturate for positive input values. This has positive effects in many cases, however, does not prevent problems caused by extremely high positive outputs of the upstream layer at all.

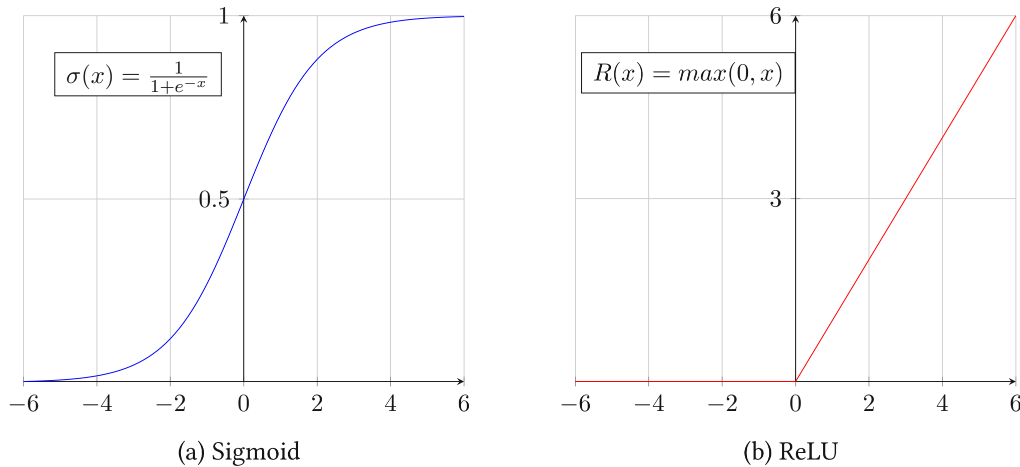
ReLU is an old but still very commonly used activation function. However, there are extended versions that solve some of the mentioned problems. For instance, Leaky ReLU is defined as $f(x) = \max(\alpha x, x)$ with α being a Hyper Parameter usually set around 0.01. This solves the "Dying ReLU" problem to some extent. Another variant, ReLU6, is defined as $f(x) = \min(\max(0, x), 6)$, limiting the output to a maximum value of 6 and preventing excessively high activations that could cause gradient instability.

The second important category of activation functions includes Softmax and Sigmoid functions, both of which produce output values between 0 and 1. In classification networks, these functions are commonly used at the output layer, where they normalize the output into probabilities corresponding to different classes. At other network types, such as RNN and Transformers, these functions play more significant roles.

2.2.4 Pool Layer

One of the most common types of Pooling Layers is Max-Pooling. Max-Pooling layers are often used after Convolutional Layers. They facilitate the extraction of prominent features, such as the eye of a cat, while discarding less relevant information, such as the background of an image. Max-Pooling achieves this by dividing the input feature map into smaller regions and only propagating the highest value from each region.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub



(a) Sigmoid (b) ReLU

Figure 2.3: Sigmoid and ReLU activation functions

In Figure 2.4, a Max-Pooling Layer with a 2x2 kernel and a stride of 2 is illustrated. As shown, Max-Pooling selects the maximum value from each 2x2 block and outputs it. Consequently, the size of the output feature map is half the size of the input feature map. This down-sampling not only has a positive impact on the network quality but also reduces the computational load on subsequent layers.

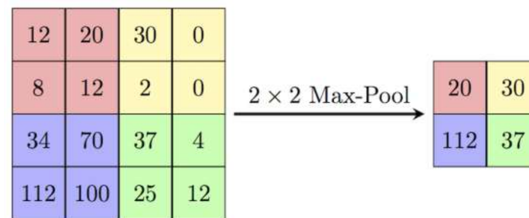


Figure 2.4: Max-Pool with 2x2 kernel and stride of 2 [3]

It is common to incorporate multiple Max-Pooling Layers within a network. However, it is generally advised against applying Max-Pooling in the initial stages of CNNs as the Kernels would be extracting edges and gradients instead of useful features. Furthermore, the final feature maps should ideally maintain a certain size and not be excessively reduced.

In addition to Max-Pooling and Average-Pooling, which computes the average value in each region, there are more sophisticated pooling techniques such as Global Average Pooling [35], Region of Interest Pooling [36], and Spatial Pyramid Pooling [37].

2.2.5 Batch-Normalization Layer

Batch Normalization Layers [38] are commonly placed after Convolutional and Linear Layers to stabilize training. The outputs of one layer serve as the inputs to the next, much like how the dataset serves as input to the first layer. In the previous section, the normalization of datasets was described. Similar

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

principles can be applied to the data between layers.

A Batch Normalization Layer consists of two trainable parameters, beta (β) and gamma (γ), along with two non-trainable parameters, Mean Moving Average (μ) and Variance Moving Average (σ^2), referred to as the state of the layer. The inputs to a Batch Normalization Layer are referred to as activations. The activations over a batch applied to one single input of the Layer are combined into an activation vector. Batch Normalization computes the mean and variance for each activation vector separately. These mean and variance values are then used to normalize, shift, and scale the output of the layer based on the trained γ and β parameters, optimizing the training behavior.

In the inference phase, it is beneficial to have access to the mean and variance computed during training. However, computing and saving the mean and variance for the entire dataset would be computationally expensive. Instead, Batch Normalization Layers compute an EMA (Exponential Moving Average) of the mean and variance during training, which is used in the inference phase. This method is more efficient as it only requires storing the most recent value of the moving average. The normalization process represents a linear function. Consequently, computational efficiency can be further improved by fusing Convolutional or Linear Layers with their subsequent Batch Normalization Layers. [39]

The following formulas describe Batch Normalization Layers mathematically. Here, x_i represents the activations, $\hat{x}_i$ represents the normalized output, m represents the batch size and ϵ is a small constant added for numerical stability:

Compute Mean and Variance:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.3}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{2.4}$$

Normalize:

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.5}$$

Scale and Shift:

$$y_i = \gamma \hat{x}_i + \beta \tag{2.6}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

2.2.6 Dropout Layer

The fundamental idea behind Dropout Layers [40] [41] is to randomly drop features during training. This helps prevent overfitting, enabling the construction of larger and more robust networks. Without applying Dropout Layers, neural networks may become overly reliant on specific features, leading to the dominance of certain weights while others become negligible. Over time, this can cause some weights to stop learning effectively.

To address this, in each training step, Dropout Layers randomly adjust the network architecture by dropping features with the probability 1 - p. Dropout probabilities for large networks are commonly between 0.2 and 0.5. When Dropout Layers are placed directly at the input of the network, 1 - p should be kept lower than 0.2 to avoid excessive feature dropout.

During inference, Dropout Layers are deactivated. Since the input values are not randomly dropped, the output values would be higher than during training. To compensate for this, the weights are scaled by the retaining factor p.

More advanced variations of Dropout, such as Gaussian Dropout [42], provide comparable or improved performance. Dropout is considered a regularization technique. Other regularization techniques include L1 and L2 regularization [43], which are not applied in this thesis.

2.3 On-device Machine Learning

This thesis focuses on fine-tuning neural networks on resource-constrained devices. Therefore, a general overview of on-device machine learning is provided in this section. On-device machine learning refers to performing machine learning tasks, including both inference and training, directly on local devices. These devices can range from laptops and smartphones to Microcontroller Units (MCUs) and Field Programmable Gate Arrays (FPGAs). The machine learning tasks executed on these devices include local data classification, such as processing images, text, sensor signals, or audio recordings directly on the device instead of transmitting the data to a server. On-device machine learning offers various advantages but also comes with limitations, which will be discussed later [44].

Typically, neural networks are not trained from scratch on-device. Instead, networks pre-trained on powerful workstations or servers are employed and locally adapted to improve or customize their behavior. This reduces the number of training epochs, thus saving energy and time compared to training from scratch. Local network adjustments are broadly categorized into fine-tuning and transfer learning. Transfer learning involves repurposing a pre-trained network for different tasks, such as classifying

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

additional or different categories, while fine-tuning refers to adjusting the pre-trained parameters to optimize network behavior.

2.3.1 Transfer Learning and Fine-Tuning

Transfer Learning allows the application of pre-trained networks for new tasks by transferring model knowledge between data domains. This is typically achieved by adding a new Linear Layer after or instead of the output of the pre-trained network. In classification networks, this new output layer may differ in the number or type of classes. For example, a network trained to classify audio keywords such as "up," "down," "left," "right," and numbers from "0" to "9" might have 14 output classes. However, the same network can be adapted to classify only a subset of these keywords or entirely new words using transfer learning.

While a transferred network benefits from pre-training, its performance may not be optimal for the new task. To enhance performance, the network is typically fine-tuned. As mentioned earlier, layers closer to the input extract general features, while those closer to the output detect details. Therefore, as long as the new data domain shares similarities with the pre-training data, layers closer to the output require more significant adjustments than those closer to the input.

Typically, during fine-tuning, some layers closer to the input are frozen, meaning they are not updated during training. This not only saves computational resources but especially preserves well-trained and generalized parts of the network, leading to better and especially more general performing networks. However, determining which layers to freeze can be challenging if the new data is not well known.

Even with frozen layers, Catastrophic Forgetting [45] is a common issue in transfer learning, particularly in continuous learning scenarios where the network continually learns from new data rather than performing explicit retraining. Catastrophic Forgetting can cause the network to forget previously learned knowledge, impacting its ability to recognize previously identified objects or patterns. Lower learning rates and periodic retraining on previous data can mitigate this problem. [46] However, retraining on previous data requires storing data samples, potentially leading to memory shortages.

Fine-tuning neural networks is not restricted to transfer learning. It is common to locally fine-tune pre-trained networks to improve or personalize network behavior. Going back to the previous key-word example. Deploying a pre-trained network on a smartphone owned by an individual and fine-tuning it using data recorded by that person can optimize the network's performance for that specific user.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

An interesting concept related to fine-tuning is Per-Layer Learning Rate. As mentioned, layers closer to the output typically require stronger adjustments during fine-tuning compared to those closer to the input. Using excessively high learning rates can negatively affect training behavior. These adjustment requirements lead to different learning rate limits per layer. In other words, assigning layer-specific learning rates allows the application of higher learning rates at later layers while keeping the learning rate at earlier layers low. While this approach may not necessarily improve network quality, it often reduces the number of epochs required for training.

2.3.2 Benefits of On-device Training

The best way to introduce the benefits of on-device training is by examining real-life examples. Considering improving home assistants like Amazon's Alexa [47], users with strong accents or speech impairments, such as stuttering, cluttering, or lisping, may face challenges in being understood correctly. On-device training allows for improvement in such cases. When the home assistant encounters a word it classifies with low confidence, it could prompt the user to confirm if the classification was correct. This feedback can then be used to enhance the network. On-device training in this example facilitates personalizing the network while ensuring privacy.

Another example is personalized face recognition on smartphones. Typically smartphones can be unlocked by simply showing the owner's face to the camera. To achieve this, users capture multiple images of their faces, which can then be used for fine-tuning a neural network directly on the device. Just as in the previous scenario, on-device training ensures that private data remains on the device and does not need to be transmitted to any server.

In addition to supervised classification tasks, anomaly detection is a common use case for on-device training across various domains. For instance, a medical device attached to a specific person can continuously train a neural network using the person's medical data. If the network detects unexpected behavior, it can issue a warning, aiding in predicting medical events and enabling timely interventions. Anomaly detection is also prevalent in self-observing systems, such as those used in industrial environments. An MCU embedded in a smart sensor can continuously adapt a neural network based on current environmental conditions and alert changes in the behavior of machines or systems. This allows for timely detection of anomalies, such as data drifts, facilitating proactive maintenance.

In addition to privacy considerations, poor or unavailable network connections pose challenges. Considering a space probe traveling to distant planets. A one-way radio signal from Pluto to Earth takes at least 4 hours and 27 minutes, highlighting the latency issues associated with network connectivity. [48] Researchers at the University of Oxford conducted experiments involving on-device anomaly

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

detection trained on a satellite. Typically, satellite data is transmitted to Earth for processing, which can take hours and limit response times to rapidly evolving events like natural disasters. The researchers developed a model to detect changes in cloud cover from aerial images. The network consists of two parts. The first part compresses the newly-seen images and was trained on the ground. The second part decides whether the image contains clouds or not was trained directly on the satellite. In the future, the researchers intend to develop more advanced models that can automatically differentiate between flooding, fires, deforestation, and much more. [49]

In summary, on-device training offers advantages over conventional cloud-based approaches [50], resulting in smarter, more secure, and more efficient devices. Besides improved performance the benefits can be broadly categorized into privacy and network aspects:

- **Privacy**: By keeping data on the device and avoiding transmission to servers, the risk of privacy leaks is reduced. This is crucial for applications handling sensitive data like business secrets or personal information such as private photos.
- **Network Conditions**: On-device training eliminates the need to send data to servers and wait for responses, making the machine learning application independent of unstable network connectivity. This not only saves bandwidth and network resources but also helps applications to react in time. Some devices do not even have network access or operate on quantity-dependent network plans instead of flat rates. Here, sending data to a server is either not possible or directly results in increased costs.

2.4 Overview of Machine Learning Platforms

The machine learning approach introduced in this thesis, Back Propagation Layer Scheduling (BPLS), focuses on time and energy-efficient fine-tuning of neural networks. BPLS could have positive effects on fine-tuning in general. However, this thesis targets resource-constrained devices. Such devices are usually tiny computers embedded in larger systems including software, hardware, and mechanical parts. These computers, in general, have strict memory, power, energy, and size constraints and often have to fulfill real-time requirements [51]. Examples are computers embedded in cars, planes, or factories, but also digital watches and computers controlling washing machines can be considered as embedded devices. Executing or training neural networks on embedded devices is often referred to as embedded machine learning [52], a sub-category of on-device machine learning. This section provides a general overview of common machine learning platforms with a focus on embedded devices, such as MCUs and FPGAs.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Machine learning tasks, both inference and training, mostly consist of many vector and matrix multiplications and accumulations. Accordingly, computational platforms executing machine learning tasks should be able to perform those computations as fast and efficiently as possible. There are several platforms commonly used for machine learning. The most common are CPUs, GPUs, TPUs, FPGAs, ASICs, and MCUs. All of them bring advantages and disadvantages with them and can be found in different fields of application.

2.4.1 CPU (Central Processing Unit)

CPUs can be found inside every desktop computer, laptop, smartphone, MCU, and so on. As the name suggests, CPUs are responsible for the central control of the computational system. They commonly provide multiple cores, allowing parallel computing.

Modern CPUs often have additional instruction-set extensions that can speed up execution. Extensions accelerating vector computations, for example, are especially interesting for machine learning. In recent years, CPU extensions specialized in machine learning tasks have been introduced, further improving performance and efficiency. [53] However, the opportunity for large parallelism in CPUs is limited, which makes them only attractive for neural networks with small or medium-scale parallelism, for sparse neural networks, and in low-batch-size scenarios.

One further disadvantage of applying CPUs for machine learning tasks is the relatively low compute density compared to other options. CPUs simply require significantly more space for the same computing power, making them less attractive for use in data centers. However, especially when considering mid-spec on-device machine learning, for example, on smartphones or laptops, the actual computing capacity required is lower. Since CPUs are needed anyway for central control, it is convenient to use them for machine learning tasks as well.

A current consumer CPU is the Ryzen 7 7700. This 65W processor consists of 8 cores, each capable of running 2 threads. The base clock is 3.8GHz and can be boosted to 5.3GHz.

2.4.2 GPU (Graphics Processing Unit)

GPUs were originally designed as CPU accelerators for rendering videos and games. These applications primarily involve vector and matrix computations, similar to machine learning tasks. GPUs consist of a large number of simple processors operating in parallel, known as shaders. This parallel architecture gives GPUs an advantage over CPUs in terms of parallelism, leading to potentially higher computation speed, throughput, and lower energy consumption, depending on the task. [54]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

A current mid-range example is the Nvidia A10 Tensor Core GPU consisting of 9216 Cuda cores (shaders) clocked with 0.9GHz to 1.7GHz.

2.4.3 MCU (Microcontroller Unit)

Besides FPGAs, MCUs are the most common type of embedded computational systems. They usually combine one or several CPU cores, RAM, Flash Memory, I/O ports, and other peripherals on one chip. MCUs are usually embedded in larger systems, where they are responsible for controlling. Besides limited memory and computing resources, MCUs are often battery-powered and therefore rely on power constraints.

To discuss the resource limitations of MCUs in terms of available power and on-chip memory, a low-spec and mid-spec MCU is compared here. The Arduino Uno consists of an ATmega328P processor [55], operating at 16 MHz. It has 32 KB of flash memory and 2 KB of SRAM. The current consumption is 12 mA. On the other side, the STM32F2 [56] consists of an ARM Cortex-M3 processor operating at 120 MHz. It has 1 MB of flash memory, 128 KB of SRAM, and consumes 21 mA. When comparing these MCU specifications with typical CNN models like AlexNet (240 MB) [57] or VGG 16 (528 MB) [58], it can be seen that these networks exceed the amount of available memory by almost a factor of one-thousand.

The model size describes the number of bytes required to store all the parameters of the network. Considering training, even more memory is required due to storing of temporal data. Therefore, to run neural networks on MCUs, they usually have to be optimized. Some optimization techniques will be introduced later.

2.4.4 FPGA (Field Programmable Gate Array)

FPGAs are programmable circuits consisting of an array of logic blocks. Configurable interconnects allow blocks to be wired together. Different from ASICs, FPGAs can be adapted and updated at a later stage. This adaptability allows for versatile application and efficient power usage, as well as high computational density [59]. While FPGAs have gained popularity in data centers [60], they are also widely used in embedded applications [61]. However, compared to other platforms, implementing applications on FPGAs requires more effort.

Despite numerous machine learning frameworks available for CPUs, GPUs [62], and others for MCUs [63], there are relatively few targeting FPGAs. Additionally, most FPGA machine learning frameworks are restricted to generating hardware for inference [64], [65]. Only a small number supports on-device training as well [66].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.4.5 ASIC (Application Specific Integrated Circuit)

ASICs offer further optimization opportunities compared to FPGAs [67]. Unlike FPGAs, ASICs do not consist of programmable blocks and interconnects but of fixed circuits that cannot be altered after production. This puts even more importance on testing and evaluation during the design phase.

Major tech companies such as Qualcomm, Facebook, and Samsung are investing in the development of their own AI-specific ASICs [68] [69] [70]. However, one of the most prominent examples is Google's specialized AI hardware, known as the Tensor Processing Unit (TPU). TPUs are tailored for Google's machine learning framework TensorFlow [71] and can be compared to GPUs that only consist of matrix computation units.

2.5 Training on Embedded Devices

This thesis focuses on fine-tuning neural networks on resource-constrained devices. During training, peak memory usage is significantly higher than during inference. To prevent the systems from running out of memory and enable training on these platforms, optimizations are typically required.

The primary optimization techniques to adapt networks for resource-limited devices are Pruning (removing redundant parameters) [72] and Quantization [73] (reducing the number of bits used to represent model parameters). These techniques can significantly reduce the amount of required memory. Furthermore, certain training techniques commonly used on powerful computer systems are unsuitable for resource-limited devices.

This section outlines advanced training methods typically employed on servers, explains why they may not suffice for resource-constrained devices, and provides an overview of techniques to enhance machine learning on embedded devices. More details can be found at [74].

2.5.1 Training Optimizers

Training optimizers play a crucial role in modern machine learning, aiding in achieving higher test accuracy in fewer epochs by adjusting learning rates and gradients. However, optimizers typically require storing additional values for each weight of the network, making them less suitable for environments with memory limitations, such as embedded devices. PyTorch [62], for example, offers more than 10 different optimizers. Here, the most important concepts are explained briefly.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Momentum: The core concept involves employing an Exponential Moving Average (EMA) of gradients to update weights instead of directly applying the most recently computed gradient. In other words, it adds a percentage of the prior update vector to the current one. This approach helps to prevent stagnation in local minima or saddle points, accelerate convergence, and dampen oscillations. Accordingly, Momentum is widely used in machine learning and serves as the base for more advanced optimizers.

$$w_{t+1} = w_t - \alpha \nabla f(w_t) + \beta(v_t - \nabla f(w_t)) \tag{2.7}$$

In the equation, w_{t+1} represents the updated weights at time step $t+1$, w_t the current weights at time step t, and $\nabla f(w_t)$ the gradient of the loss function with respect to the weights at time step t. Besides that, α denotes the learning rate, β signifies the momentum parameter, and v_t represents a moving average of past gradients. The parameter β dictates the weight attributed to the last N data points. For instance, $\beta = 0.9$ averages the last 10 data points, while $\beta = 0.98$ considers the last 50 data points. According to the formula, newly computed gradients have a greater influence on the moving average than older ones. Averaging over a broad window leads to a slowly adapting average and consequently smoother loss curves, as depicted in Figure 2.5.

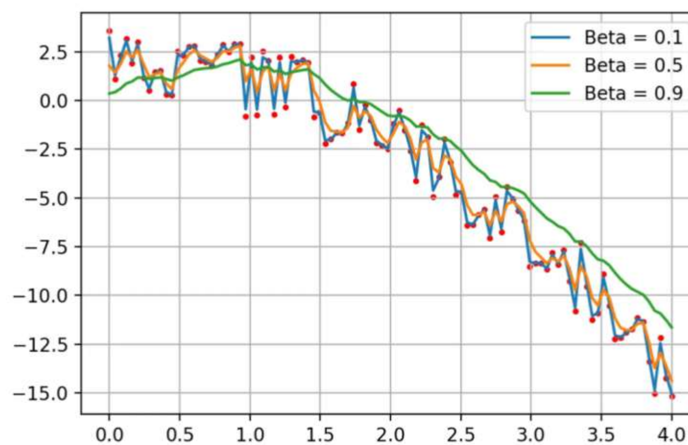


Figure 2.5: EMA with different β-values [4]

It would be insufficient to store the last n gradients. Fortunately, the Exponential Moving Average allows past gradients to be tracked by storing a single value. This makes this optimization relatively computationally inexpensive while enhancing training performance. However, storing an additional value for each trainable weight approximately doubles the size of the neural network. Consequently, this, along with most other optimization techniques, is inadequate for resource-constrained devices.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Adaptive Learning Rate: Determining suitable learning rates for training neural networks can be challenging. High learning rates may cause overshooting, while low ones slow down training. This issue can be mitigated by adapting the learning rate based on past gradients.

Optimizers like AdaGrad or Adadelta automatically adjust the learning rate. They decrease it for high gradients and increase it for low ones. These optimizers compute distinct learning rates for each trainable weight. This differentiation is necessary because sparse feature weights require higher learning rates than dense feature weights due to the lower frequency of sparse feature occurrences.

$$\eta_t = \frac{\eta}{\sqrt{\alpha_t}} \tag{2.8}$$

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\delta L}{\delta w_{t-1}}\right)^2 \tag{2.9}$$

The equations illustrate the behavior of AdaGrad. η_t denotes the current learning rate for a specific weight w. As the sum of previous gradient squares grows over time, the learning rate automatically decreases. Consequently, learning rates approach zero eventually, stopping the network from learning. Adadelta addresses this issue by averaging only the last n gradients instead of accumulating the entire gradient history. This can be efficiently implemented using the Exponential Moving Average.

Implementing Adaptive Learning Rates as described here demands significant memory, similar to Momentum. Alternative approaches with lower resource requirements, such as Cyclical learning rate schedules, are discussed later on.

ADAM: ADAM essentially combines the principles of Momentum and Adadelta, making it highly effective and well-suited for a wide range of problems. Consequently, ADAM stands out as one of the most favored optimizers nowadays.

However, ADAM requires storing both the Exponential Moving Average for past gradients (used in Momentum) and the past squared gradients (utilized in Adadelta). Accordingly, ADAM leads to relatively high memory requirements, making it unsuitable for resource-constrained devices.

2.5.2 Learning Rate Scheduler

As mentioned, dynamic learning rate adjustments generally lead to good training behavior but are relatively memory and computationally expensive. As an alternative, static learning rates or learning rate schedules can be applied.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

In general, changing the learning rate over time by following a learning rate schedule can be beneficial. One approach is to decrease the learning rate when the loss stops improving, known as the Plateau Learning Rate Schedule. Alternatively, the Exponential Learning Rate Decay gradually reduces the learning rate over time. Figure 2.6 compares the training performance of different learning rate schedules. As can be seen, using a static learning rate does not perform significantly worse. However, the decreased learning rates in the later stages lead to a more stable curve.

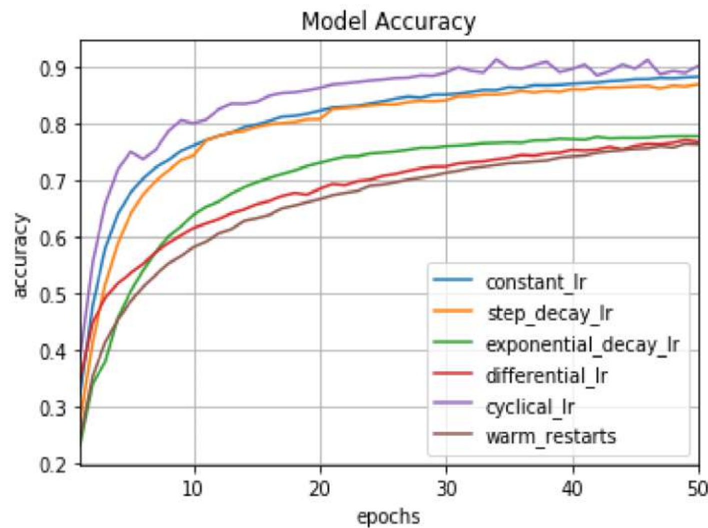


Figure 2.6: Comparison of different learning rate schedules [5]

Especially considering pre-training, relatively high learning rates in early training stages can be beneficial because random weights are far from optimal. During training, the learning rate can decrease to allow more fine-grained weight updates. Good training behavior can be achieved by selecting the learning rate with the highest decrease in loss. [75] Increasing the learning rate until a certain point decreases the achieved error and improves the training process accordingly. However, increasing the learning rate further beyond a certain point will cause the training to start diverging.

Resetting the learning rate periodically, as illustrated in Figure 2.7, can improve training performance even more. According to [6], this approach can perform as well as Adaptive Learning Rate methods, but with practically no computational expenses.

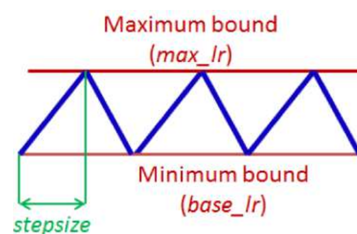


Figure 2.7: Cyclical Learning Rate Adjustment [6]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.5.3 Pruning

One highly effective technique for optimizing neural networks on resource-restricted devices is Pruning, also referred to as Sparsification. It relates not just to one single method, but rather to a category of optimization algorithms. The fundamental idea behind pruning is to eliminate parameters that do not significantly contribute to the network's performance. This not only speeds up inference during network execution but also accelerates training by skipping unnecessary gradient computations and weight updates.

Pruning typically is an iterative process. Initially, the entire network is trained and evaluated for accuracy on test data. Subsequently, weights or even entire neurons are removed following various approaches [76]. While the accuracy may decrease during pruning, fine-tuning can often recover the network's performance, if the pruning was not too aggressive. This process of training, evaluation, and pruning can continue until a sufficient performing reduced version of the network is achieved.

Pruning typically leads to a trade-off between model performance and efficiency. Drastically reducing the network's size decreases memory requirements and computation time but may also impact accuracy. Moreover, pruned networks can sometimes generalize as effectively as, or even better than, the original dense networks. [77]

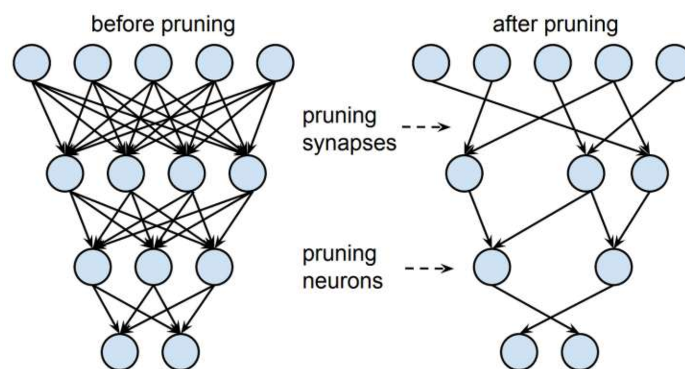


Figure 2.8: Network before and after pruning [7]

There are various methods for pruning neural networks. One approach is to individually set weights to zero, effectively removing them from further computations while maintaining the network's architecture. Alternatively, entire nodes can be removed, altering the network's structure. Determining which nodes to prune without significantly impacting performance involves various strategies. For instance, neurons can be ranked based on their influence on the network's output. Neurons that frequently output values around zero or are rarely activated play no significant role in the model's task and may therefore be candidates for pruning. Another approach involves removing weights that are close to zero. This can be achieved by removing all weights below a certain threshold. Further, if sev-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

eral neurons behave very similarly by having very similar weights and activations, only one of these neurons is required, while the rest represents redundancies that can be pruned.

In addition to pruning, there are other techniques for reducing network size. First of all, choosing smaller networks that match the task requirements is advisable. For instance, there are several versions of the VGG network [78] differing in depth, such as VGG13, VGG16, and VGG19. Furthermore, reducing the size of input images results in smaller feature maps and reduced memory requirements, although it can affect the performance of certain tasks like object detection. [79] While pruning involves removing parameters, Sparse Updates involve updating only specific parameters of the network during training while freezing others. This technique, which can include freezing individual parameters or entire layers during fine-tuning, helps reduce peak memory usage and speed up training, making it common in resource-constrained environments. Sparse Updates is also the main technique utilized in the approach introduced in this thesis, BPLS.

2.5.4 Quantization

Quantization is another technique for enabling efficient machine learning on small devices. Typically, neural networks are trained using full-precision floating-point values. However, approximating these values with low bit-width numbers, such as 8-bit or 16-bit fixed-point, can substantially reduce memory requirements and computational costs for executing and training neural networks, albeit with some loss in accuracy. [80]

Figure 2.9 illustrates the relationship between network accuracy and the power consumed by Multiplication and Accumulation (MAC) operations. Higher bit-widths generally lead to increased power consumption. Worth mentioning, the reduction from 32-bit floating-point to 32-bit fixed-point or 16-bit floating-point does not negatively affect network accuracy in the illustrated example.

Neural networks can initially be trained with full precision and then quantized afterward. In most cases, quantization results in inferior-performing networks, necessitating fine-tuning to improve the accuracy of the quantized network. [81] An alternative approach involves quantizing during training, known as Quantization-Aware Training (QAT) [82]. This method typically yields better-performing networks. QAT involves using two separate networks for training: one operating on floating-point (continuous range) and the other on low-bit fixed-point precision (discrete range). The fixed-precision network is updated in the forward pass using the float-precision, while the float-precision network is updated in the backward pass using the fixed-precision. Ultimately, only the fixed-precision network, with reduced size is applied for on-device operation. This training process is more complex and costly than traditional training methods. [83]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

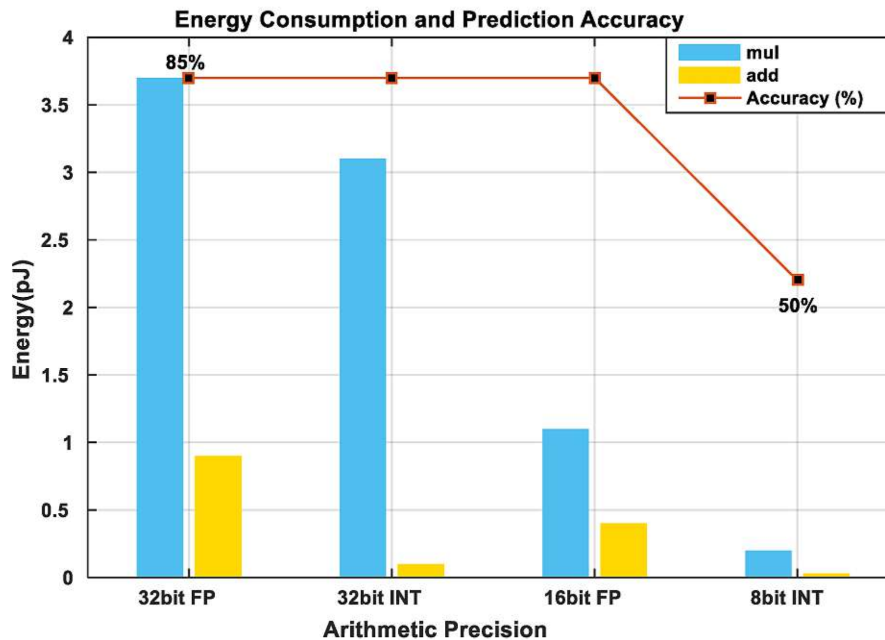


Figure 2.9: *Energy consumption and prediction accuracy of a DNN as a function of the Arithmetic Precision adopted for a single MAC unit in a 45 nm CMOS.* [8]

An extreme case of quantization is Binary Neural Networks, where values are restricted to binary states (+1 or -1). BinaryConnect [84], for example, maps float-values to binary values by simply taking the sign of the float-value. This approach can yield significant advantages, particularly with specialized hardware, by replacing many MAC operations with simple, less time, and power-intensive accumulations. BinaryConnect focuses not only on quantized inference but also on training networks with binary weights.

Apart from quantizing weights, input values of layers can also be quantized. This involves replacing the activation function with a quantization function, converting the layer output to low bit-width. [85] Per-layer scaling factors can then be applied to align quantized values with float values. The special case of binary activations can degrade network accuracy, typically more than the binary quantization of weights. [86]

The input and output layers of neural networks typically perform fewer operations and may suffer more from quantization than other layers. Hence, it is common to compute these layers in full-precision floating-point to maintain higher accuracy without significant speed loss, often termed Mixed Precision. [87]

In addition to Quantization, there is Dequantization as well. [14] Dequantization converts quantized parameters and activations back to high-precision continuous values typically by dividing the quantized values by their scale factor. Applying dequantization helps mitigate the accuracy loss caused by

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

quantization while still achieving memory savings.

2.5.5 Small Batch Sizes

As mentioned, the training data can be passed to the network in several batches. The gradients for each data sample inside the batch are averaged. These averaged gradients are then used to update the weights and biases.

One reason high batch sizes are favored for training is that they allow the computation of several data samples (elements of the batch) simultaneously. This accelerates training significantly when using devices such as GPUs or TPUs, at the cost of increased memory requirements.

On devices with limited memory and computational capabilities, batch elements cannot usually be computed in parallel. However, by sequentially processing one element of the batch after another and performing the network update after the batch is finished, arbitrary batch sizes can be achieved without the need to keep every data sample and corresponding gradients in memory. Although this sequential processing of data samples does not improve training speed with larger batch sizes, it can result in smoother convergence.

Conversely, smaller batch sizes typically introduce more variation between updates, which can have a regularizing effect, potentially leading to better-performing networks. Moreover, smaller batch sizes result in more frequent network updates, which can be advantageous when considering real-time conditions. Accordingly, smaller batch sizes are more common when training on resource-limited devices.

2.5.6 Hardware optimization

The previously mentioned optimization techniques all focused on training or model storage. However, there are additional techniques that focus on classical hardware utilization, such as recomputing and paging.

Recomputation reduces memory requirements by deleting intermediate layer outputs. The corresponding values do not need to be held in memory and are simply computed when needed. While this technique reduces peak memory usage during training, it increases the number of required computations.

Paging, on the other hand, is a common practice in Operating System. The OS shifts currently unneeded data to secondary storage. This technique can be utilized during training to reduce peak memory by paging out the unneeded intermediate activations.

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.6 Analyzing and Comparing Neural Networks

The simplest way to analyze the training behavior of neural networks is by investigating accuracy and loss. This allows to see how a model improves during training, but it does not reveal the underlying decision-making processes.

When evaluating new training approaches, it is important to compare them with meaningful baselines. Common baselines include the ideal baseline, also known as the cloud baseline, which describes training with sufficient hardware resources and typically yields the best training accuracy. The ideal baseline can be used to illustrate the potential accuracy gap and reduction in resource consumption of new training approaches. Besides that, it is common to use unoptimized training models or pre-trained models without further fine-tuning as baselines.

2.6.1 Vector Analysis

Neural networks consist of millions of weights, each trainable layer having its own. The weights of a layer can be considered as vector. Analyzing the magnitude and orientation of these vectors can be useful for understanding networks and their training behavior.

The magnitude can be computed using metrics such as the Euclidean or Middle Distance, but also by less common concepts like the Frobenius Inner Product. In fine-tuning, this analysis can reveal the difference between the pre-trained base model and the final fine-tuned one. The magnitude indicates how strongly the weights were adjusted during training, allowing for the evaluation of which layers were modified more and which less during fine-tuning. For instance, the classifier layer in most scenarios will show a relatively high magnitude, while layers closer to the input will exhibit lower magnitudes.

Additionally, the distance between layers of models trained with different hyperparameters can be analyzed to assess the similarity of different training approaches. The following formulas demonstrate how to compute the Euclidean and Middle Distance. Here, $\vec{w}_i$ and $\vec{w}_j$ represent the weights of different configurations of the same layer, and n represents the number of weights.

$$D_{\text{euclidic } i,j} := \sqrt{\sum(\vec{w}_j - \vec{w}_i)} \tag{2.10}$$

$$D_{\text{middle } i,j} := \sqrt{\frac{1}{n}\sum(\vec{w}_j - \vec{w}_i)} \tag{2.11}$$

Another useful metric is Cosine Similarity, which measures the similarity of two vectors by considering their orientation or direction while ignoring their magnitude. The output of Cosine Similarity is a single

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

value in the range of [-1, 1], where 1 indicates very similar vectors, 0 indicates very different ones, and -1 indicates opposite vectors with an angle of 180 degrees between them. Cosine Similarity can only be computed if the scalar product of the two vectors is defined, meaning that both vectors are part of the same inner product space. Cosine Similarity is computed using the following formulas, where investigated network configurations are denoted by i and j. Here, $\vec{w}_u$ represents the weights of these configurations, and $\vec{w}_{start}$ represents the configuration from which fine-tuning began.

$$\vec{w}'_u = \vec{w}_u - \vec{w}_{\text{start}} \tag{2.12}$$

$$S_{\cos i,j} := \frac{\langle \vec{w}'_i, \vec{w}'_j \rangle}{\|\vec{w}'_i\| \|\vec{w}'_j\|} \tag{2.13}$$

2.6.2 Visualization of Networks and Training Processes

Neural networks typically consist of a large number of neurons making it impractical to plot all neuron activities individually. Instead, Dimensionality Reduction tools are commonly used to create human-readable plots spanning 2 to 3 dimensions. Common tools for this purpose include UMAP (Uniform Manifold Approximation and Projection) [88] and t-SNE [89]. Both tools allow to approximate the behavior of neural networks, while UMAP is the one utilized in this work.

For example, a 2D map can be generated where all weights of one layer are represented as a single point. Plotting different epochs in a row allows to observe how the network changes during training. This is particularly useful when comparing different training runs. The absolute position of the points in the Dimensionality Reduction plot does not carry any information. It is the relative position that matters. Accordingly, points plotted close to each other indicate relatively similar weights.

Dimensionality Reduction tools can also be used to analyze the output of the network or individual layers. For example, in image recognition tasks, it can be helpful to visualize which input images the network considers similar, thus providing insights into the network's decisions. Figure 2.10 illustrates this using UMAP, where the background color represents the category chosen by the network for specific input images, with wrongly categorized images outlined in red. UMAP automatically clusters the images based on the activation behavior of the network. As shown in the example, apples are well isolated, while the network exhibits similar behavior when processing images of ants and plains.

In addition to UMAP, Euclidean Distance, and Cosine Similarity, there are other more complex ways of analyzing training behavior, such as DeepTracker [9] or TensorView [90]. However, these methods are not utilized in this thesis.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

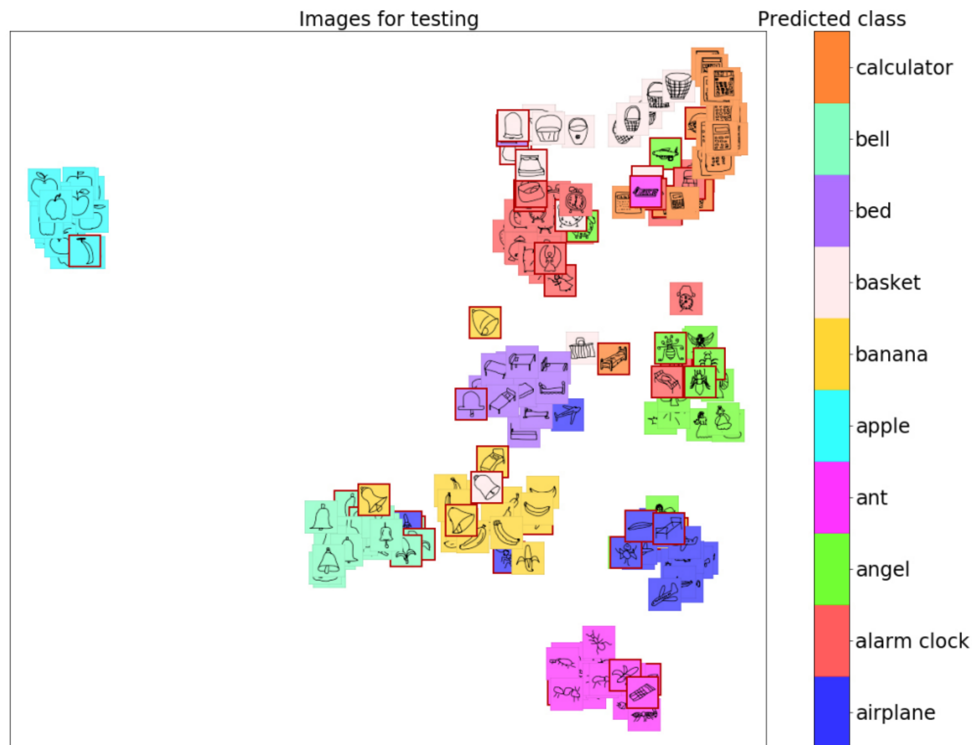


Figure 2.10: UMAP - clustering of image classification [9]

2.6.3 Execution Time and Power Consumption

Especially in the field of on-device machine learning, execution time and power consumption play important roles. Even though both metrics can be measured independently, they are directly influenced by the number of operations executed. Therefore, by computing the number of operations, the execution time and power consumption can be estimated. Fortunately, at its core, training, and inference of neural networks, at least for basic CNNs, is relatively simple. The required multiplications and accumulations (MACs) can easily be derived.

Table 2.1 shows the number of MAC operations required for processing the forward propagation of a single data sample (batch size 1) for certain layers of a CNN. Here, c_{in} and c_{out} represent the number of input and output channels, k_{size} represents the dimensions of filter kernels used in Convolutional Layers, pad the number of padding elements, i_{size} the size of the input feature map, and o_{size} the size of the output feature map. Depending on the input dimension (1D signals or 2D images), k_{size}, i_{size} and o_{size} consist of one or two values each.

Backward propagation is more complex than forward propagation. To perform backward propagation, three types of computations have to be executed:

- **Weight Gradient**: Computing the gradients of the weights.
- **Weight Update**: Applying the Weight Gradient to modify the network parameters.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Layer Type	Number of MACs
Convolutional	$c_{in} \cdot c_{out} \cdot (k_{size} \cdot (i_{size} - 2 \cdot pad) + 2 \cdot (pad \cdot k_{size} \cdot pad^2 + \frac{pad^2 - pad}{2})) + c_{out} \cdot o_{size}$
Linear	$c_{in} \cdot c_{out} + c_{out}$
ReLU	$c_{in} \cdot i_{size}$
Max Pool	$c_{in} \cdot i_{size}$
Gl-Avrg-Pool.	$c_{in} \cdot i_{size}$

Table 2.1: Number of MAC operations - Forward Path

Layer Type	Number of MACs
Conv-Input-Grad.	$c_{in} \cdot c_{out} \cdot (k_{size} \cdot (i_{size} - 2 \cdot pad) + 2 \cdot (pad \cdot k_{size} \cdot pad^2 + \frac{pad^2 - pad}{2}))$
Conv-Weight-Grad.	$c_{in} \cdot c_{out} \cdot (1 + 2 \cdot (pad \cdot o_{size} \cdot pad^2 + \frac{pad^2 - pad}{2}))$
Conv-Update	$c_{in} \cdot k_{size} \cdot c_{out} + c_{out}$
Lin-Input-Grad.	$c_{in} \cdot c_{out}$
Lin-Weight-Grad.	$c_{in} \cdot c_{out}$
Lin-Update	$c_{in} \cdot c_{out} + c_{out}$
ReLU	0
Max-Pool.	$c_{in} \cdot i_{size}$
Gl-Avrg-Pool.	$c_{in} \cdot i_{size}$

Table 2.2: Number of MAC operations - Backward Path

- **Input Gradient**: Propagating the loss to the next layer.

Depending on the position of the layer and whether the layer is trained or frozen, some of these computations can be skipped:

- If the layer is trained and the last in the Back Propagation chain, only Weight Gradients and Weight Updates need to be determined. The Input Gradient can be left out.
- If the layer is not trained but not the last in the Back Propagation chain, the Input Gradient has to be computed, while Weight Gradients and Weight Updates can be left out.

Table 2.2 lists the number of MAC operations required for processing the backward propagation of a single data sample.

While the mathematical model introduced here allows for estimating the execution time and power consumption, in reality, there are additional aspects that should be considered:

- **Memory Management**: The mathematical model completely ignores memory management. Memory operations typically consume more time and energy than actual computations. They are complex and hard to predict, particularly on computers running multiple tasks simultaneously. Factors such as process scheduling and caching contribute to increased variability in execution time. Typically, these factors do not apply when executing code on MCUs. MCUs rarely run

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Layer Type	Memory Requirements
Input-Feature-Map	$c_{in} \cdot i_{size}$
Input-Gradient	$c_{in} \cdot i_{size}$
Conv-Weight-Grad.	$c_{in} \cdot k_{size} \cdot c_{out} + c_{out}$
Lin-Weight-Grad.	$c_{in} \cdot c_{out} + c_{out}$

Table 2.3: Memory requirements of different layers

multiple tasks simultaneously and often have no Operating System. Accordingly, measuring timing behavior on MCUs can lead to clear results with little or no variance.

- **Machine Learning Frameworks**: Advanced frameworks like PyTorch [62] or TensorFlow [71] are typically used for training neural networks. Even though these frameworks may not bypass the basic operations of inference and training, they apply several optimizations. One reason why Pytorch is that efficient, despite providing a user friendly Python interface is that underneath fast CPP code is executed. This CPP code has to be called and initialized, affecting execution time. Therefore, accurately estimating the execution time for machine learning tasks performed with these frameworks would require an adaptation of the mathematical model.

2.6.4 Memory requirements

Besides training performance, execution time, and energy consumption, memory requirements are very important metrics when investigating neural networks and training behavior, especially when executing or training on resource-constrained devices. While the evaluation of BPLS in regard to memory requirements is not that deeply explored in this thesis, basic concepts are introduced here.

Similar to the earlier introduced timing estimations, at its core, the peak memory requirements of training neural networks can be easily determined. Basically, the data to store consists of Input Feature Maps, Weight Gradients, and Input Gradients, where the size of an Input Gradient matches the size of the corresponding Input Feature Map. Combining the number of values to store with the corresponding data type leads to the peak memory requirements for training. Table 2.3 introduces a simple model for estimating the memory requirements for various layer types.

Depending on whether a layer is trained or not, the amount of data required to store differs. Storing the Weight Gradients for a layer is only required if the layer gets updated. Storing the Input Gradient of a layer is only required if the loss has to be propagated to the subsequent layer.

Considering the execution or training of neural networks under real conditions, things become increasingly complex, especially when considering memory optimizations such as dynamic memory reuse.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 3

Related Work

The most common techniques for improving training efficiency on devices relying on resource constraints are the previously introduced pruning and quantization methods. Even when BPLS and those techniques might be used together in future, BPLS follows a completely different approach. It is closer related to work focusing on fine-tuning neural networks, regardless the target device.

This section is split into three parts. First, current ideas for efficient training on devices relying on resource constraints are introduced. Afterwards, ideas for analyzing fine-tuning behavior and static fine-tuning optimization algorithms are touched. Finally, algorithms for selecting layers for fine-tuning dynamically depending on the training state and training data are introduced.

3.1 Embedded Training

As mentioned earlier, techniques for improving training efficiency on devices relying on resource constraints are mostly based around pruning and quantization or add further layers to the network. Some methods relying on those techniques are MiniLearn [14], TinyOL [15] and TinyTL [91]. More complex methods, such as TTE [92] and POET [16] have an additional preparation stage, where a computing graph and training plan is generated before training starts. The plan schedules when and how to use specific techniques during training. However, applying predefined schedules often leads to insufficient training behavior. Therefore, it can be beneficial to adapt the schedules during run-time. Here, just a short summary is provided. For further information feel free to take a look at [93].

MiniLearn [14] improves re-training of Convolutional Neural Networkss (CNNs) on resource-constrained devices by storing the weights and intermediate outputs in integer-precision and dequantize them to floating-point during training. The process can be split into three steps:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

- **Dequantizing and pruning filters**: While the first layer of the network is kept in integer-precision to retain the input shape, the others are dequantized to floating-point format. Next, during training filters with small L1 norms are considered less important and pruned. Finally, depending on the specific task, the number of neurons in the output layer are adjusted to fit the number of output classes.
- **Training filters**: The quantized output of the first layer is converted to floating-point during training and further used for training.
- **Fine-tuning Linear Layers**: The pruned filters get converted back to integer precision and frozen. Further, the Linear Layers are fine-tuned to compensate potential loss of information caused by pruning and quantization.

It's also quite common to add new specialized layers to the network and restrict training on this new layers, just as **TinyOL** [15] does by adding a TinyOL layer on top or by replacing an old one.

TinyTL [91] on the other hand only tunes the biases of the network while freezing the rest. Compared to classical pruning approaches, this allows to reduce the memory footprint even further. However, only tuning the biases can lead to limited generalization ability. To address this issue, TinyTL adds the so called lite residual module to the network, consisting of group convolution [94] and a 2×2 average pooling. This helps TinyTL to save memory without losing accuracy.

TTE [92] proposes the so called Quantization-Aware Scaling (QAS). The key idea of QAS is to multiply the square of scaling factors corresponding to precision with intermittent output to relieve the disorder caused by quantization and leading to low convergence accuracy. Further, TTE combines QAS and sparse update techniques to reduce memory usage. The preperation of TTE can be split into three parts. Firstly, TTE generates a static backward computing graph. Secondly, TTE prunes away the gradient nodes with the sparse layer update method. Thirdly, TTE reorders the operators to immediately apply the gradient update to a specific tensor before back propagating to earlier layers, so that the memory occupied by the gradient tensors can be released as soon as possible. Afterwards, training is executed according to the computation graph.

POET [16] consists of two stages. In the first stage, POET analyzes the memory and computation costs for training a given neural network and selects a suitable technology to optimize them during training. Besides that POET takes hardware constraints into account by searching for the optimal schedule of paging [95] and re-materialization [96]. Afterwards, the actual training is performed according to the previously defined schedule.

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3.2 Static Layer Freezing

When fine-tuning neural networks it is common to freeze certain layers closer to the input. However, the amount of layers to freeze and the learning rate relation of the remaining layers is hard to define. They depend on multiple aspects just as the specific neural network, the applied pre-training and the data used for fine-tuning.

AutoLR [97] for example defines an algorithm that automatically finds good fitting learning rates for each layer after freezing layers that do not contribute to the fine-tuning process.

Besides that Barakat and Huang [98] present a way for finding out the block of layers most relevant to the target data. This information is then used to freeze all other layers, except the classifier layer. The method starts with adapting the weights of each layer to search for the layer leading to salient improvement when fine-tuned. For that a small part of the training data (e.g. 10%) is used. Instead of analyzing separate layers, layers can be combined to blocks at the beginning. Those blocks can be defined by dividing the network into several parts. The divisions can be done by non-trainable layers, such as Max-Pooling Layers for example.

DEFT [99] as well deals with finding and selecting layers of a given CNN best to fine-tune. The method consists of two components. Firstly, layers are selected. Secondly, the selected layers are fine-tuned and the performance of the network analyzed. For evaluation the output loss is used to compute the so called performance evaluation score (fitness value). This fitness value is then passed back to the layer selection mechanism. The process reiterates, trying to minimize the fitness value, until the maximum number of model evaluations, a manually set parameter is reached. The best performing selection of layers is then used for the actual fine-tuning of the network.

LISA [100] was published during the final stages of this thesis and follows a relatively similar idea as BPLS. It focuses on fine-tuning Large Language Models by randomly freezing layers according to their probability. While the first and last layer of the network stays permanently unfrozen, the middle layers are frozen with probability $\frac{\gamma}{N_L}$, where N_L represents the number of layers and γ controls the amount of unfrozen layers. This allows to speed up training and reduce GPU memory consumption. In addition, it was shown that LISA has a regularization effect.

Besides their similar base idea, BPLS can be clearly distinguished from LISA. By freezing layers (skipping training steps) according to a pre-defined schedule, BPLS can not just reduce execution time and peak memory, but the maximum of operations per training step can be limited as well. This makes BPLS especially for real time applications more suitable.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.3 Dynamic Layer Freezing

While static fine-tuning methods have proven that they can achieve better performance then conventional training, they are not able to adapt to changes automatically. Some approaches were developed to overcome this issue.

During fine-tuning, **AutoFreeze** [101] adaptively determines layers that can be frozen. Once layers are frozen, the backward computation for those layers can be avoided. The approach promises to improve performance without affecting accuracy, different to the conventional approach, where the number of layers to freeze is determined before training starts.

Therefore AutoFreeze proposed an online algorithm based on the SVCCA metric [102] that can decide which layers should be frozen and when. The algorithm ranks layers by their rate of change and selects the slowest changing layers, where all the previous layers are frozen, for freezing. While some layers converge fast, others take significantly longer. The idea behind AutoFreeze is to freeze layers as they converge.

SLIMFIT [103] follows a similar approach as AutoFreeze but focuses on reducing memory requirements. It dynamically freezes less-contributory layers during fine-tuning and adopts quantization and pruning to minimize the memory footprint of static activations. Static activations refer to those that cannot be discarded regardless of freezing.

To decide which layers to freeze, layers are first ranked based on their distance values (change over two subsequent iterations) at each training iteration. Those with small distance values are kept frozen according to the freezing rate. The freezing rate represents an Hyper Parameter and can be chosen based on the on-device GPU memory budget. High freezing rates lead to high memory savings while affecting the accuracy. Low freezing rates usually provide high accuracy by saving less memory.

The intuition of SLIMFIT is that layers with small distance values are less contributory to the fine-tuning process as their parameters are not being updated much. On the other hand, the layers with large distance values are learning task-specific patterns by making more significant adjustments to their parameters.

Wanjiku, Nderu and Kimwele [104] proposed the use of Kullback – Leibler Divergence (DKL) on the weights cosine similarity distributions to select layers for fine-tuning dynamically. The DKL shows how far a specific distribution of one layer is from a distribution of another layer. This evaluation ensures that the selected layers contain weights that lead to the slightest generalization error in the learning process of the model. The layers with the lowest DKL terms are finally selected as candidates

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

for fine-tuning. As mentioned in the Background chapter, there are several ways for computing the similarity between two weights, such as the Euclidean Distance for example. However, [104] applies the Cosine Similarity due to it's relatively low time complexity.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Chapter 4

BPLS

According to [97], during fine-tuning, layers closer to the output require more adaptation compared to those closer to the input. Therefore, layers closer to the output are typically updated with higher learning rates. The idea behind BPLS is as follows: instead of applying different learning rates to different layers, updates for layers that require less adaptation are simply skipped. This is based on the hypothesis that skipping updates from certain layers behave similar as applying the corresponding fraction of the base learning rate to the specific layer. If this holds true, BPLS has the potential to accelerate training, conserve energy, and even reduce memory requirements.

4.1 Base Idea

The base idea of BPLS derives from the concept that assigning different learning rates to different layers during fine-tuning can enhance training performance. While this typically does not result in higher test accuracy, it allows the network to be trained in fewer epochs compared to using the same learning rate for all layers.

During fine-tuning, layers near the input are typically frozen, while those closer to the output are trained. This is because, in CNNs, layers near the input primarily extract general information, such as basic shapes. These layers generally do not need to learn new information during fine-tuning.

When using the same learning rate for all unfrozen layers, the learning rate is constrained by the layers that require less adaptation. As a result, layers closer to the output, which require stronger adaptation, are limited to lower learning rates. While training can still be effective in this scenario, it often requires many epochs due to the low learning rate.

In contrast, when learning rates are assigned to layers individually, layers requiring stronger adaptation can be assigned higher learning rates, while layers requiring less adaptation can receive lower learning rates. This approach allows each layer to train optimally and typically results in effective training outcomes in fewer epochs.

The core concept of BPLS builds upon the Per-Layer Learning Rate approach but focuses on adjusting the update frequency instead of the learning rate. By skipping updates of certain layers that require less adaptation, BPLS aims to save operations, energy, and time. Depending on the update order, memory savings can be achieved as well.

Layers where training steps are skipped only have access to a fraction of the training dataset. For instance, a layer that is updated every second training step has access to half of the training data, while a layer updated every fourth training step only has access to 25% of it. To address this issue, dataset shuffling or shifting can be applied.

Formula 4.1 illustrates a computationally inexpensive shifting approach utilized in this thesis. In this formula, s represents the number of indexes shifted each training step, b denotes the applied batch size, and e represents the number of epochs after the entire dataset has been shifted completely. After e epochs, the shifting process restarts from the beginning.

$$s = \left\lceil \frac{b}{e} \right\rceil \tag{4.1}$$

4.2 Definition

Tables 4.1 and 4.2 show the so-called schedules. A BPLS-schedule basically describes at which training step which layer is updated at which learning rate. Each training step represents the processing of one batch of training data. After executing the last step of the schedule (in the provided example step 4), the schedule restarts at step 1.

In the tables, the layers of the network are listed from top (output) to bottom (input). n denotes the relative learning rate, where $n = 1$ represents the base learning rate. $n = 0.5$ represents half, and $n = 0.25$ quarter of it. The rightmost column shows the average learning rate, calculated by summing up the learning rates of each training step and dividing it by the number of training steps within the schedule.

Cells containing a "-" are not updated during the specific training step. If all steps are marked as "-", the layer is frozen. The bottommost row shows the estimated number of operations for each training

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Per-Layer Learning Rate					
	Step 1	Step 2	Step 3	Step 4	mean
Classifier	n = 1	n = 1	n = 1	n = 1	n = 1
Conv5	n = 1	n = 1	n = 1	n = 1	n = 1
Conv4	n = 0.5	n = 0.5	n = 0.5	n = 0.5	n = 0.5
Conv3	n = 0.25	n = 0.25	n = 0.25	n = 0.25	n = 0.25
Conv2	-	-	-	-	-
Conv1	-	-	-	-	-
kMACs per step	8229	8229	8229	8229	8229

Table 4.1: Example of Per-Layer Learning Rate training configuration

BPLS					
	Step 1	Step 2	Step 3	Step 4	mean
Classifier	n = 1	n = 1	n = 1	n = 1	n = 1
Conv5	n = 1	n = 1	n = 1	n = 1	n = 1
Conv4	n = 1	-	n = 1	-	n = 0.5
Conv3	n = 1	-	-	-	n = 0.25
Conv2	-	-	-	-	-
Conv1	-	-	-	-	-
kMACs per step	8229	2055	5052	2055	4348

Table 4.2: Example of BPLS-schedule

step and their average. More information about the number of operations will be provided later.

The upper schedule represents the classical Per-Layer Learning Rate approach, where all unfrozen layers are trained in each training step with specific assigned learning rates for each layer. The second schedule shows an example of BPLS. Assumed the hypothesis holds true, on average, both the Per-Layer Learning Rate approach and the BPLS-schedule have the same learning rates applied. In addition, the estimated number of training operations required by the BPLS-schedule is significantly lower. While the Per-Layer Learning Rate approach executes 8229 kMACs per training step, the BPLS-schedule only executes 4348 kMACs on average. This represents a theoretical reduction in backpropagation time of 47%. Considering forward propagation as well, this leads to a theoretical speedup of 23%. This reduction in the number of executed operations not only affects the execution time but also reduces the energy requirements.

The reason updates of layers closer to the input of the neural network can be skipped is that these layers typically require less adaptation compared to layers closer to the output. However, this assumption relies on the network having been previously trained to solve related tasks. Accordingly, BPLS is specifically designed as a fine-tuning optimizer rather than for training randomly initialized networks. Subsequently, additional essential aspects of BPLS, including potential use cases and the notation format applied, are discussed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.3 Schedules

Besides being a variant of Per-Layer Learning Rate with improved energy and time performance, BPLS can also optimize training for specific cases depending on the applied schedule. When defining a schedule, an optimization metric can be chosen, such as Accuracy, Number of Operations, Training Time, Power Consumption, or Peak Memory.

Finding the best performing schedule for unknown training data can be challenging, a common issue in fine-tuning neural networks. Similar to the approach typically used with Per-Layer Learning Rate assignments, heuristics can be applied to define schedules that are more or less suitable. For example, just as the learning rate decreases from the output layer to the input layer in Per-Layer Learning Rate approaches, the number of skipped updates can increase accordingly. Frozen layers represent a learning rate of 0 or the skipping of all updates for that specific layer.

In addition to focusing on high accuracy and training speed, schedules can also be defined with a focus on resource limitations. Therefore, the simple mathematical timing model introduced in the Background chapter can be utilized. For example, a maximum number of operations for single training steps, epochs, or complete training can be specified. Based on this limit, layer updates can be scheduled.

This can further be used to define an energy budget for training. Since the energy consumption during training correlates with the number of operations executed, this energy optimization problem can be translated into limiting operations. For example, consider a device powered by a solar cell with a limited amount of energy available per day. This device performs several tasks, including the classification of images captured by the device using a neural network. To improve the network accuracy, fine-tuning is applied. Given that all tasks executed by the device require energy, only a small amount should be allocated for the training process.

Besides energy consumption, timing behavior can also be a focus. Execution time can be considered as a combination of the number of operations to execute and the amount of time each operation requires on average. This can be used to temporally align the training with other processes running on the device, such as communication tasks or to meet deadlines in Real-Time Systems. For example, if the device receives new training data every few seconds, the schedule can be designed to process all of the current training data in time before the new batch arrives.

In addition to reducing the number of operations required for training, peak memory usage can be reduced by applying specific schedules. Those schedules avoid training all unfrozen layers within the same training step. This also reduces the peak number of operations. Table 4.3 illustrates a corresponding schedule.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Memory Optimization					
	Step 1	Step 2	Step 3	Step 4	mean
Classifier	n = 1	n = 1	n = 1	n = 1	n = 1
Conv5	n = 1	n = 1	n = 1	n = 1	n = 1
Conv4	n = 1	-	n = 1	-	n = 0.5
Conv3	-	n = 1	-	n = 1	n = 0.5
Conv2	-	-	-	-	-
Conv1	-	-	-	-	-
kMACs per step	5052	7215	5052	7215	7215 (max)
peak memory [kB]	1061	1072	1061	1072	1072 (max)

Table 4.3: Example of memory optimizing BPLS-schedule (peak memory reduction = 6.38%, peak operations reduction = 12.32%)

4.4 Notation

To better understand the upcoming chapters, it is important to closely examine the notation used in BPLS-schedules.

Schedule names, such as *4421* represent the layers from output to input. The leftmost digit corresponds to the output layer and determines the schedule length. In the example of *4421*, the schedule cycle consists of 4 training steps. The subsequent digits show how many times the corresponding layer is updated within one schedule cycle.

Based on the BPLS notation scheme, other training approaches can be described by adding suffixes. Those other training approaches will later be used to evaluate BPLS.

- **4444** (Conventional): All unfrozen layers are trained with the same learning rate in each training step. This represents the most basic training approach.
- **4444_lrf4421** (Per-Layer Learning Rate): Each layer is assigned a specific learning rate independently, such that the average learning rate of each layer matches that of the corresponding BPLS-schedule (*4421* in this example).
- **4421_lrf4444** (Upscaled): BPLS, but the learning rate of each layer is independently scaled by factor n, where $n = \frac{T}{s}$. T is the total number of training steps inside the schedule and s the number of executed training steps of a specific layer. After scaling, the average learning rate of each layer is equal.
- **4421_rdo** (Random-Dropout): In this approach, Dropout Layers integrated within the backward path randomly prevent weight updates from being executed based on the corresponding BPLS-schedule. The layer-wise dropout rates corresponding to 4421 are 0%, 0%, 50%, 75%. The dropout rates align with the layers from output (left) to input (right), following the schedule notation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4421					
	Step 1	Step 2	Step 3	Step 4	mean
Classifier	n = 1	n = 1	n = 1	n = 1	n = 1
Conv5	n = 1	n = 1	n = 1	n = 1	n = 1
Conv4	n = 1	-	n = 1	-	n = 0.5
Conv3	n = 1	-	-	-	n = 0.25
Conv2	-	-	-	-	-

4444					
	Step 1	Step 2	Step 3	Step 4	mean
Classifier	n = 1	n = 1	n = 1	n = 1	n = 1
Conv5	n = 1	n = 1	n = 1	n = 1	n = 1
Conv4	n = 1	n = 1	n = 1	n = 1	n = 1
Conv3	n = 1	n = 1	n = 1	n = 1	n = 1
Conv2	-	-	-	-	-

4444_lr4421					
	Step 1	Step 2	Step 3	Step 4	mean
Classifier	n = 1	n = 1	n = 1	n = 1	n = 1
Conv5	n = 1	n = 1	n = 1	n = 1	n = 1
Conv4	n = 0.5	n = 0.5	n = 0.5	n = 0.5	n = 0.5
Conv3	n = 0.25	n = 0.25	n = 0.25	n = 0.25	n = 0.25
Conv2	-	-	-	-	-

4421_lr4444					
	Step 1	Step 2	Step 3	Step 4	mean
Classifier	n = 1	n = 1	n = 1	n = 1	n = 1
Conv5	n = 1	n = 1	n = 1	n = 1	n = 1
Conv4	n = 2	-	n = 2	-	n = 1
Conv3	n = 4	-	-	-	n = 1
Conv2	-	-	-	-	-

4421_rdo					
	Step 1	Step 2	Step 3	Step 4	mean
Classifier	d = 0%	d = 0%	d = 0%	d = 0%	n = 0%
Conv5	d = 0%	d = 0%	d = 0%	d = 0%	n = 0%
Conv4	d = 50%	d = 50%	d = 50%	d = 50%	n = 50%
Conv3	d = 75%	d = 75%	d = 75%	d = 75%	n = 75%
Conv2	-	-	-	-	-

Table 4.4: Example of BPLS-schedules

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Chapter 5

Methodology

The evaluation of BPLS addresses two main research questions:

1. How does BPLS behave regarding the Per-Layer Learning Rate approach?
2. Can BPLS improve the efficiency of fine-tuning without affecting training behavior negatively?

To answer these questions, the accuracy and execution time of BPLS were measured and compared with the following training approaches:

- Conventional Training (uniform learning rate applied to each layer)
- Per-Layer Learning Rate (each layer assigned a specific learning rate)
- Random-Dropout (randomly dropping individual gradients)

This chapter describes the evaluation methodology, detailing the methods used to ensure unbiased comparisons and explaining how accuracy and execution time were investigated. The actual evaluation results are presented in the Results chapter.

5.1 Training Conditions

BPLS was initially implemented in PyTorch [62]. The implementation primarily utilizes PyTorchs parameter freezing mechanism, which sets a *requires_grad* flag for layer parameters. These parameters can be weights or biases. When the flag is set, PyTorch computes gradients for these parameters and updates them during training. If the flag is not set, parameter updates are skipped. Modifying these flags after each training step enables the implementation of BPLS.

BPLS is designed to enhance the efficiency of fine-tuning neural networks on resource-constrained devices, especially MCUs. Therefore, all Hyper Parameters were selected as if the training were to occur on such a device. That rules out advanced, memory-intensive training techniques such as Momentum,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Adaptive Learning Rate, and particularly the ADAM optimizer. Instead, basic SGD (Stochastic Gradient Descent) with a simple plateau learning rate schedule was used. Additionally, the training batch size for each experiment was set to 1, which is often the only feasible batch size for training on MCUs.

To assess execution speed, the estimation of the number of training operations described in the Background chapter was employed. These estimates were validated through timing measurements conducted on various platforms. Details on this will be provided later.

5.2 Selecting BPLS-schedules of Interest

Before starting the actual evaluations, a baseline and specific BPLS-schedules of interest need to be defined. The baseline represents conventional fine-tuning applied to a partially frozen network. In this context, conventional means that every trained layer has the same learning rate assigned.

The number of frozen layers heavily depends on the applied pre-training, fine-tuning scenario, and network architecture. To determine the optimal number of frozen layers, various configurations were investigated. The number of frozen layers that resulted in the highest test accuracy was selected as the baseline. If multiple freezing configurations performed equally well, the configuration with the highest number of frozen layers was chosen.

The baseline was then used to derive several BPLS-schedules for further investigation. Most BPLS-schedules were defined by skipping certain operations of unfrozen layers furthest from the output layer. However, to better understand the behavior of BPLS, schedules that skipped training steps of layers closer to the output were also examined. The Per-Layer Learning Rate counterparts were derived by simply taking the average learning rate from each layer of the corresponding BPLS-schedule.

5.3 Performance Evaluation

One research question addressed in this thesis is whether BPLS can enhance fine-tuning efficiency without negatively affecting training behavior. To investigate this, we compared different training runs for the same network and dataset, employing various training techniques.

As previously mentioned, all Hyper Parameters were chosen as if the training were conducted on resource-constrained devices. Cyclical Learning Rate Schedules [6] are highly favored for training on such devices as they can achieve comparable training behavior to memory-intensive optimizers like ADAM with a minimum of resources. However, they introduce multiple Hyper Parameters that require tuning, such as the base learning rate, maximum learning rate, and step size. Instead, depending on the

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

scenario, we used either no scheduler or a simple Plato Learning Rate Scheduler with default settings for each run. The Plato Scheduler halves the learning rate if the loss does not decrease for specified number of epochs. The number of epochs until halving the learning rate depends on the fine-tuning scenario but remains consistent across all training runs in a specific scenario. More information about learning rate schedules can be found in the Background chapter.

The only Hyper Parameter that varies between runs is the base learning rate, which is the learning rate assigned to the classifier layer. In the Per-Layer Learning Rate approach, fractions of this base learning rate are assigned to different layers. To determine the best-performing base learning rate for a given run, we conducted a grid search [105]. This search technique involves training the network with different learning rates iteratively:

1. **Rough Search**: Provides a rough estimation of learning rates best suited for training a network with a specific approach. Several training runs are performed successively. The initial run uses the upper bound base learning rate lr_a. The base learning rate is then logarithmically reduced by dividing it by the rough step size $step_{\text{rough}}$ until it reaches the lower bound lr_b.
2. **Fine Search**: If the rough search was successful, the range around the optimal base learning rates can be estimated. The search is then repeated, focusing on the area around the best-performing learning rate from the previous measurements. If multiple learning rates achieved equivalent test accuracy, the highest of them is chosen as the center of the new search. Like before, the learning rate is logarithmically decreased starting from an upper bound. The step size ($step_{\text{fine}}$) in this phase is smaller than the step size used in the rough search, further refining the range around the optimal base learning rates.
3. **Very Fine Search**: At this stage, depending on the specific search configurations, the outcome is likely close to an optimal base learning rate. The search can be repeated with progressively reduced range and step size to search for an even better-performing learning rate.

In our experiments, we set the step sizes to $step_{\text{rough}} = 1.4$, $step_{\text{fine}} = 1.2$, and $step_{\text{very_fine}} = 1.1$. Figure 1.1 displays the achieved test accuracy for different base learning rates during conventional fine-tuning. Each training run performed 30 epochs. The area with a white background represents the Rough Search, the yellow background represents the Fine Search, and the orange background represents the Very Fine Search. The red circle indicates the learning rate selected for further evaluation.

Once the grid search process is completed, the estimated optimal base learning rate is used to perform additional training epochs. Finally, these training runs are compared in terms of test accuracy and training time.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

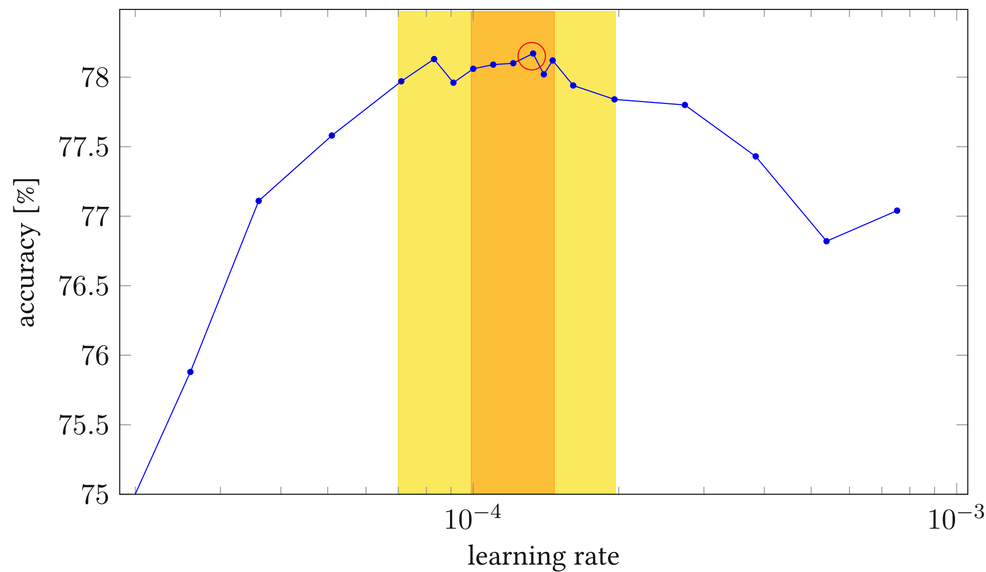


Figure 5.1: Learning Rate Grid Search

5.4 Time and Memory Estimation

One goal of BPLS is to reduce power consumption and training time by minimizing the number of executed operations. As mentioned earlier, there is a direct relationship between the number of operations, execution time, and power consumption. Instead of measuring execution time for a specific implementation and platform, we estimated the number of executed MAC-operations using the formulas introduced in the Background chapter. This estimation only includes computations involved in forward and backward propagation and excludes the evaluation phase and dataset preparation.

To validate the mathematical estimations, we conducted measurements on three different platforms: PyTorch code executed on a server CPU, and CPP code executed on a desktop CPU and MCU. Although the same CPP code was used for the desktop CPU and MCU, we employed different compilers and optimizations. Due to the limited resources of the MCU, we defined a specific fine-tuning scenario involving a relatively small network and dataset. This same scenario was used for the measurements on the desktop platform. The results on these platforms were validated against the Pytorch implementation to ensure correct training behavior. Details of all fine-tuning scenarios will be provided later. The platform specifications are as follows:

1. **MCU**: Nucleo-f303k8 development board, featuring an ARM Cortex-M4 based STM32f303k8t6 MCU. It is a 32-bit MCU clocked at 72MHz, equipped with 64KB flash, 16KB SRAM, and a floating-point unit (FPU).
2. **Desktop**: Intel Core i7 12700. The network is small enough to fit entirely into the cache.
3. **Server**: Intel Xeon 5118 Gold @ 2.3GHz

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

On all platforms, training was executed on a single CPU core using a single thread. Network parameters, inputs, and outputs were represented using full-precision floating-point values, not only on the server and desktop platforms but also on the MCU, utilizing the integrated FPU. To reduce variance, we performed 100 warm-up training steps before the actual measurements began.

While training on the server and desktop platforms is straightforward, there are some important details about the MCU configuration to note. The training code executed on the MCU was compiled twice using different optimization settings. The Omin setting prioritizes minimal flash size, while Ofast prioritizes fast execution. It can be assumed that the Ofast compiler utilizes optimizations like loop unrolling and multiload/store, although it is difficult to predict the exact optimizations applied. The Omin setting also applies optimizations, but the primary optimization goal is to minimize code size, making it unlikely that optimizations like loop unrolling are used.

Even when unusual, the CPP code is implemented to store all data produced during training on the stack. Compiling the CPP code provides information about the static memory stack. This information can be used to validate memory estimations, which in turn can be used to estimate the peak memory requirements of BPLS-schedules.

5.5 Similarity Evaluation

The second research question this thesis aims to answer is how similar BPLS behaves compared to other training approaches, especially Per-Layer Learning Rate. To address this question, training runs were performed where every run of a scenario uses the same Hyper Parameters.

To delve deeper, we not only compare BPLS with its corresponding Per-Layer Learning Rate counterpart but also with training that utilizes Dropout Layers in the backward path. These Dropout Layers randomly discard gradients, preventing certain weights from being updated. Additionally, we performed training in the most conventional way, where the same learning rate is applied to each layer. For comparison, we also investigated BPLS with up-scaled learning rates. A detailed explanation of the different training approaches can be found in the BPLS chapter.

For each layer separately, the weights of the network state leading to the highest test accuracy, denoted as $\vec{w}_{best_a}$ and $\vec{w}_{best_b}$, with a and b representing different training runs, are directly compared. It should be noted that in this case, the comparisons are not commutative. The comparison between training run a and b compares the weights of the training epoch where a performed best. However, the comparison between b and a compares the weights of the training epoch where b performed best. The metrics used for evaluating the similarity are mainly Euclidean Distance and Cosine Similarity. More

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

details about these metrics, including a mathematical description, can be found in the Background chapter.

Additionally, the UMAP tool (Uniform Manifold Approximation and Projection for Dimension Reduction) [88] was used to investigate the similarity of training runs over several epochs. UMAP generates a 2D representation for each combination of layer, training run, and epoch and clusters these 2D mappings based on their similarity. As a baseline, the Conventional and Per-Layer Learning Rate training runs are defined. UMAP automatically maps the BPLS, Random-Dropout, and Upscaled training runs in relation to these baselines.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Chapter 6

Network Architectures and Scenarios

To evaluate the behavior of BPLS in terms of performance and similarity to other training approaches, three experiments were conducted. These experiments involved taking pre-trained neural networks and fine-tuning them to classify new or modified data. The investigated networks are simple VGG-style [78] CNNs composed of convolutional, linear, max-pooling, dropout, batch normalization layers, and ReLU activation functions.

The experiments utilized well-known datasets commonly used for evaluating CNNs, such as CIFAR10 and CIFAR100 [28]. Additionally, a custom-generated keyword dataset was used to fine-tune a network pre-trained on a subset of Googles Speech Commands dataset [30]. To evaluate the number of operations and peak memory usage, a separate, relatively small network and dataset were defined. This dataset is part of the UCR Time Series Classification Archive [10].

6.1 CIFAR10 - Scenario

While MNIST [25] is perhaps the most well-known dataset for image recognition, CIFAR10 is commonly used for professional evaluation of CNNs. It consists of 60,000 color images sized 32x32, equally distributed across 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset is divided into 50,000 training images and 10,000 test images. More information can be found on the official website [28].

In this experiment, the network was pre-trained on CIFAR10 with several augmentations, including random horizontal flip, rotation, perspective transformation, and color jitter. To generate the data used for fine-tuning, the original CIFAR10 images were rotated by 180° and converted to grayscale by averaging the color channels. When defining these modifications, it was important to induce changes

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

to several layers of the network during fine-tuning, not just the output layer, while still benefiting from the pre-training. This allows to effectively apply and evaluate BPLS. Example images of the datasets used for pre-training and fine-tuning are shown in Figure 6.1.

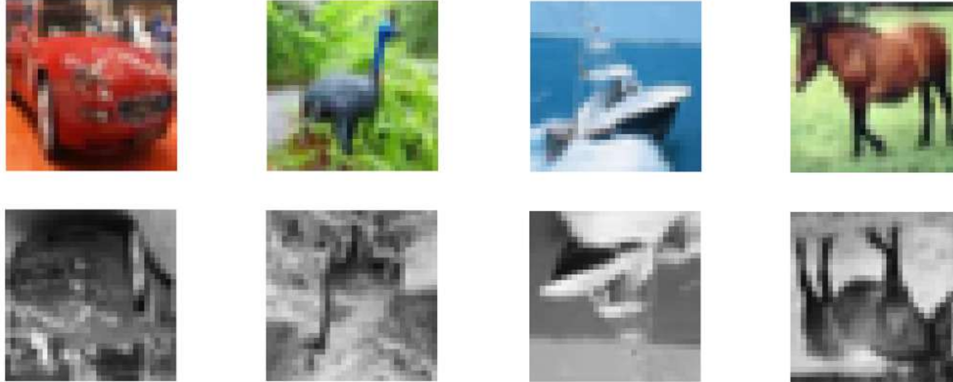


Figure 6.1: CIFAR10 - original for pre-training (top) and modified for fine-tuning (bottom)

A corresponding real-life scenario could involve a toy factory with three production lines. Each line produces various toy categories and is equipped with four smart cameras to count the number of toys produced per category and ensure the correct toy is produced according to a production schedule. The toy categories correspond to the CIFAR10 classes (e.g., cars, airplanes, horses). The smart cameras capture grayscale images. To conserve bandwidth and energy, the cameras classify the images locally and only transmit the results. However, due to fixed camera orientations, adjustments to the neural network are necessary to maintain performance.

After installing the cameras, a test run of the production line is conducted under worker supervision to verify that the captured images match the expected labels. This results in a labeled dataset which is then used to fine-tune the neural network. Modifications to products or production lines prompt a new test run, during which the smart cameras automatically fine-tune their neural network to accommodate the new conditions.

This scenario, tailored to the CIFAR10 dataset, is unlikely to occur in real life. In practice, neural networks are more commonly used for detecting production issues, such as damaged or defective products. Additionally, this experiment does not involve capturing images on a production line but instead uses modified CIFAR10 images. CIFAR10 images typically depict outdoor scenes, like boats on water or frogs in natural habitats, while images captured at a production line usually feature more uniform structures and backgrounds. Ideally, a custom dataset would be created for this specific task. However, the primary focus of this thesis remains the introduction and evaluation of BPLS, an efficient fine-tuning approach.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

6.2 CIFAR100 - Scenario

CIFAR100 is very similar to CIFAR10 but consists of 100 classes instead of 10. Each class comprises 500 training images and 100 test images. These 100 classes are referred to as fine classes and are organized into 20 superclasses. Examples of classes include Fish (Aquarium Fish, Flatfish, Shark), Flowers (Orchids, Poppies, Roses), and Insects (Bees, Beetles, Butterflies).

Due to the higher complexity of CIFAR100 compared to CIFAR10, we applied similar but less intensive modifications to generate the fine-tuning dataset. Instead of converting images to grayscale as we did for CIFAR10, we simply inverted the colors. This allows us to reconstruct the original images by color-inverting the modified dataset again, ensuring no information is lost due to the modification. Additionally, similar to CIFAR10, we rotated the dataset by 180°. Given the similarities to CIFAR10, a comparable real-life scenario can be envisioned. Example images are shown in Figure 6.2.

Figure 6.2: CIFAR100 - original for pre-training (top) and modified for fine-tuning (bottom)

6.3 Key-Word - Scenario

While CIFAR10 and CIFAR100 are popular for evaluating image classification neural networks, the scenarios introduced earlier are somewhat removed from reality. The third and final scenario used for performance evaluation aims to be more realistic, focusing on audio keyword classification. In this scenario, a simple CNN is pre-trained to accurately classify 8 English keywords, including Go, Stop, Left, and Right.

The dataset used for pre-training contains 6400 training samples and 1600 test samples stored as 16 kHz uncompressed mono-channel audio files. Pre-processing involves converting these audio files from WAV format to spectrograms. Spectrograms visually represent frequencies over time as images with a single color channel.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

English	Deutsch (German)	日本語 (Japanese)
go	los	行け (ike)
left	links	左 (hidari)
right	rechts	右 (migi)
up	rauf	上 (ue)
down	runter	下 (shita)
stop	halt	止まれ (tomare)
yes	ja	はい (hai)
no	nein	いいえ (iie)

Table 6.1: Key-words used in the experiment

This dataset is a subset of a larger dataset developed by Google [30], which originally contains 32 keywords. In addition to the mentioned keywords, the original dataset includes others such as Cat, Five, House, or Learn. The pre-training process for this scenario is featured as a TensorFlow tutorial using the exact same dataset and network [106].

When discussing the fine-tuning of this network, a real-life use case immediately comes to mind. The neural network could be personalized to increase classification accuracy for a specific individual. For this purpose, the individual would record the same 8 keywords multiple times and use this data for fine-tuning. To emphasize differences between various training approaches more clearly, we decided to go a bit further. Instead of simply recording the 8 English keywords, we recorded counterparts in different languages, specifically German (the authors mother tongue) and Japanese (due to parts of this thesis being executed at the University of Tokyo, Japan).

To induce adaptation of not only the linear layers close to the output but also the convolutional layers, we made a slight modification to the data format. We downsampled the newly recorded audio files to 8 kHz, half the sampling rate used during pre-training. The fine-tuning dataset consists of 480 training samples and 160 test samples, equally distributed across the 8 classes.

Table 6.1 lists the keywords in all three languages, followed by two waveforms with corresponding spectrograms (Figure 6.3). The left waveform represents a 16 kHz pre-training data sample for the command ”Go!” The right waveform represents an 8 kHz data sample for the Japanese counterpart ” 行け” (ike!), used during fine-tuning. When comparing the spectrograms, it can be seen that the rectangles in the right spectrogram are taller and wider, due to the lower sampling rate.

A real-life example could be as follows: Imagine a small robot dog initially trained to understand the 8 commands in English. However, not everyone is able or willing to speak English, particularly older individuals or young children who are just learning their native language. Therefore, the robot dog could have a configuration mode. This mode could be used to record the pre-trained English words

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

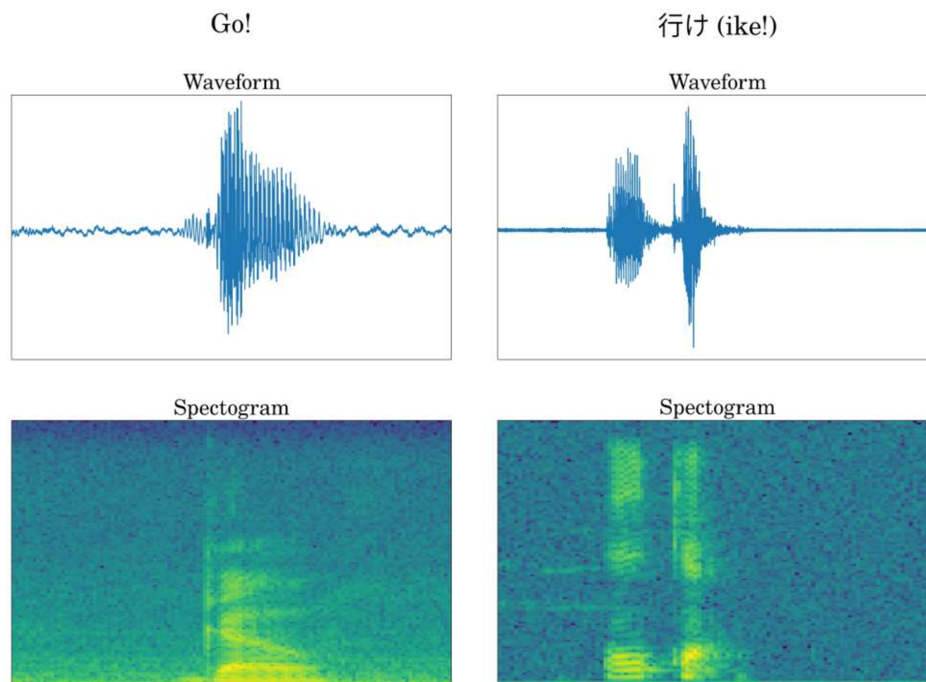


Figure 6.3: Key-words - Go! for pre-training (left), 行け (ike!) for fine-tuning (right)

to enhance the dogs comprehension. But, it could be used to teach the dog completely new words, such as those listed in Table 6.1. When the dog hears these commands, it could attempt various actions randomly. If the dog performs the correct action, the user could reward it with a treat, which allows the dog to label the command.

This robot dog could, for example, be utilized in retirement homes or kindergartens. In both settings, supervisors could assist children or older individuals in training the robot dog in an enjoyable manner. Studies have shown that these types of robot pets can have the same entertaining and therapeutic effects on people as real animals. [107]

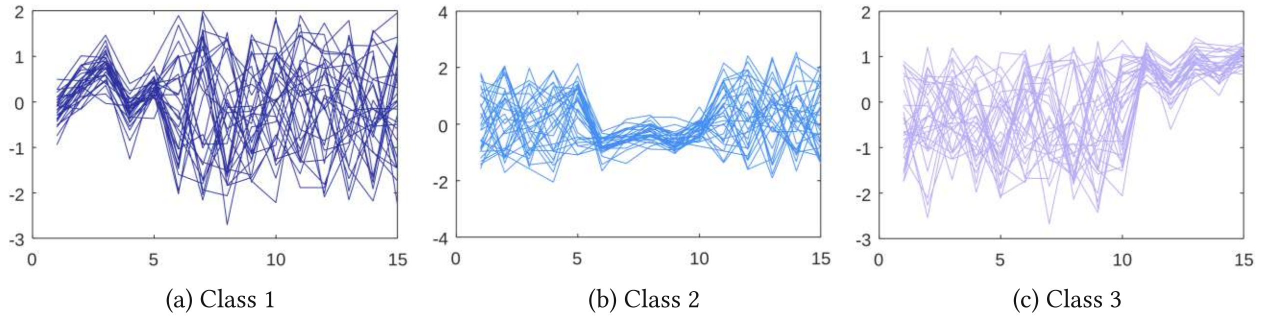
6.4 MCU - Scenario

Differing from the previously introduced scenarios, this one is specifically designed to validate the number of operations and memory estimations, rather than for performance evaluation. For this purpose, we utilized the smallest dataset from the UCR Time Series Classification Archive [10] to train a relatively small neural network from scratch.

The specific dataset used is called SmoothSubspace, which consists of 150 training and 150 evaluation samples. These samples are time series of length 15 representing 3 different classes. Examples of

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the dataset are shown in Figure 6.4.



(a) Class 1 (b) Class 2 (c) Class 3

Figure 6.4: SmoothSubspace - UCR Time Series Classification Archive [10]

6.5 Network Architectures

Following, for each scenario the originally pre-trained network and their corresponding fine-tuning version are illustrated and briefly explained. The fine-tuning networks are derived from the pre-trained networks but exclude certain layers in most cases.

This thesis focuses on fine-tuning on resource-constrained devices. To reduce requirements, the Batch-Norm parameters defined during pre-training were merged with the corresponding Convolutional or Linear Layer located just before the Batch-Norm Layer. Additionally, in the networks of the CIFAR10 and CIFAR100 scenarios, Dropout Layers were removed. However, in the Key-Word Scenario, keeping the Dropout Layers led to significantly better performance, so they were retained. Furthermore, all 2D Convolutions use 3x3 filter kernels with a stride of 1. Padding was applied to maintain constant data size during convolution in all networks except for the one related to the Key-Word Scenario.

The network related to CIFAR10 is based on a reduced version of VGG8 known as VGG-Small [108]. To optimize the network further for resource-constrained devices, the feature depth (number of feature maps) was reduced. While VGG-Smalls Classifier Layer has 8192 input values, the version used in this thesis has only 1024. The network for CIFAR100 is a slightly reduced variant of VGG11 [109]. In [110] it can be found under the name *baseline network Base B*. However, this network is quite large for execution and training on resource-constrained devices, even after applying pruning and quantization techniques discussed in the Background section. The network related to the Key-Word Scenario was sourced from the TensorFlow tutorial [106] and retained in its original form.

The network used to evaluate the number of operations and peak memory requirements is a reduced version of the Fully-Convolutional network described in [111]. The goal of [111] was to achieve high accuracy on the UCR Time Series Classification Archive [10]. Because the original network exceeds the capacities of the MCU device used for the evaluation, we reduced the number of feature maps.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

	Cifar10-Net	Cifar100-Net	Key-Word-Net	MCU-Net
Total params	82,330	7,935,652	1,625,608	1,155
Trainable params	82,330	7,935,652	1,625,608	1,155
Total mult-adds	9.95 MOps	212.88 MOps	16,40 MOps	0.02 MOps
Input size	0.01 MB	0.01 MB	0.00 MB	0.00 MB
For-/backward pass size	0.46 MB	1.98 MB	0.63 MB	0.00 MB
Params size	0.33 MB	31.74 MB	6.50 MB	0.00 MB
Estimated Total Size	0.80 MB	33.74 MB	7.14 MB	0.01 MB

Table 6.2: Statistics of utilized networks

Table 6.2 compares the number of parameters for each network. It is important to note that the training was performed in full precision. Quantizing the parameters would reduce the memory footprint. These values were provided by Torchsummary which does not allow for easy format changes.

Figures 6.5 to 6.9 display the applied networks. Layers filled with blue color are frozen, meaning they are not updated during fine-tuning. Each layer is annotated with the number of parameters on the right side. For Conv2D layers, details such as the number of output channels, image width, and image height are provided below the layer type. The kernel size for each Conv2D layer is fixed at 3x3. For Conv1D layers, the number of output channels and signal width are listed below the layer type. Unlike Conv2D layers, each Conv1D layer has a specific kernel size, listed on the right side of the layer type.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

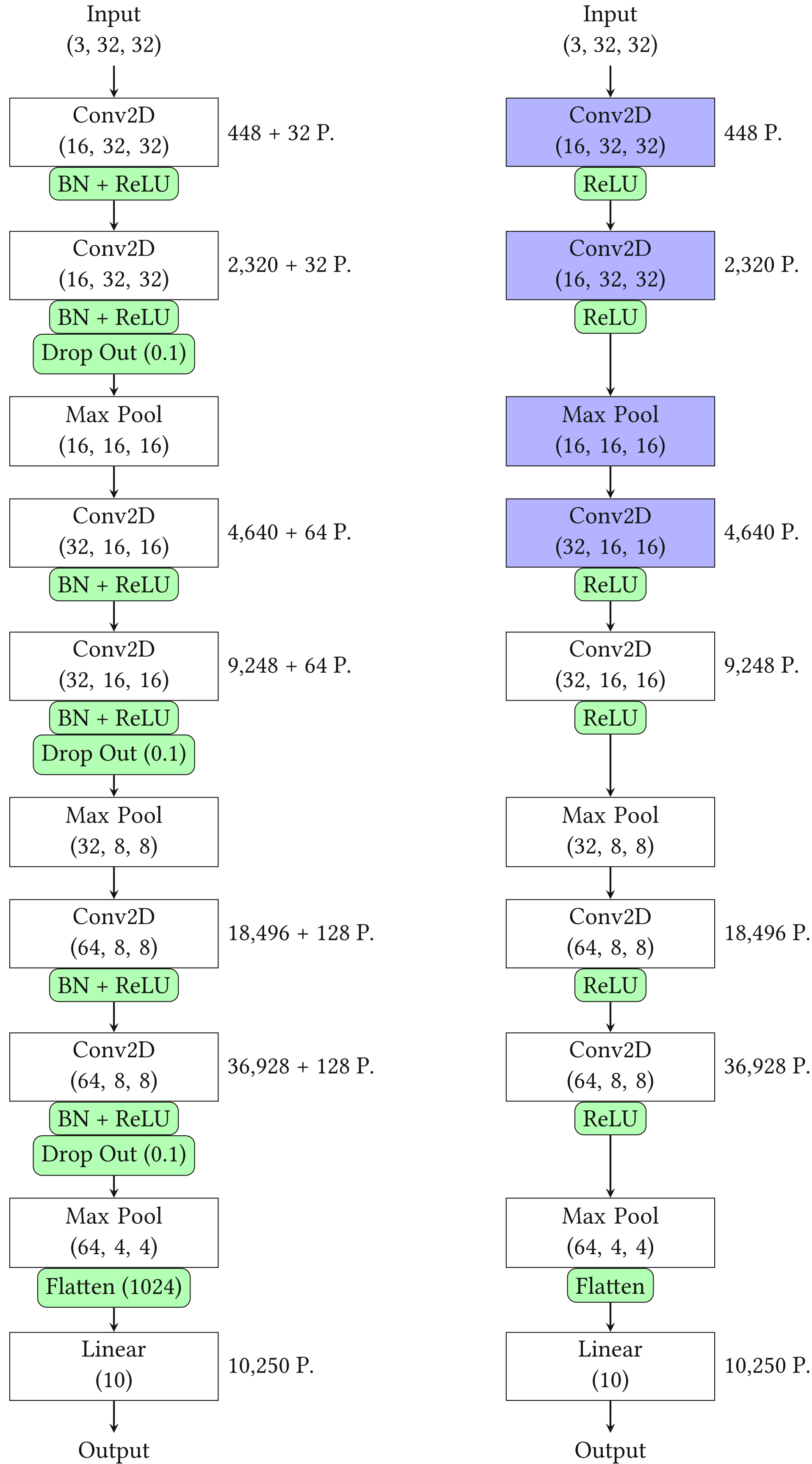


Figure 6.5: Network of the Cifar10 - Scenario (Pre-Training: left, Fine-Tuning: right)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

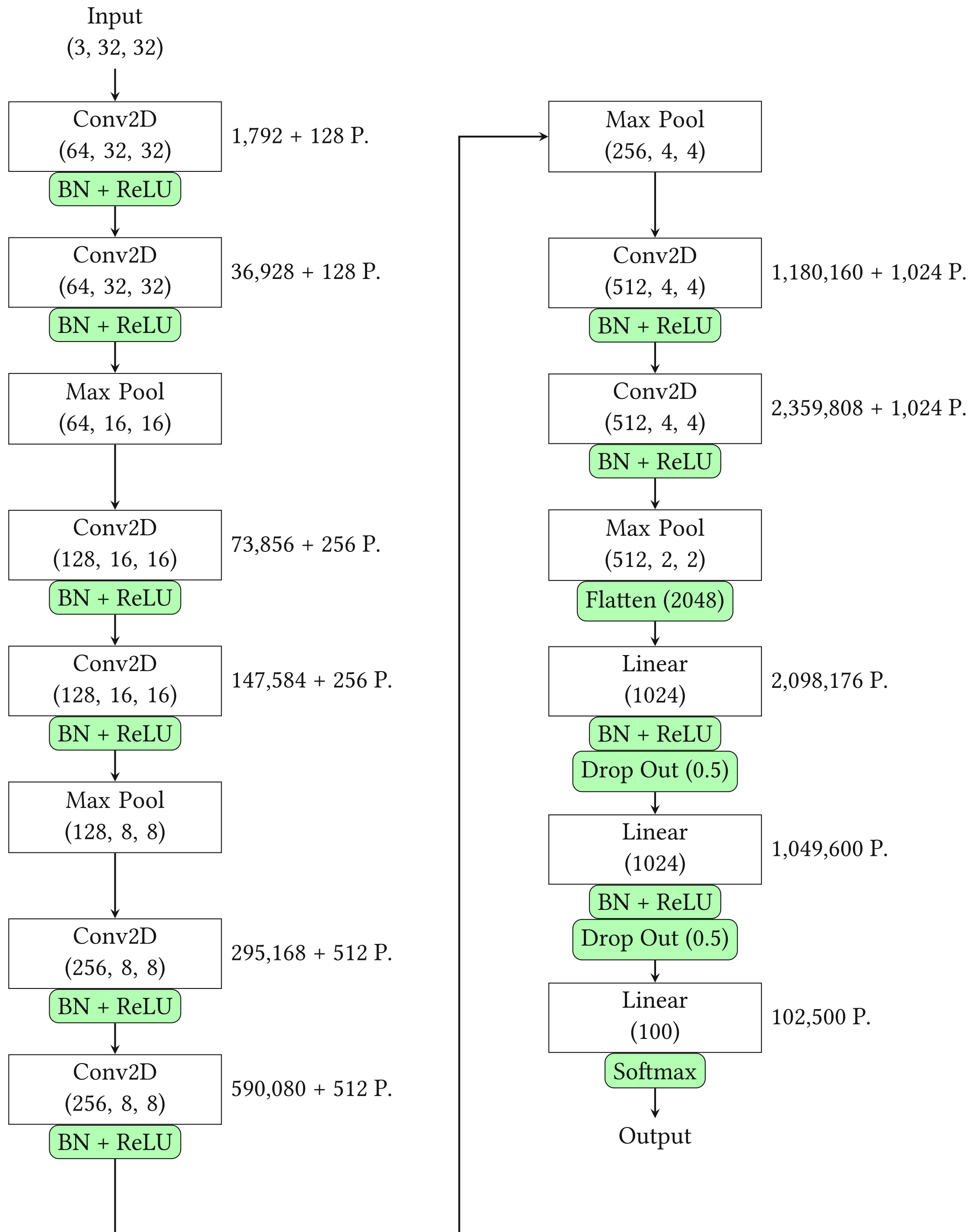


Figure 6.6: Network of the Cifar100 - Scenario (Pre-Training)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

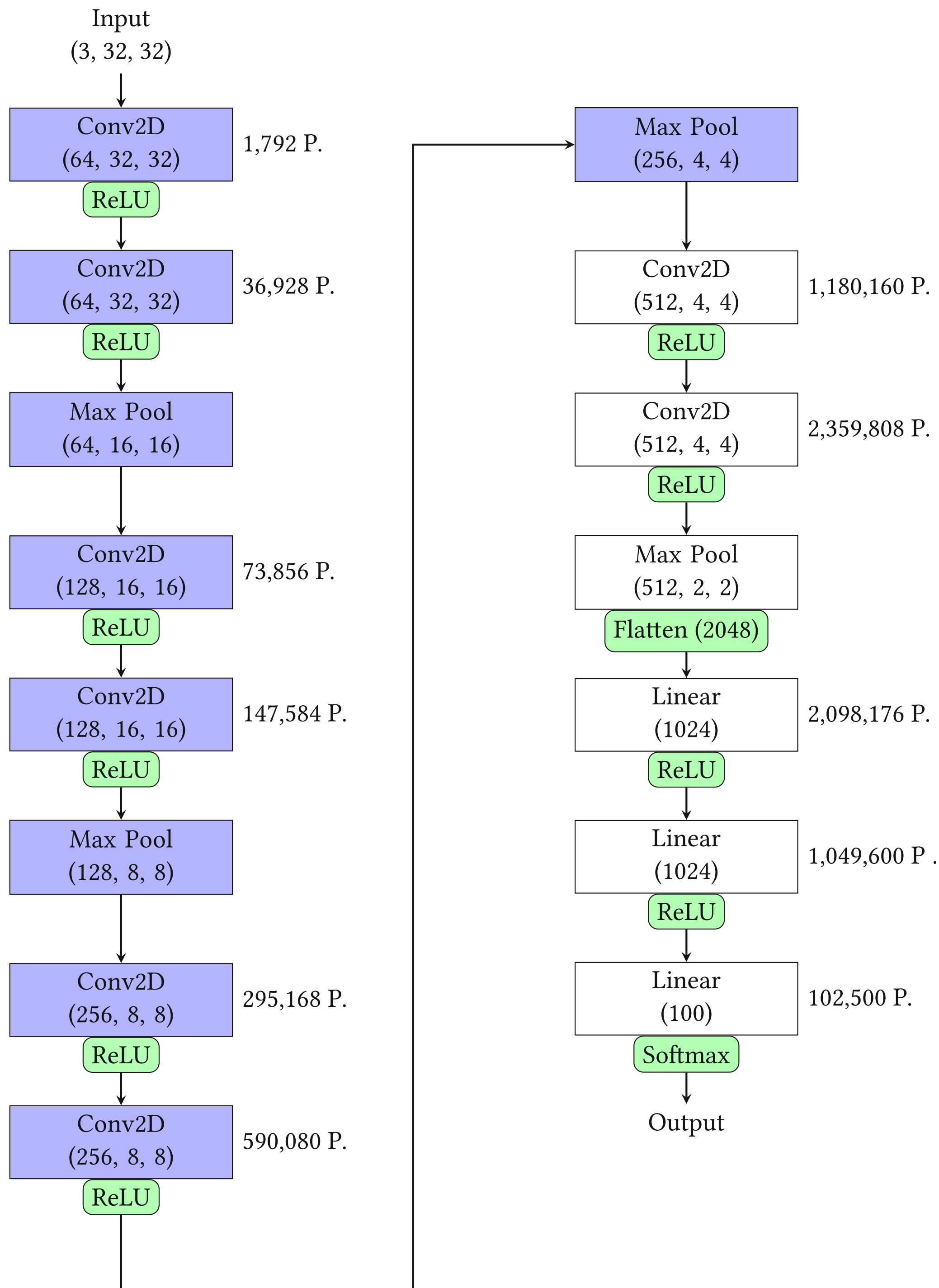


Figure 6.7: Network of the Cifar100 - Scenario (Fine-Tuning)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

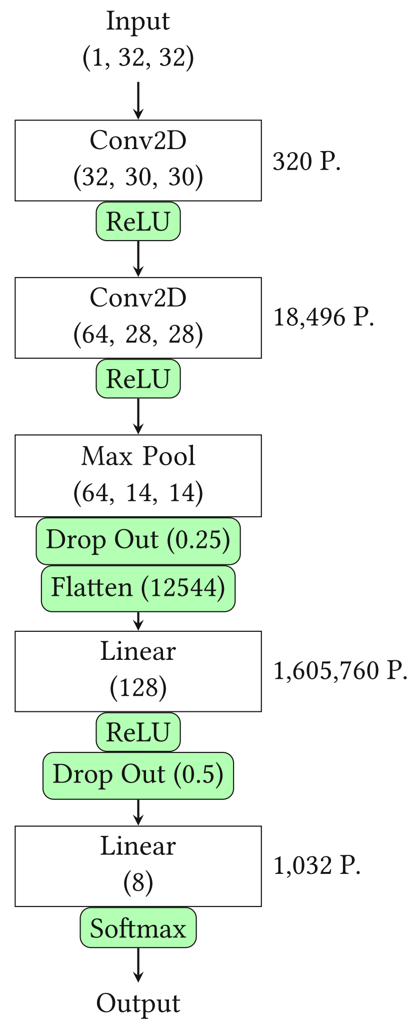


Figure 6.8: Network of the Key Word - Scenario (Pre-Training and Fine-Tuning)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

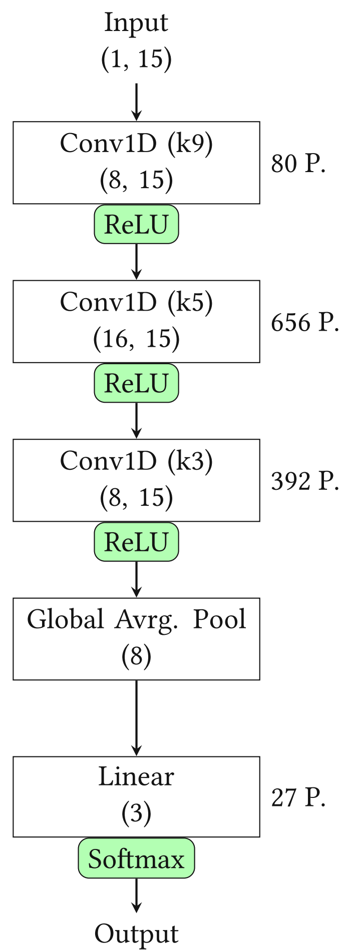


Figure 6.9: Network used for time and memory measurements on MCU and Desktop

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Chapter 7

Results

This chapter addresses the research questions of this thesis one by one. Firstly, the results concerning the similarity of different training approaches are presented. This is followed by performance tests under real-life conditions. Instead of measuring specific training runs executed on particular hardware, the comparisons are based on estimations of number of operations and peak-memory. The final part of this chapter involves the evaluation of these estimations.

To fully understand this chapter, it is essential to know the naming conventions introduced earlier, which can be found in the BPLS chapter under Notation. Several BPLS schedules were investigated. Their concrete configuration may not always be clearly deducible from their notation. Accordingly, all applied schedules can be found in the Appendix section.

7.1 Terms

In this chapter, several terms frequently appear. They are briefly explained here to prevent confusion:

- **Training category**: Refers to the technique applied for training, such as BPLS, Per-Layer Learning Rate, Random-Dropout, Up-Scaled, or Conventional Training.
- **Training approach**: Refers to a specific configuration of BPLS-schedule, Per-Layer Learning Rate, Random-Dropout, Up-Scaled, or Conventional Training.
- **Training run**: The execution of a training approach with specific hyperparameters.
- **Accuracy**: Refers to the test accuracy. After each training epoch, the network is tested using the test dataset. The proportion of correctly classified samples to the total number of test samples defines the test accuracy.
- **Best accuracy**: The highest test accuracy achieved within a training run.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- **Learning rate**: Refers to the base learning rate assigned to the classifier layer.
- **Operations**: Refers to Multiplication and Accumulation (MAC) operations.
- **Training time**: Number of epochs or MAC operations until sufficient test accuracy is achieved. ”Sufficient test accuracy” refers to accuracy close to the best accuracy of the specific training run.
- **Conv(n)**: Refers to the nth Convolutional Layer of a network.
- **Lin(n)**: Refers to the nth Linear Layer of a network.
- **Peak memory**: Maximum amount of memory required for training, defined by the training step requiring the most memory.
- **Peak operations**: Highest number of executed operations, defined by the training step executing the most operations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

7.2 Similarity of different Training Approaches

One research question addressed in this thesis is *how BPLS behaves in regard of the Per-Layer Learning Rate approach*. Therefore, this chapter analyzes the similarity between different training approaches for the CIFAR10 and CIFAR100 scenarios in detail. The Key-Word scenario does not differ significantly and is therefore left out.

The similarity metrics used are Euclidean Distance and Cosine-Similarity. To enhance readability, the Cosine-Similarity values were converted to angles. As explained in the Methodology chapter, the similarity between training approaches within one fine-tuning scenario was measured by applying the same hyperparameters to each training run.

Both metrics, Distance and Cosine-Similarity, are visualized using Polar Plots and Color Maps. Furthermore, UMAPs (Uniform Manifold Approximation and Projection) are used to provide a different perspective on the similarity of training approaches. More information on these metrics and tools can be found in the Background chapter. All measured data are listed in the appendix section.

7.2.1 Interpretation

To help understanding the subsequent plots, a brief explanation is provided here.

Polar Plots

The Polar Plots are structured as follows: along the borders, the base training approaches (baselines) are arranged. Next to each baseline, the corresponding top 3 most similar training approaches are listed. Training approaches closer to the center of the plot are less similar to the baseline, while those closer to the borders show higher similarity. It should be noted that training approaches that are part of the baselines are excluded from the comparisons. For each plot, similarity is defined by the corresponding metric, either Distance or Cosine-Similarity. Accordingly, the center of the plot represents high distances and wide angles, while the borders represent low distances and narrow angles.

Color Maps

The Polar Plots provide a good overview. However, some information is difficult to extract. To deepen understanding, Color Maps are drawn. These Color Maps consist of training approaches sorted from most similar (left) to least similar (right), while ignoring their relative values. Similar to the Polar Plots, the similarity was measured using Distance and Cosine-Similarity. The entries of the Color Maps are encoded by a combination of colors and symbols, interpreted as follows:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- **Background color**: Represents the target learning rate (e.g., BPLS-schedule 4421 has the same background color as its Per-Layer Learning Rate counterpart 4444_lr4421).
- **Symbol**: Indicates the technique applied to achieve the target learning rate (BPLS, Random Dropout, Per-Layer Learning Rate).
- **Fill color**: Denotes the start learning rate (e.g., the background color of BPLS-schedule 4421 corresponds to the fill color of its Up-Scaled counterpart 4421_lr4444. Conversely, the background color of 4421_lr4444 corresponds to the background color of 4444).

UMAP

To provide an alternative view of the similarity of different training approaches, the Dimensionality Reduction tool UMAP is used. UMAP maps all weights of one layer to a single point. Plotting different epochs in a row allows us to see how specific training approaches modify the network during training. Unlike the previously introduced Polar Plots and Color Maps, this allows for similarity analysis not only at a specific epoch but over the entire training process.

The absolute position of the points in the plot generated by UMAP does not carry any information. It is the relative position that matters. Points close to each other indicate high similarity, while those far apart indicate low similarity. More information about Dimensionality Reduction tools in general and UMAP in particular can be found in the Background chapter.

Instead of allowing UMAP to map the points completely freely, baselines are defined, similar to the Polar Plots. For each mapping, UMAP chooses the location of the baseline point that is most similar. Accordingly, it can happen that the mapping for a training approach starts at baseline A, jumps to baseline B after a few epochs, and ends at baseline C. To improve clarity, the point corresponding to the epoch where a training approach first reaches its peak test accuracy is displayed as a large dot.

7.2.2 Cifar10 - Scenario

For the Cifar10 scenario the learning rate was set to 140E-6. Following, Polar Plots, Color Maps and UMAPs describe the similarity behavior observed at the Cifar10 scenario.

Polar Plots - Distance

Figure 7.1 displays the Euclidean Distance between training approaches as Polar Plots. The following observations can be made:

- **Range**: Each Polar Plot shows a different distance range, where a wide distance range indicates high variation. At the first two layers, Conv4 and Conv5, the training approaches are differently

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

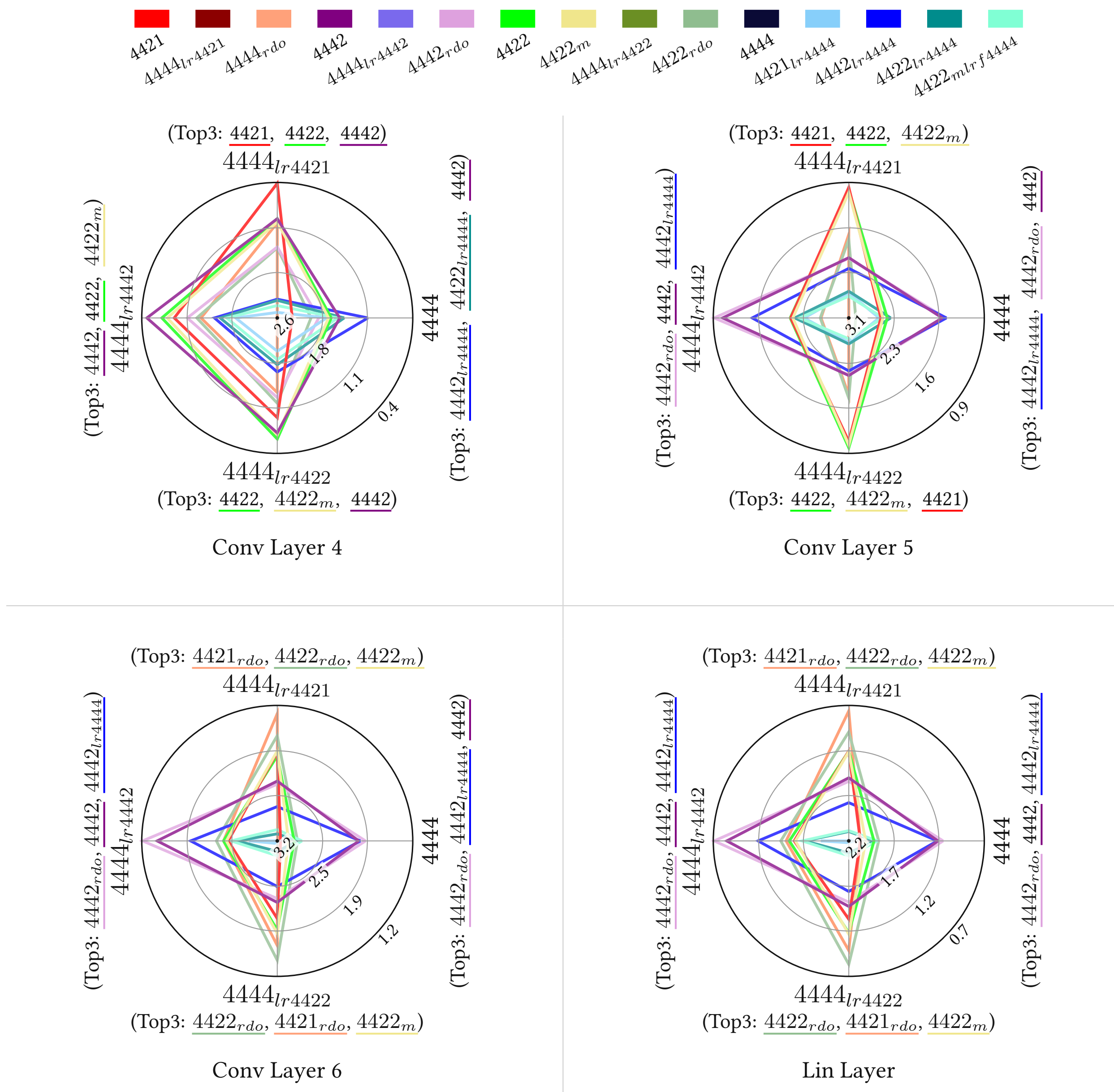


Figure 7.1: Cifar10 Network - Distance Polar Plots

configured, while the subsequent ones, Conv6 and Lin1 are treated equally by each training approach. Therefore, one would expect a wide distance range at the first two layers and a narrow range at the last two layers. Upon closer examination of the actual values, the range at both, Conv4 and Conv5 is 2.2. In contrast, the range at Conv6 is 2.0, and at Lin1 1.5.

- **Conv4**: Especially for the first unfrozen layer (Conv4), the BPLS-schedules show very high similarity to their corresponding baselines. More specifically, the BPLS-schedules 4442, 4422, and 4422_m exhibit relatively high similarity to the 4442 and 4422 baselines, but not to the 4421 baseline. On the other side, the BPLS-schedule 4421 shows relatively low similarity to the 4442 and 4422 baselines. That could be related to the underlying schedule configurations. While 4421

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

trains layer Conv4 each 4th step, at 4442 and 4422 Conv4 is trained every 2nd step. Additionally, the plots reveal a relatively low similarity between Random Dropout and the Per-Layer Learning Rate baselines.

- **Conv5**: When examining Conv5, it can be observed that the BPLS-schedules 4421, 4422, and 4422_m are all very similar to the baselines of 4421 and 4422. For all these BPLS-schedules, Conv5 is trained every 2nd step. In contrast, 4442, where Conv5 is trained every single step, is not similar to either 4422, 4422_m, or 4421. But, 4442 shows very high similarity to Conventional Training (4444), where Conv5 is trained every single step as well. This indicates a correlation between the training frequency of certain layers and their learning rate.
- **Conv6 and Lin1**: Conv6 and Lin1 are trained by each BPLS-schedule at every step. Based on the previously discussed observations, it might be expected that at Conv6 and Lin1, each BPLS-schedule shows approximately the same similarity to each baseline. However, as can be seen in the corresponding plots, this is not the case. Conv6 and Lin1 reveal a similar structure as Conv5.
- **Random Dropout**: Across all layers, a consistent pattern can be observed. At layers where baselines apply a fraction of their base learning rate, the similarity to BPLS is significantly higher than to Random Dropout. In these layers, BPLS skips training steps while Random Dropout drops gradients. Conversely, at layers where baselines apply their base learning rate, the similarity to BPLS is lower than to Random Dropout. In these layers, BPLS does not skip any training step and Random Dropout does not drop any gradients. Consequently, the configuration of all training approaches at these layers is essentially equivalent. This indicates that they are influenced by their prior layers to some extent.
- **Up-Scaled**: In addition to BPLS and Random Dropout, the Polar Plots also display up-scaled variants of the BPLS-schedules. These are expected to show high similarity to Conventional Training. However, this appears to be true only for the up-scaled variant of 4442, especially evident at Conv4. The other up-scaled BPLS-schedules do not show increased similarity to Conventional Training. In fact, besides Conv4, it seems that the up-scaled variants of 4422, 4422_m, and 4421 are more similar to the 4442 baseline than to Conventional Training.

Polar Plots - Cosine-Similarity

Figure 7.2 displays the Cosine-Similarity values corresponding to the previously shown Distance Polar Plots. While the overall structure remains similar, these plots offer the following insights:

- **Range**: The range of Cosine-Similarity consistently decreases from Conv4 to Lin1. Conv4 has the widest range at 33.87°, followed by Conv5 with 30.42°. Conv6 exhibits a Cosine Similarity range of 22.8°, and Lin1 of 20.27°. As mentioned earlier, wide ranges were expected at Conv4 and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

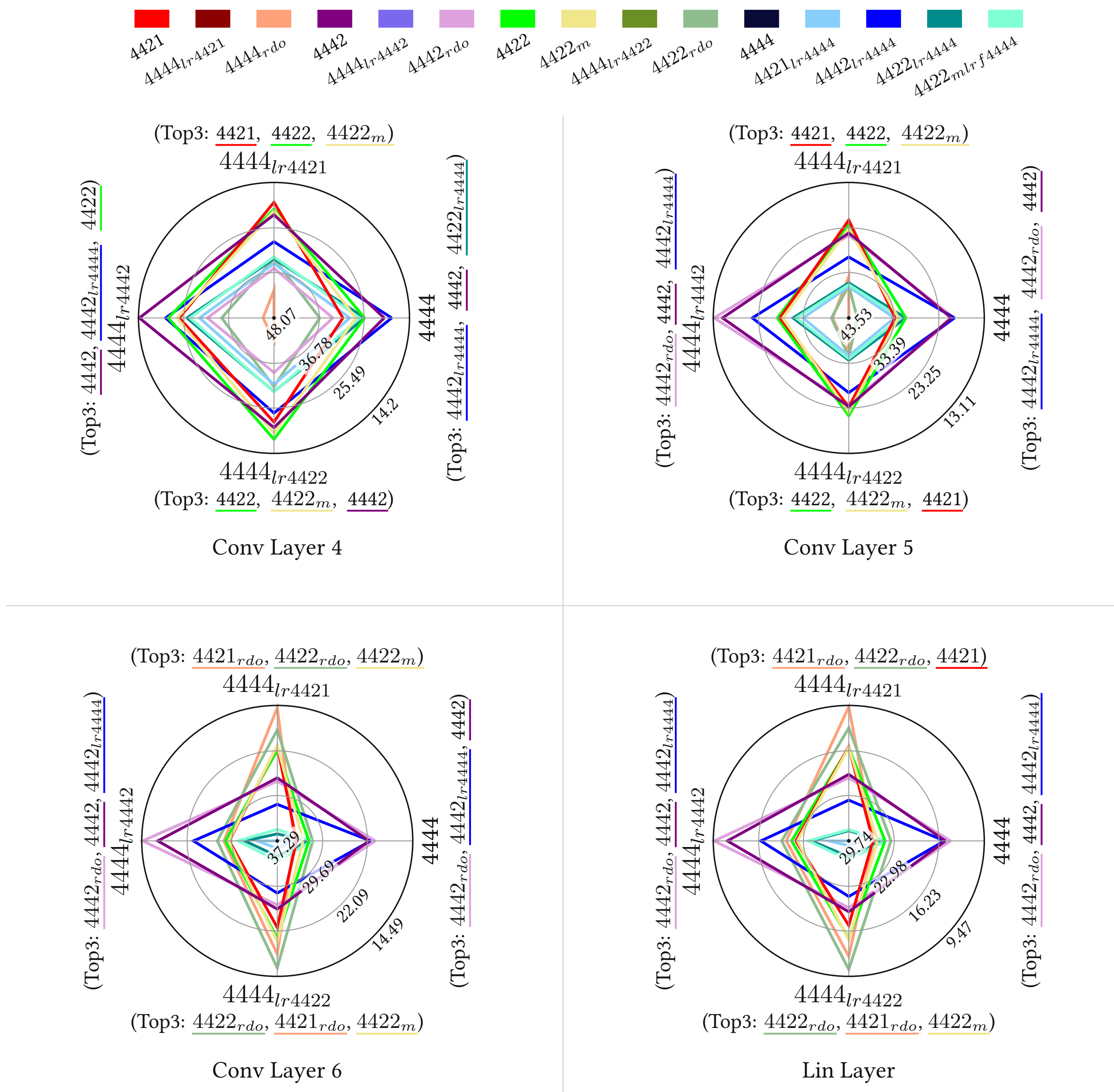


Figure 7.2: Cifar10 Network - Angle Polar Plots

Conv5, while narrower ranges were expected at Conv6 and Lin1. The Cosine-Similarity plots align with these expectations even more clearly than the Distance Polar Plots.

- **Conv4**: The structure visible at Conv4 also differs from its Distance counterpart. The corresponding Distance Polar Plot showed a relatively high similarity between BPLS-schedules 4442, 4422 and 4422_m and their baselines, but a low similarity to the 4421 baseline. Here, the similarity between BPLS-schedules 4442, 4422 and 4422_m to the 4421 baseline, but also to Conventional Training is significantly higher. Additionally, 4421 shows higher similarity to the baseline of 4422 in this case.
- **Conv5**: At Conv5, the similarity between BPLS-schedules 4421, 4422, and 4422_m to the base-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

line of 4442 is lower compared to Conv4. Nevertheless, the similarity between BPLS-schedule 4442 and the 4421 and 4422 baselines remains relatively high. These observations contradict the assumption of a direct correlation between the training frequency of certain layers and their corresponding learning rate.

- **Conv6 and Lin1**: Conv6 and Lin1 show very similar structures compared to their Distance counterparts.
- **Random Dropout**: Further, throughout the layers the Cosine-Similarity and Distance Polar Plots show similar patterns. At layers where baselines apply a fraction of their base learning rate, BPLS is more similar to the baselines, while at layers where baselines apply their base learning rate, Random Dropout is more similar.
- **Up-Scaled**: Aside from that, the up-scaled variant of 4442 is the only showing high similarity to Conventional Training, while the other up-scaled training approaches show relatively low similarity to Conventional Training.

Color Maps - Distance

The Polar Plots provided deep insights by comparing specific Distance and Cosine-Similarity values. However, they are limited to illustrating the similarity between training approaches and their baselines. Similarities between different BPLS-schedules or between BPLS-schedules and their Random Dropout counterparts cannot be directly derived from the Polar Plots.

The Color Maps provided in Figures 7.3 and 7.4 display the similarity between all training approaches. It is important to note that the Color Maps completely ignore specific values and only describe the relative order. Therefore, when the subsequent text refers to similarity, it is always referring to relative similarity. Following, each category of training approaches is analyzed one after the other:

- **Per-Layer Learning Rate**: The Per-Layer Learning Rate variants were used as baselines in the Polar Plots. Some previously observed aspects are also evident in the Color Maps. For each Per-Layer Learning Rate approach, their BPLS-counterparts show very high similarity at Conv4 and Conv5, while their Random Dropout counterparts show very high similarity at Conv6 and Lin1. Additionally, it can be seen that the Per-Layer Learning Rate approach 4444_lr4421 shows high similarity to 4444_lr4422 not only at Conv5, where both configurations are identical, but across all layers. 4444_lr4422 on the other side shows high similarity to 4444_lr4442 at Conv4 and to 4444_lr4421 at all other layers, indicating a high degree of similarity within the Per-Layer Learning Rate category.
- **BPLS**: When examining the BPLS-schedules, it becomes apparent that especially at layers where

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

training steps are skipped the similarity to training approaches with equivalent configurations is highest. Only Random Dropout does not follow this pattern. Particularly at layer Conv4, the similarity between BPLS and Random Dropout counterpart is relatively low. At Conv6 and Lin1, BPLS exhibits increased similarity to Random Dropout, consistent with observations made for the Per-Layer Learning Rate approach.

Taking a closer look at BPLS-schedule 4422, we can observe the following: at Conv4, 4422 skips every 2nd training step. The four most similar training approaches at this layer are 4444_lr4422, 4442, 4444_lr4442, and 4422_m. All of these approaches either skip every second training step or apply half their base learning rate at this layer, which aligns with expectations. Similar can be observed at Conv5.

- **Random Dropout**: The Polar Plots indicated a relatively low similarity between Per-Layer Learning Rate and their Random Dropout counterparts. However, the Color Maps reveal that Random Dropout is actually most similar to its Per-Layer Learning Rate and BPLS counterparts, with some minor exceptions.

  Looking at 4422_rdo, we observe the following: at Conv4, the most similar approaches are 4444_lr4422, 4422, and 4444_lr4442, followed by 4421_rdo, 4442, and 4422_m. This generally meets expectations, except for the unexpectedly high similarity of 4421_rdo. One reason for this could be a relatively high level of similarity within the Random Dropout category. On the other hand, when examining the plot for 4442_rdo, we notice that at Conv4 even 4444_lr4421 is more similar than 4422_rdo, contradicting our previous assumption.

- **Up-Scaled**: Almost all investigated training approaches show very low similarity to the upscaled category. The 4442 training approaches are the only ones that exhibit similarity to their upscaled counterpart. While this is expected, the actual question is how similar the upscaled training approaches are to each other and to Conventional Training.

  With some exceptions, the similarity within the up-scaled category is relatively high. While especially Conv4 behaves as expected in most cases, 4422_4444 shows unexpectedly low similarity to 4422_m_4444 at this layer. Looking at subsequent layers, it can be seen that none of the upscaled training approaches are similar to 4422_m_lr4444 at those layers. On the other hand, 4422_m_lr4444 exhibits especially low similarity to 4421_lr4444 and 4422_lr4444.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

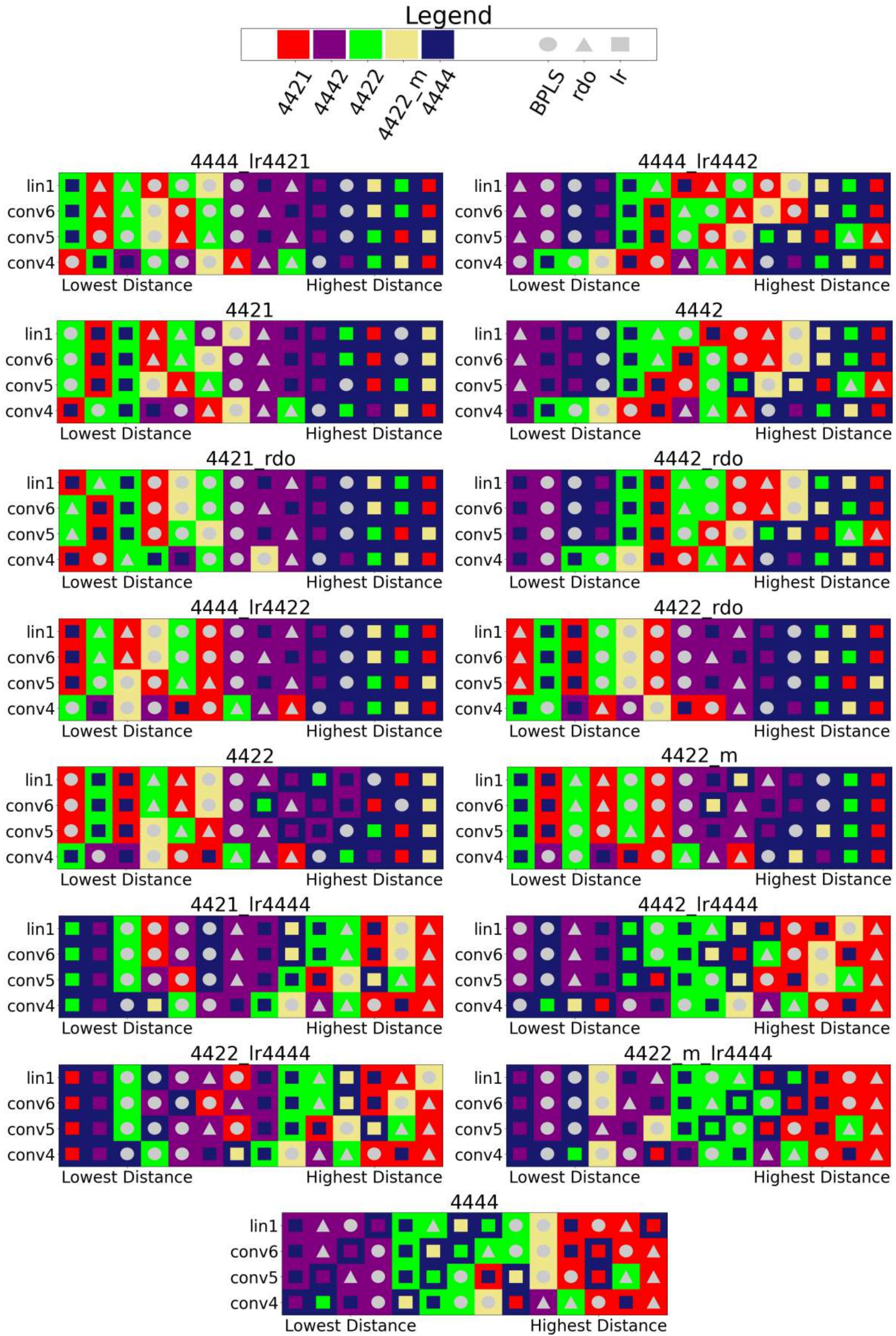


Figure 7.3: Cifar10 Network - Distance Color Maps

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- **Conventional Training**: The plot featuring Conventional Training shows relatively clear clusters. The 4442 variants exhibit the highest similarity, as expected, while the 4422 variants show medium similarity and the 4421 variants low similarity. Surprisingly, the similarity to upscaled training approaches is unexpectedly low.
- **Summary**: Overall, especially at layers where BPLS skips training steps, the similarity between Per-Layer Learning Rate and BPLS meets expectations. In the case of Random Dropout, Up-Scaled, and Conventional Training, the expectations are only partially met.
  Furthermore, while all 4422 and 4421 training approaches show similar patterns throughout the layers, the 4442 training approaches differ. In return, this suggests that regardless of the applied training technique, 4422 and 4421 are relatively similar to each other.
  On the other hand, at Conv4, 4422 and 4442 (Per-Layer Learning Rate and BPLS) show very high similarity to each other. This is likely due to sharing the same configuration at this layer. Nevertheless, both training approaches are more similar to their own counterparts than to each other. Conv5 is the only layer that distinguishes between the configurations of 4422 and 4442. The Color Maps accordingly show how the configuration of single layers can affect training behavior.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Color Maps - Cosine-Similarity

Following, Figure 7.4 displays Color Maps for the Cosine-Similarity metric. While the overall structure resembles the previously shown Distance Color Maps, there are some differences that need to be pointed out:

- **Per-Layer Learning Rate**: Just as at the Distance Color Maps, the Per-Layer Learning Rate shows high similarity to its BPLS counterpart. However, here at Conv4, 4444_lr4421 shows the highest similarity to 4444_lr4422 and vice versa. Besides that, to 4444_lr4442, 4422 is less similar than 4444_lr4421. In other words, the similarity within the Per-Layer Learning Rate category seems to be even higher compared to the Distance metric.
- **BPLS**: At the BPLS-schedules, similar can be observed. While the patterns are mostly as expected, especially at Conv4, 4421 shows the highest similarity to 4422 instead of one of its counterparts. At 4422, a relatively high similarity to 4421 can be observed as well. Additionally, 4422 and 4442 show very low similarity to 4422_m, and 4442 seems to be very similar to its up-scaled counterpart.

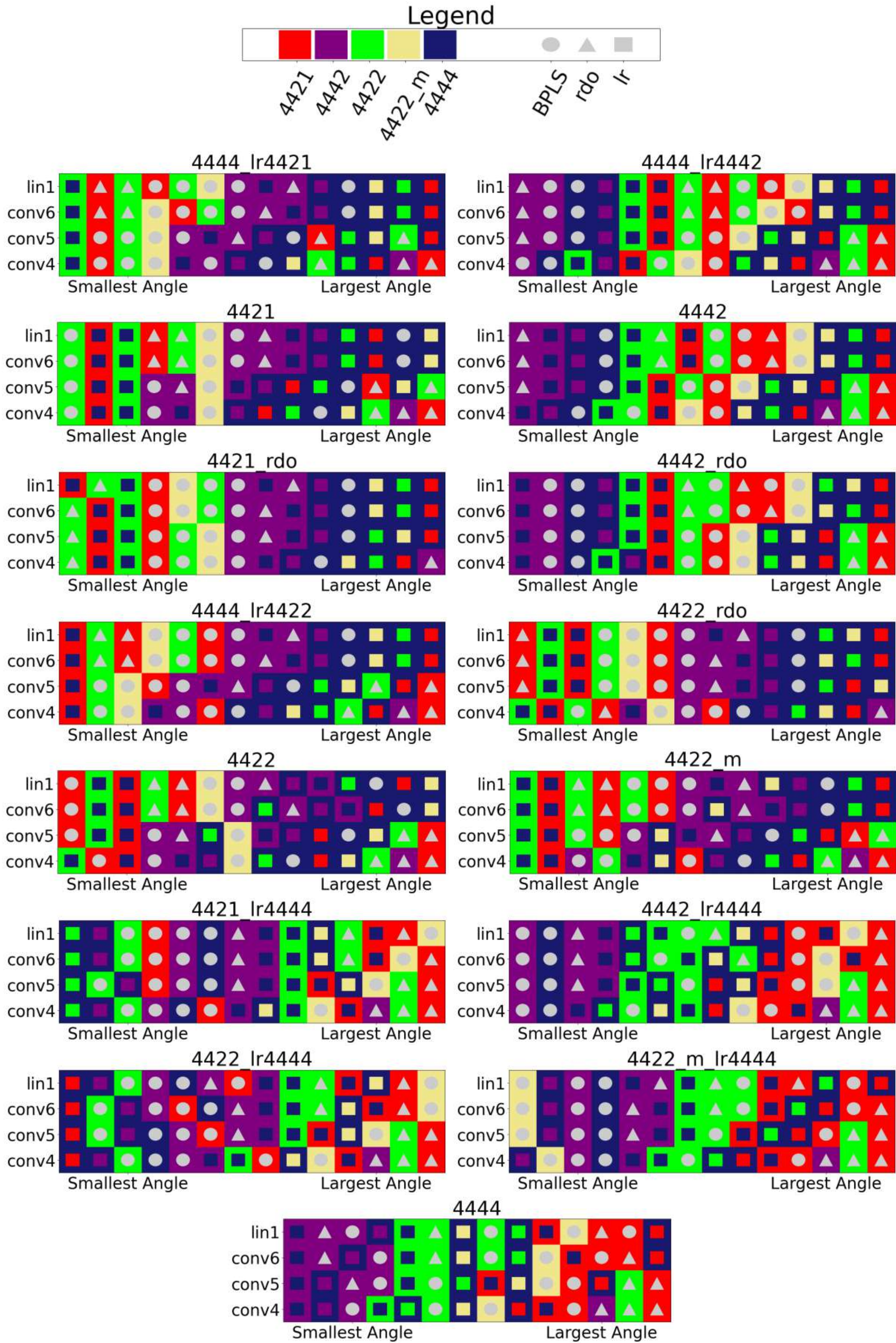


Figure 7.4: Cifar10 Network - Angle Color Maps

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- **Random Dropout**: Here, the similarity between Per-Layer Learning Rate and their Random Dropout counterparts is even lower compared to the Distance Color Maps. However, Random Dropout is still most similar to their Per-Layer Learning Rate and BPLS counterparts, with some exceptions. For example, at Conv4 4421_rdo is most similar to 4422_rdo and more similar to 4444_lr4422 then to 4421. On the other side, 4422_rdo shows high similarity to 4444_lr4421 and 4421_rdo. This indicates a high similarity within the Random Dropout category. However, 4421_rdo and 4422_rdo are least similar to 4442_rdo and the other way around, contradicting this assumption.
- **Up-Scaled**: The up-scaled variants show similar patterns compared to the Distance Color Maps too. The largest difference is, that 4422_m_lr4444 shows especially high similarity to 4422_m while the similarity to other up-scaled BPLS-schedules is even lower as at its distance counterpart.
- **Summary**: When comparing the Cosine-Similarity and the Distance Color Maps more generally, several aspects stand out. In the Cosine-Similarity Color Maps, the sequence of the most similar training approaches varies less throughout the layers compared to the Distance Color Maps. Additionally, a slightly higher similarity within the Per-Layer Learning Rate and BPLS categories can be observed. However, this is not the case for Random Dropout, where the similarity within its category appears to be lower compared to the Distance Color Maps.

UMAPs

Figure 7.5 displays weights of different training approaches and epochs mapped to 2D points by UMAP. The baselines used here are the same as those used for the polar plots: 4444, 4444_lr4421, 4444_lr4422, and 4444_lr4442. UMAP generates four clusters. These clusters not only contain the baselines but also their counterparts, as expected. For example, BPLS-schedule 4421 and its Random Dropout counterpart (4421_rdo) are mapped to the corresponding Per-Layer Learning Rate baseline (4444_lr4421). The same applies to BPLS-schedules 4422, 4422_m, and 4442 and their Random Dropout counterparts. Additionally, all up-scaled variants are mapped to 4444. This clustering remains consistent across all layers.

Moreover, at Conv4, while the 4421-cluster points in one direction individually, all other clusters initially show similarity for the first few epochs before diverging. Comparing the schedules, it is evident that 4421 trains Conv4 every 4th training step, while 4422 and 4442 train the layer every 2nd step, and 4444 trains it every single step. While a certain similarity between 4422 and 4442 is expected, it is interesting that the 4444-cluster also shows some similarity to them.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Similar behavior is observed at Conv5. While 4421 and 4422 train Conv5 every 2nd step, 4442 and 4444 train it every single step. Consequently, it is not surprising that initially, 4421 and 4422 point in one direction while 4442 and 4444 point in another direction for the first few epochs. Layers Conv6 and Lin1 are trained by all training approaches in the same way. Nonetheless, the mapping still follows the structure observed at Conv5. This indicates once more that the training configuration of layers Conv4 and Conv5 affects the weights of subsequent layers.

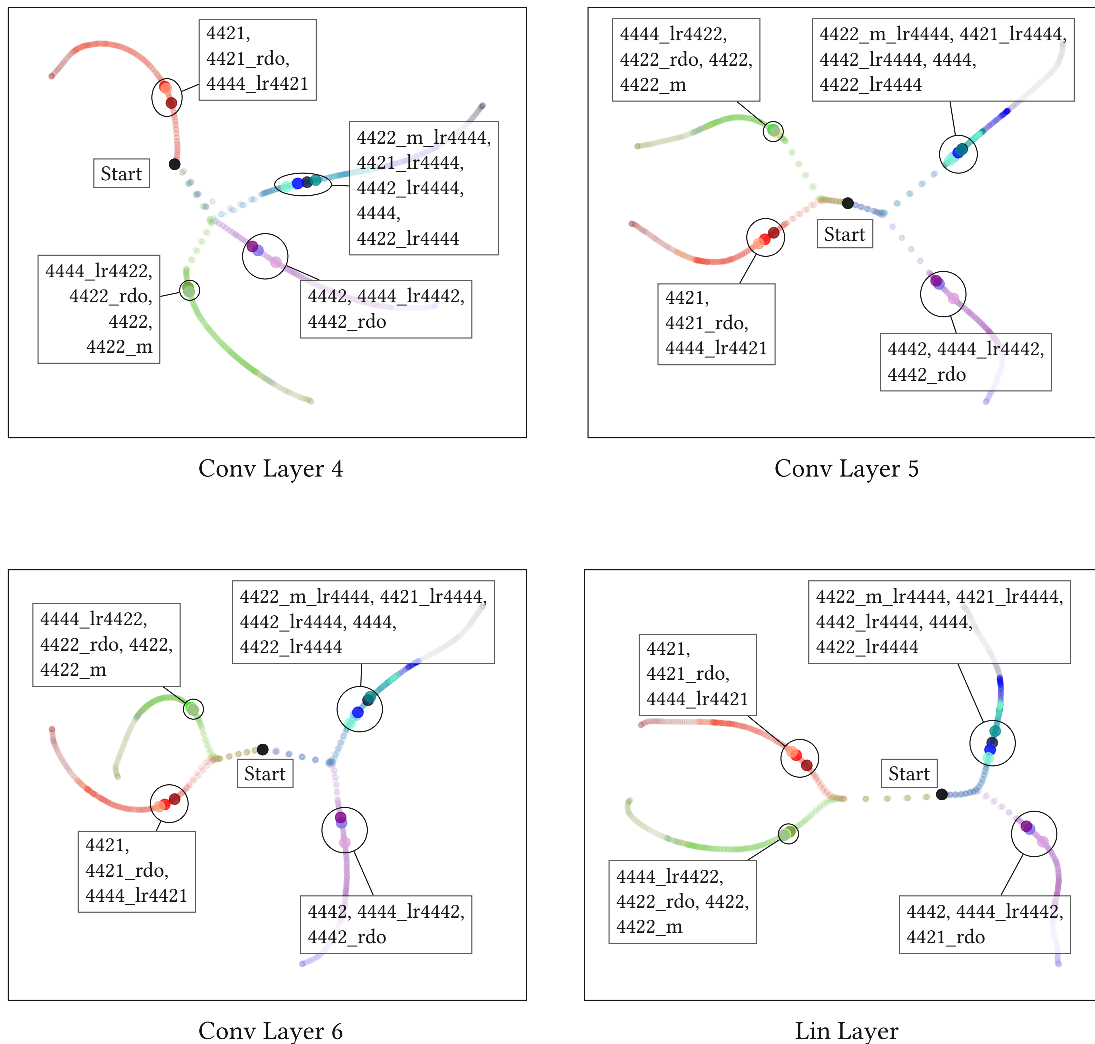


Figure 7.5: Cifar10 Network - UMAPs

7.2.3 Cifar100 - Scenario

To deepen the understanding of the similarity between training approaches, the previously discussed investigations were conducted for the Cifar100 scenario as well. Here, the base learning rate was set to 200E-6.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Following, UMAPs and Distance Polar Plots are drawn. The UMAPs follow the same structure as the Cifar10 UMAPs. Due to their clarity, they are drawn first. Subsequently, two variations of Polar Plots are shown: first, with Per-Layer Learning Rate training approaches defined as the baseline, followed by Polar Plots using BPLS-schedules as the baseline. To maintain manageability, Color Maps are not provided for the Cifar100 scenario.

In the Cifar100 scenario, training involved 6 layers. However, most training configurations primarily differ in the first two layers. As observed in the Cifar10 scenario, these initial layers are particularly significant, while subsequent layers tend to follow similar patterns.

UMAPs

Figure 7.6 presents the UMAPs corresponding to the Cifar100 scenario. Upon closer examination of the first unfrozen layer, Conv7, it is evident that almost all training approaches are mapped to their expected baselines, with the exception of BPLS-schedule A98765. Instead of being mapped to the A98765 cluster, UMAP assigns it to the 444442 cluster. Upon comparing the configurations of A98765 and 444442, it becomes noticeable that both train Conv7 every 2nd step, making the decision by the UMAP tool comprehensible.

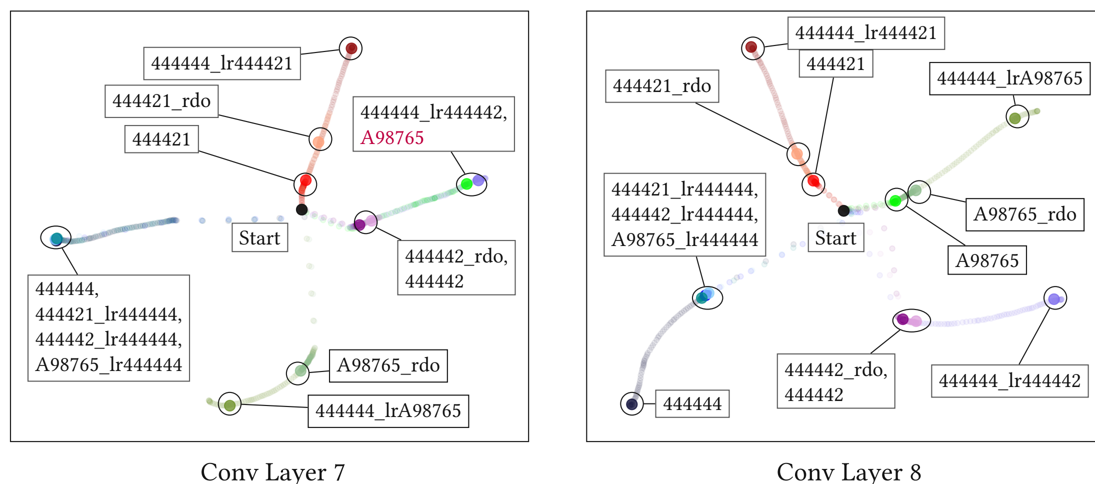


Figure 7.6: Cifar100 Network - UMAPs

For Conv8, the training approaches were mapped as expected. However, upon closer inspection of BPLS-schedule 444442 and its Random Dropout counterpart, an increased similarity to A98765 can be detected. For the first few epochs, these training approaches are mapped to the A98765 baseline before transitioning to the 444442 baseline in later epochs.

The clustering for all subsequent layers (Conv9 - Lin3) align with expectations. Accordingly, the corresponding UMAPs are not displayed here.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Both displayed plots exhibit an increased gap within the clusters compared to the Cifar10 scenario. This suggests a higher difference between the training approaches, indicating a wider distance range in the subsequent Polar Plots.

Polar Plots - Distance

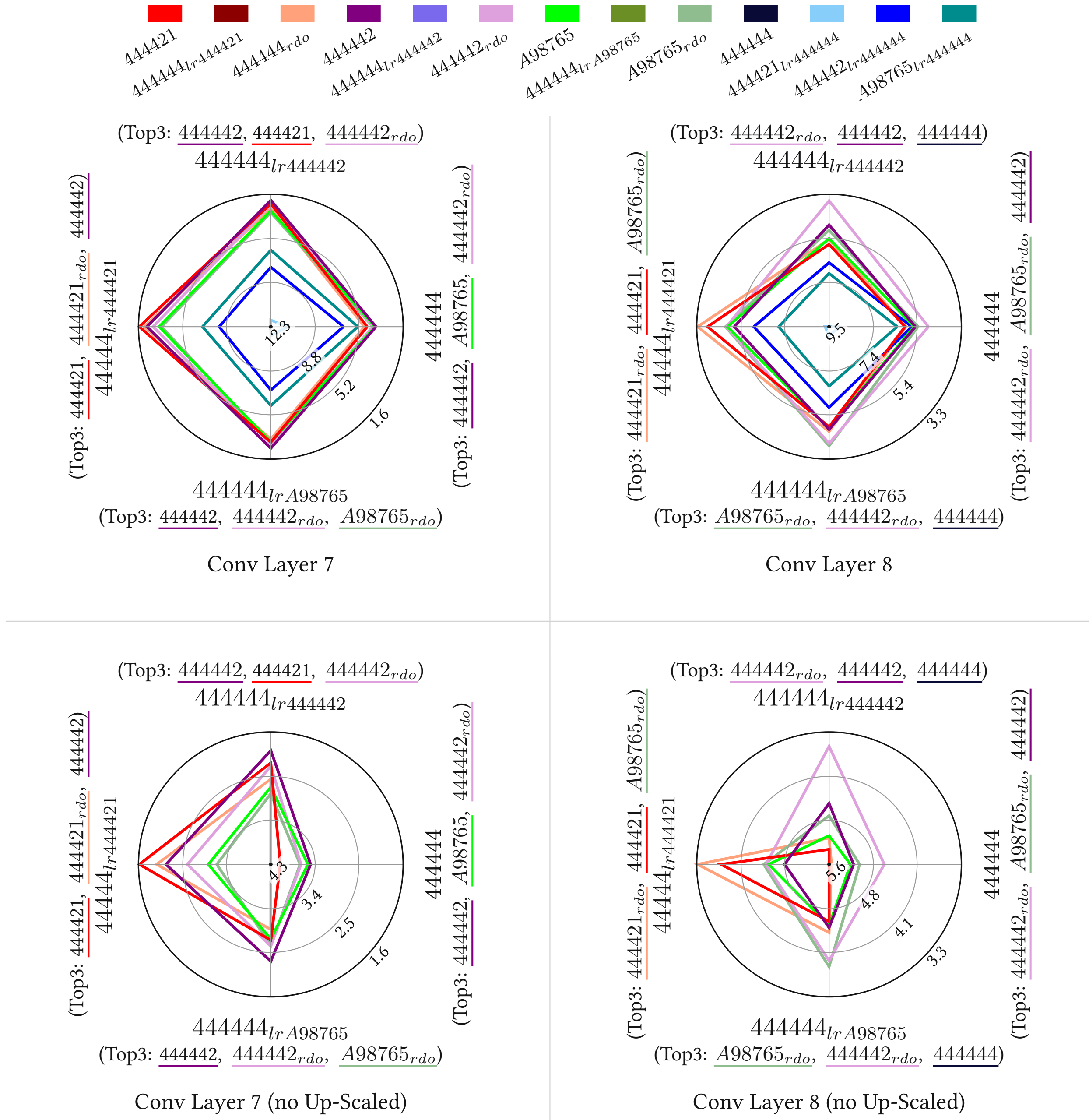


Figure 7.7: Cifar100 Network - Distance Polar Plots (Baseline: Per-Layer Learning Rate)

Figure 7.7 displays the corresponding Distance Polar Plots using the same baselines as the UMAPs. As expected, there is a relatively high distance (low similarity) between the baselines and the Up-Scaled variants. To enhance readability, the Polar Plots are presented twice. The upper row includes the Up-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Scaled variants, while they are omitted in the lower row. The plots reveal the following insights:

- **Range**: The distance range at Conv7 is 10.7 and at Conv8 6.2. Just as expected due to the UMAPs, those values are significantly larger compared to the Cifar10 scenario. At the Cifar10 scenario, the first two layers had distance ranges of 2.2 and 3.2. Accordingly, the training approaches vary significantly more at the Cifar100 scenario. When ignoring the Up-Scaled variants, the distance ranges shrink to 2.7 at Conv7 and 2.3 at Conv8.
- **Conv7**: The baselines corresponding to the schedules 444421 and 444442 show relatively high similarity to their BPLS counterparts. However, especially at Conv7, the baseline corresponding to A98765 shows higher similarity to 444442 than to its corresponding BPLS-schedule. On the other hand, the baseline of 444442 shows relatively low similarity to A98765. In fact, the similarity between the 444442 baseline and 444421 is even higher, despite that the configuration of 444421 at layer Conv7 differs more from 444442 than A98765 does.
- **Random Dropout**: Conv7 and Conv8 represent the layers where training approaches differ the most. In contrast to the Cifar10 scenario, where corresponding layers showed very low similarity to Random Dropout, the situation here is different. At Conv7, the similarity between the baselines and Random Dropout is only slightly lower compared to BPLS, while at Conv8, the similarity to Random Dropout is even higher.
- **Up-Scaled**: Each baseline exhibits low similarity to the Up-Scaled training approaches, similar to observations in the Cifar10 scenario. However, especially at Conv7, the Up-Scaled training approaches appear to be most similar to Conventional Training compared to other baselines.

Figure 7.8 switches the baselines from Per-Layer Learning Rate to BPLS. The layers investigated are again Conv7 and Conv8. Following can be seen:

- **Range**: The distance ranges are almost identical to those observed in the previous Polar Plots, where Per-Layer Learning Rate was used as baseline. On one hand, this is not surprising. The highest similarity (lowest distance) at Conv7 is observed between 4421 and its Per-Layer Learning Rate counterpart. Accordingly, switching the baselines does not affect this value. On the other hand, the Up-Scaled training approaches show just as little similarity to BPLS as they do to the Per-Layer Learning Rate. Particularly, the similarity to 444421_lr444444 is very low.
- **444442**: While 444421 shows expected patterns, 444442 and A98765 differ. At Conv7, 444442 is most similar to its Per-Layer Learning Rate counterpart. At Conv8, the similarity is highest to the A98765 Per-Layer Learning Rate variant closely followed by 444442 Random Dropout and Per-Layer Learning Rate.
- **A98765**: The A98765 baseline on the other side is most similar to the Per-Layer Learning Rate approaches 444442 and A98765 at Conv7. At Conv8 it is most similar to the Per-Layer Learning

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

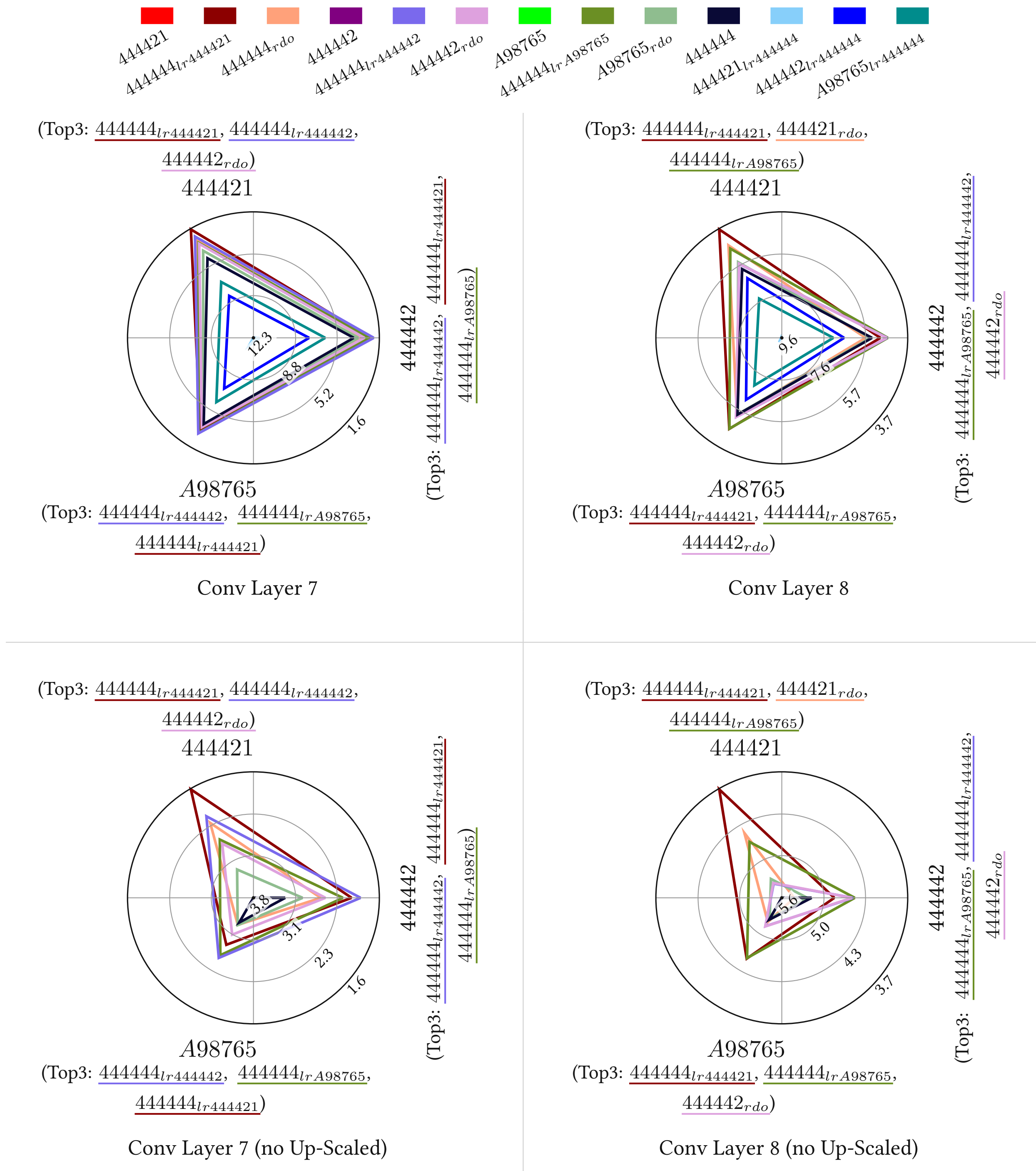


Figure 7.8: Cifar100 Network - Distance Polar Plots (Baseline: BPLS)

Rate approaches 444421 and A98765. In both cases, the A98765 baseline shows highest similarity to (almost) identically configured training approaches.

- **Random Dropout**: When looking carefully, it can be seen that at Conv8 the pattern of Per-Layer Learning Rate approach 444442 is almost identical to its Random Dropout counterpart. None of the previous investigations showed comparable behavior. Overall, in comparison to Per-Layer Learning Rate, BPLS seems to be less similar to Random Dropout.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- **Up-Scaled**: While the similarity between 444421 and Up-Scaled is relatively low, A98765 shows the highest similarity to the Up-Scaled variants. In case of Conventional Training, A98765 and 444442 show approximately the same similarity. The similarity to 444421 is lower.

7.2.4 Summary

Especially, the mapping performed by the UMAP tool shows the similarity between Per-Layer Learning Rate and its counterparts clearly. The clustering was as expected with one small exception. At Layer Conv7 of the Cifar100 scenario, the BPLS-schedule A98765 was mapped to the 444442 Per-Layer Learning Rate baseline. Taking a closer look at the underlying configurations reveals that Conv7 is trained every 2nd step in both cases. In other words, both BPLS-schedules, A98765 and 444442, are equally configured at layer Conv7.

In general, it seems that two training approaches sharing the same configuration at a specific layer tend to show similar patterns at this layer. For example, BPLS-schedules 4442 and 4422 at layer Conv4 of the Cifar10 scenario. This can especially be seen in the corresponding Distance Polar Plot.

Still, during the investigations, it became clear that layers influence each other. This can especially be seen in the Color Maps of the Cifar10 scenario. While Conv6 and Lin1 share the same configuration in each training approach, they continue the pattern visible at layers where configurations between training approaches differ. For example, in the Cifar10 scenario, even when the configurations of Per-Layer Learning Rate 4422 and 4421 apply the same learning rate at Conv6 and Lin1, they show increased similarity to their own BPLS and Random Dropout counterparts at those layers.

In general, it was observed that especially at Cifar10, the similarity between Per-Layer Learning Rate and Random Dropout is significantly lower at layers differing from Conventional Training than at later layers. This indicates that even when both training approaches behave differently at those specific layers, both influence later layers similarly. One example of this can be seen in the Cosine-Similarity Color Map of the Cifar10 scenario, where 4421 Per-Layer Learning Rate shows very low similarity to its Random Dropout counterpart at Conv4 and Conv5 but very high similarity at Conv6 and Lin1.

Besides that, it has to be pointed out that Random Dropout, with some exceptions, is most similar to the corresponding Per-Layer Learning Rate and BPLS counterparts throughout the layers. This could be the reason why UMAP maps them as expected.

While the Up-Scaled training approaches show very low similarity to each investigated baseline, they are most similar to Conventional Training, as expected. This can especially be seen in the UMAPs of the Cifar10 and Cifar100 scenarios and in the Color Maps of the Cifar10 scenario.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

7.3 Performance Evaluation

After analyzing the similarity between BPLS, Per-Layer Learning Rate, and Random-Dropout, it is time to answer the second research question: *Can BPLS improve the efficiency of fine-tuning without affecting training behavior negatively?*

Therefore, for each fine-tuning scenario, this section lists the highest achieved test accuracy, the base learning rate applied to achieve the highest test accuracy, and the estimated number of executed MAC-operations. Additionally, the estimated reduction in peak memory and the maximum number of operations executed per training step are listed for memory-optimized BPLS-schedules.

Some of the BPLS-schedules and their Per-Layer Learning Rate counterparts are visualized. The visualization allows for comparing test accuracy and training time directly in terms of number of epochs and MAC-operations.

The scenarios are not presented in any specific order but are simply listed based on their execution. While the Cifar10 scenario was the first investigated, the Cifar100 scenario was the second, and the Key-Word scenario the third.

The accuracy was measured following the Grid Search approach introduced in the Methodology chapter. It is worth mentioning that even when searching with very fine steps, the best performing learning rate and highest test accuracy should be viewed in relation to the limitations of the grid search conducted. Additionally, performing the same training with slightly modified parameters (variation of pre-trained network, learning rate, etc.) could lead to slightly different results. Accordingly, a small tolerance around the test accuracy should be considered. Test accuracies that differ only slightly can be considered equivalent.

7.3.1 Interpretation

To understand the following tables and diagrams, some explanations are provided here.

Tables

Table 7.1, 7.2, 7.3 and 7.4 summarize the performance evaluation of the Cifar10, Cifar100, and Key-Word scenarios. For each investigated training approach, these tables contain the highest achieved accuracy and the corresponding learning rate. Each training was performed for a defined number of epochs.

As mentioned, accuracies that only slightly differ should be considered as equal. Accordingly, a learning rate range is provided. This range contains learning rates that result in accuracies differing

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

only slightly from the highest achieved accuracy. The range starts at the lowest and ends at the highest learning rate that leads to sufficient training. Learning rates inside the range that result in an accuracy too low do not affect the range borders.

Additionally, the tables contain the average number of MAC-operations per training step (training of one single data sample) and its relation to Conventional Training, which of course executes the highest number of MAC-operations per training step.

Diagrams

Diagrams 7.9, 7.10, 7.12 and 7.14 visualize some of the training approaches. The diagrams are structured as follows: the left side shows accuracy in relation to the number of training epochs, while the right side shows accuracy in relation to the number of MAC-operations executed. Most plots compare a BPLS-schedule with its corresponding Per-Layer Learning Rate counterpart. Each plot includes the best-performing conventional fine-tuning run in two variants: one with n frozen layers and one with $n+1$ frozen layers, where n represents the optimal number of layers to freeze. This demonstrates how freezing layers affects training.

In addition, diagrams 7.11, 7.13 and 7.15 plot the number of epochs required for reaching a specific accuracy in relation to the reduction in operations. The diagrams do not necessarily include the training runs achieving the highest accuracy but rather those performing well in terms of accuracy and training time. These training runs fall within the previously introduced learning rate range. The complete grid search outcomes are listed in the Appendix section.

7.3.2 Cifar10 Performance

Training Conditions

In the case of the Cifar10 scenario, training was performed for 30 epochs. Approximately 20 training runs were executed for each training approach. In this experiment no Plato Learning Rate Schedule or Dropout Layer was utilized.

Overview

Table 7.1 provides an overview of the training performance corresponding to this experiment. Several important aspects should be highlighted:

- There is a noticeable gap in test accuracy between 4444 and 4440, indicating that training layer Conv4 significantly impacts performance.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

schedule	learning rate [E-6]	best test accuracy	learning rate range (0.1%-points) [E-6]	ops per step	rel. ops
4444	133	78.17%	83 – 146	17.24M	100.00%
4440	140	77.72%	100 – 140	14.06M	81.56%
4444_{lr4442}	154	78.13%	100 – 205	17.24M	100.00%
4444_{lr4422}	169	78.09%	169 – 251	17.24M	100.00%
4444_{lr4421}	274	77.81%	100 – 332	17.24M	100.00%
4444_{lr4432}	225	78.10%	140 – 225	17.24M	100.00%
4442	96	78.25%	87 – 196	15.65M	90.78%
4422	207	78.15%	186 – 225	14.15M	82.09%
4422_m	225	78.17%	225 – 225	15.14M	87.84%
4421	196	78.10%	116 – 274	13.35M	77.48%
4432	154	78.20%	116 – 154	14.90M	86.44%
4211_m	179	78.15%	179 – 179	12.09M	70.14%

Table 7.1: Cifar10 - Training Performance Summary

- 4421 performs fewer operations than 4440 but achieves higher test accuracy. 4422 requires slightly more operations than 4440 but achieves a test accuracy practically as good as 4444 while executing significantly fewer operations.
- BPLS-schedules 4422_m and 4211_m are optimized for peak-memory and peak-operations reduction. According to mathematical estimations, 4422_m can reduce peak-memory by 6.31% and 4211_m by 14.01%. The peak-operations are 94.12% at 4422_m and 82.38% at 4211_m. Both schedules reduce the number of overall executed operations significantly while achieving good accuracy. In comparison, the Conventional Training approach 4440 reduces peak-memory by 7.35% but achieves significantly worse accuracy. 4440 executes 81.56% operations at each training step.
- The Per-Layer Learning Rate approach achieves slightly lower test accuracy than BPLS at the investigated configurations but performs as many operations as 4444.
- As expected, the Per-Layer Learning Rate approach allows for higher base learning rates. In case of 4421, the base learning rate of the Per-Layer Learning Rate variant can be up to 21% higher than the learning rate of its BPLS counterpart. In case of 4422 and 4422_m, it can be up to 12%, and in case of 4432, up to 46% higher. However, for 4442, the base learning rate of its Per-Layer Learning Rate counterpart can only be up to 5% higher, possibly limited by layer Conv5, allowing no higher base learning rate without negatively affecting test accuracy. This comparison is based on the upper bound of the learning rate range.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Diagrams

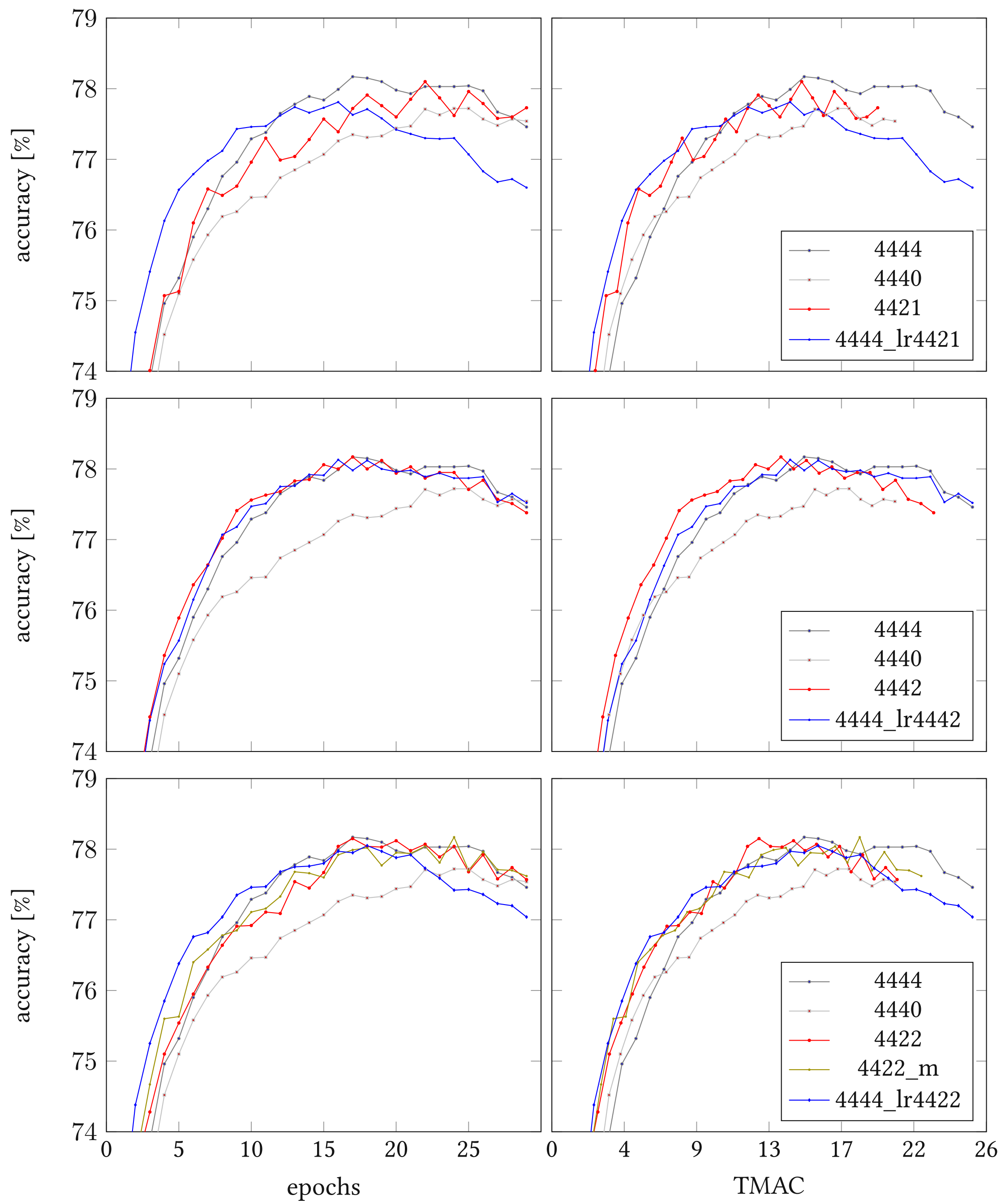


Figure 7.9: Cifar10 Network - Training Performance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Figure 7.9 visualizes the training of BPLS-schedules 4421, 4442, 4422 and 4422_m and their Per-Layer Learning Rate counterparts.

In the upper-left plot (featuring 4421), it can be observed that the Per-Layer Learning Rate variant achieves a relatively high test accuracy, requiring the fewest epochs, as expected. However, the achieved test accuracy is slightly lower compared to the conventional approach. In contrast, BPLS requires more epochs to reach its peak compared to the conventional approach, possibly due to frequent layer skipping. Nonetheless, the test accuracy of 4421 is very close to that achieved by Conventional Training. Additionally, there are noticeable fluctuations in accuracy with the BPLS-schedule, likely also due to frequent layer skipping.

The upper-right plot presents the same data but switches the x-axis to the number of executed MAC-operations. Here, it can be observed that the reduction in operations allows the BPLS-schedule to train nearly as fast and energy efficiently as its Per-Layer Learning Rate counterpart while achieving test accuracy comparable to Conventional Training.

In the second row (featuring 4442), it can be observed that all displayed training runs, except for 4440, require approximately the same number of epochs to reach high test accuracy. This could be attributed to the configuration constraints that prevent assigning higher base learning rates to the Per-Layer Learning Rate compared to other training approaches. Additionally, BPLS-schedule 4442 trains layers Conv4 and Conv5 more frequently than schedule 4421, avoiding the situation where BPLS requires more epochs than Conventional Training. Looking at the right plot, despite BPLS-schedule 4442 performing more operations than schedule 4421, its overall performance is superior, outperforming both its Per-Layer Learning Rate counterpart and Conventional Training.

The last row features BPLS-schedules 4422 and 4422_m. As mentioned earlier, 4422_m has the potential to reduce peak memory by 6.31% and peak operations by 5.88%. Despite being optimized for memory reduction, 4422_m performs comparably in terms of training time and accuracy to its Per-Layer Learning Rate counterpart. When considering the number of operations required to achieve accuracy similar to the best achieved by Conventional Training, 4422 requires 17.91% fewer operations. This reduction directly translates to savings in training time and energy consumption, at least on the targeted MCUs.

For this experiment, no Plato Learning Rate Schedule was utilized. Accordingly, after reaching the accuracy peak, the learning rate was not reduced, leading to a decline in accuracy after the peak. While Figure 7.9 plots the actual accuracy of the network after each epoch, Figure 7.10 shows the best accuracy achieved up to a specific epoch. Additionally, instead of displaying all 30 executed training epochs, the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

left plots display only 20 epochs, while the right plots display 26.71 TMAC-operations, corresponding to the number of MAC-operations executed by Conventional Training to perform 20 epochs.

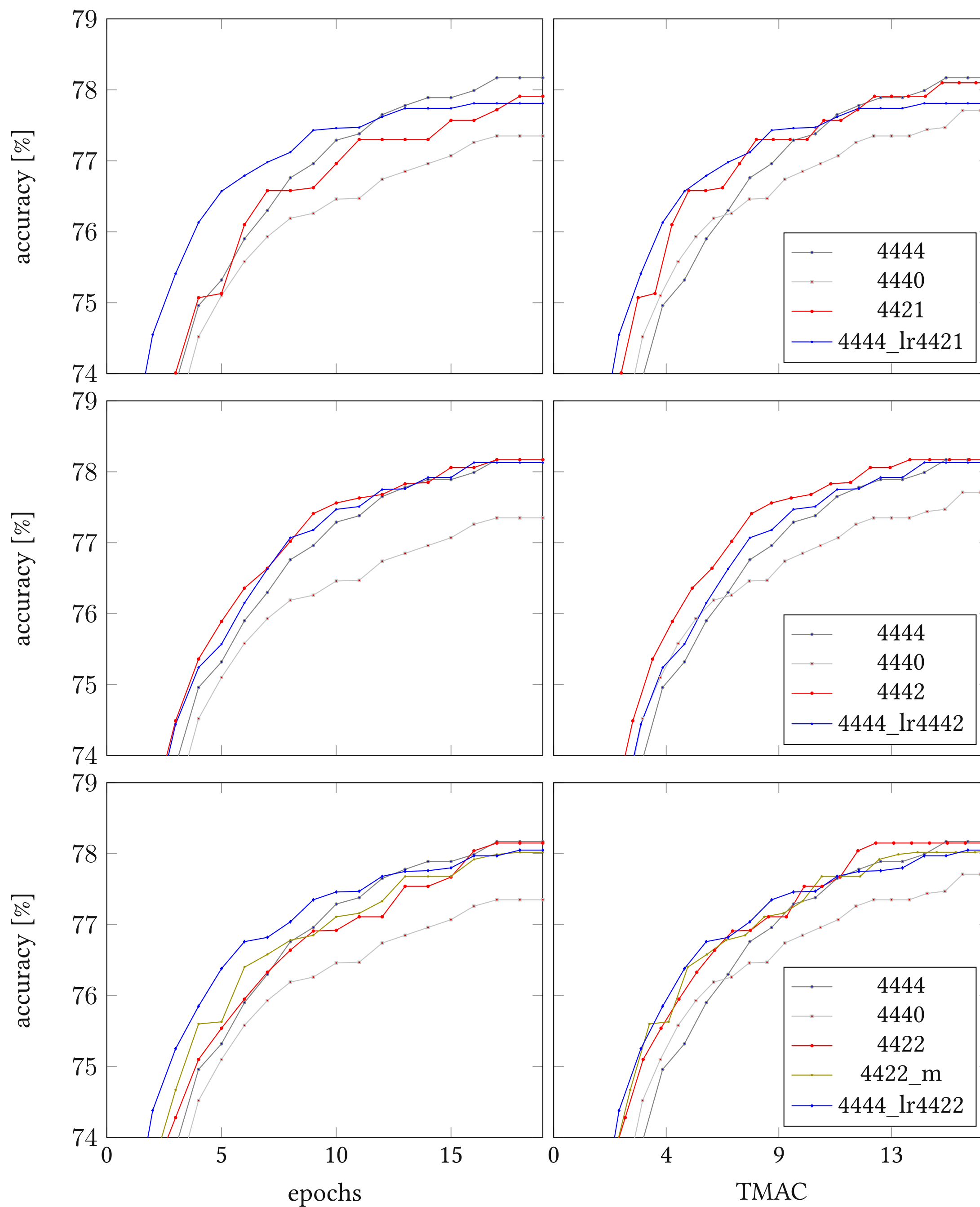


Figure 7.10: Cifar10 Network - Training Performance (best accuracy)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Figure 7.11 displays the number of epochs required to reach 77.8% accuracy in relation to the reduction in operations. Here, all BPLS-schedules listed in Table 7.1, including their highest achieved accuracy are displayed. The plot can be split into three parts. Until a 9.2% reduction in operations, no increase in epochs can be noted. Between 9.2% and 17.9%, a slight increase, and from 17.9% onwards, a high increase in epochs can be seen. The reduction in number of operations does only lead to minor

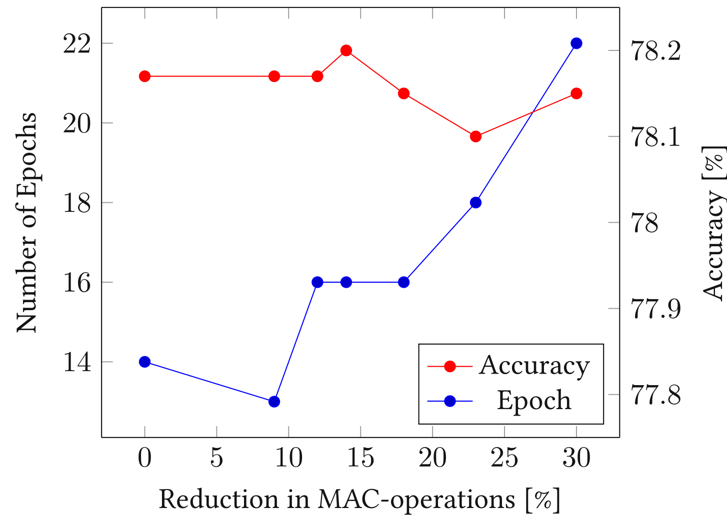


Figure 7.11: Cifar10 Network - Epochs per Operations

decline in accuracy.

7.3.3 Cifar100 Performance

Training Conditions

In the case of the Cifar100 scenario, the network was trained for 40 epochs. Approximately 18 different learning rates were investigated for each training approach. During this experiment, a Plato Learning Rate Schedule was employed, where if the test loss did not decrease for 5 consecutive epochs, the learning rate was halved.

Overview

The following tables summarize the training performance of the Cifar100 scenario. When working with datasets containing numerous classes, such as Cifar100, it is common to evaluate not only the top-1 accuracy but also the top-5 accuracy. In top-5 accuracy, a classification is considered correct if the ground truth class is among the top 5 predicted classes with the highest probabilities. Table 7.2 lists the top-1 training performance, while Table 7.3 contains the top-5 test accuracy and corresponding base learning rates. The tables reveal the following insights:

- BPLS achieves similarly high accuracies as their Per-Layer Learning Rate counterparts while performing fewer operations. However, Conventional Training performs surprisingly well in this experiment, especially considering the top-1 accuracy, outperforming all investigated BPLS schedules and their Per-Layer Learning Rate counterparts.
- As observed in the Cifar10 experiment, the Per-Layer Learning Rate approach allows for higher base learning rates than their BPLS counterparts, potentially leading to training that requires

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

schedule	learning rate [E-6]	best test accuracy	learning rate range (0.25%-points) [E-6]	ops per step	rel. ops
444444	1048	60.83%	1048 – 1048	301.44M	100%
444440	945	58.61%	945 – 1258	256.00M	84.93%
$444444_{lr444442}$	1134	60.37%	1031 – 1247	301.44M	100.00%
$444444_{lrA98765}$	1522	60.26%	1143 – 1522	301.44M	100.00%
$444444_{lr444421}$	1485	59.86%	1485 – 2009	301.44M	100.00%
444442	807	59.95%	596 – 960	278.72M	92.46%
A98765	969	60.06%	673 – 969	252.05M	83.61%
444421	807	59.81%	807 – 1162	247.11M	81.98%
444222_m	720	59.70%	538 – 720	244.18M	81.00%
663333_m	541	58.88%	541 – 595	224.14M	73.51%
633111_m	720	59.08%	720 – 720	231.72M	76.87%

Table 7.2: Cifar100 - Training Performance Summary - top-1

fewer epochs. Considering top-1 accuracy, the Per-Layer Learning Rate variant of 444421 allows base learning rates up to 73% higher than its BPLS counterpart, while for 444442 it is 30%, and for A98765 57%.

- The base learning rates resulting in good top-5 accuracy are lower for most training approaches compared to those achieving good top-1 accuracy. However, the relationship between BPLS and their Per-Layer Learning Rate counterparts regarding base learning rates is very similar for both, top-1 and top-5 accuracy. For top-5 accuracy, the Per-Layer Learning Rate variant of 444421 allows base learning rates up to 70% higher than its BPLS counterpart, while for 444442 it is 30%, and for A98765 73%.
- The network investigated here has high potential for peak-memory optimization due to the relatively high number of trained layers. While 444440 achieves a 7.78% reduction in peak-memory, 444222_m and 633111_m achieve 22.86% and 663333_m even 49.61%. The peak-operations are 90.52% at 444222_m, 82.38% at 633111_m, and 89.13% at 663333_m. In terms of accuracy, 663333_m, like 444440, performs relatively poorly. However, 444222_m achieves accuracy as high as other well-performing training approaches, and 633111_m performs somewhere in between.

Diagrams

Figure 7.12 visualizes the top-1 accuracy of several BPLS-schedules and their Per-Layer Learning Rate counterparts. Conventional Training with n and n+1 frozen layers is provided as a baseline, similar to the Cifar10 experiment. The training runs displayed represent a compromise between accuracy and training speed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

schedule	learning rate [E-6]	best test accuracy	learning rate range (0.25%-points) [E-6]
444444	716	85.17%	467 – 716
444440	807	83.8%	673 – 1258
$444444_{lr444442}$	1031	85.12%	467 – 1031
$444444_{lrA98765}$	1162	84.68%	673 – 1395
$444444_{lr444421}$	1162	84.58%	1162 – 1485
444442	794	84.94%	596 – 794
A98765	673	84.91%	673 – 807
444421	873	84.39%	561 – 873
444222_m	447	84.99%	447 – 720
663333_m	541	84.46%	406 – 541
633111_m	720	84.81%	720 – 720

Table 7.3: Cifar100 - Training Performance Summary - top-5

First of all, in each case, Conventional Training performed the best, while BPLS and the Per-Layer Learning Rate approach performed approximately equally well.

For 444421, BPLS achieves practically the same accuracy as its Per-Layer Learning Rate counterpart, albeit requiring slightly more epochs to reach its peak. However, because BPLS performs fewer MAC-operations per training step, it actually reaches the peak with fewer MAC-operations than its Per-Layer Learning Rate counterpart.

In the case of A98765, both the BPLS-schedule and its Per-Layer Learning Rate counterpart achieve slightly higher accuracy than in the case of 444421. Here, the Per-Layer Learning Rate approach requires a relatively low number of epochs for training. Consequently, the epoch-gap between the BPLS-schedule and its Per-Layer Learning Rate counterpart is too wide to be compensated by the savings in MAC-operations offered by BPLS.

The last row features the peak-memory optimizing BPLS-schedules 444222_m and 663333_m. As mentioned earlier, 444222_m allows a reduction in peak-memory by 22.86%, while 663333_m achieves an impressive reduction of 49.61%. In comparison, 444440 reduces peak-memory by only 7.78%. Despite this, in terms of accuracy and training speed, especially 444222_m performs relatively well, whereas 663333_m achieves performance similar to 444440.

This experiment demonstrates how the performance of BPLS depends heavily on the specific training scenario. The BPLS-schedules investigated here did not achieve advantages in terms of accuracy and training speed compared to the Per-Layer Learning Rate approach or Conventional Training. However, the peak-memory optimizing BPLS-schedules yielded very good results.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

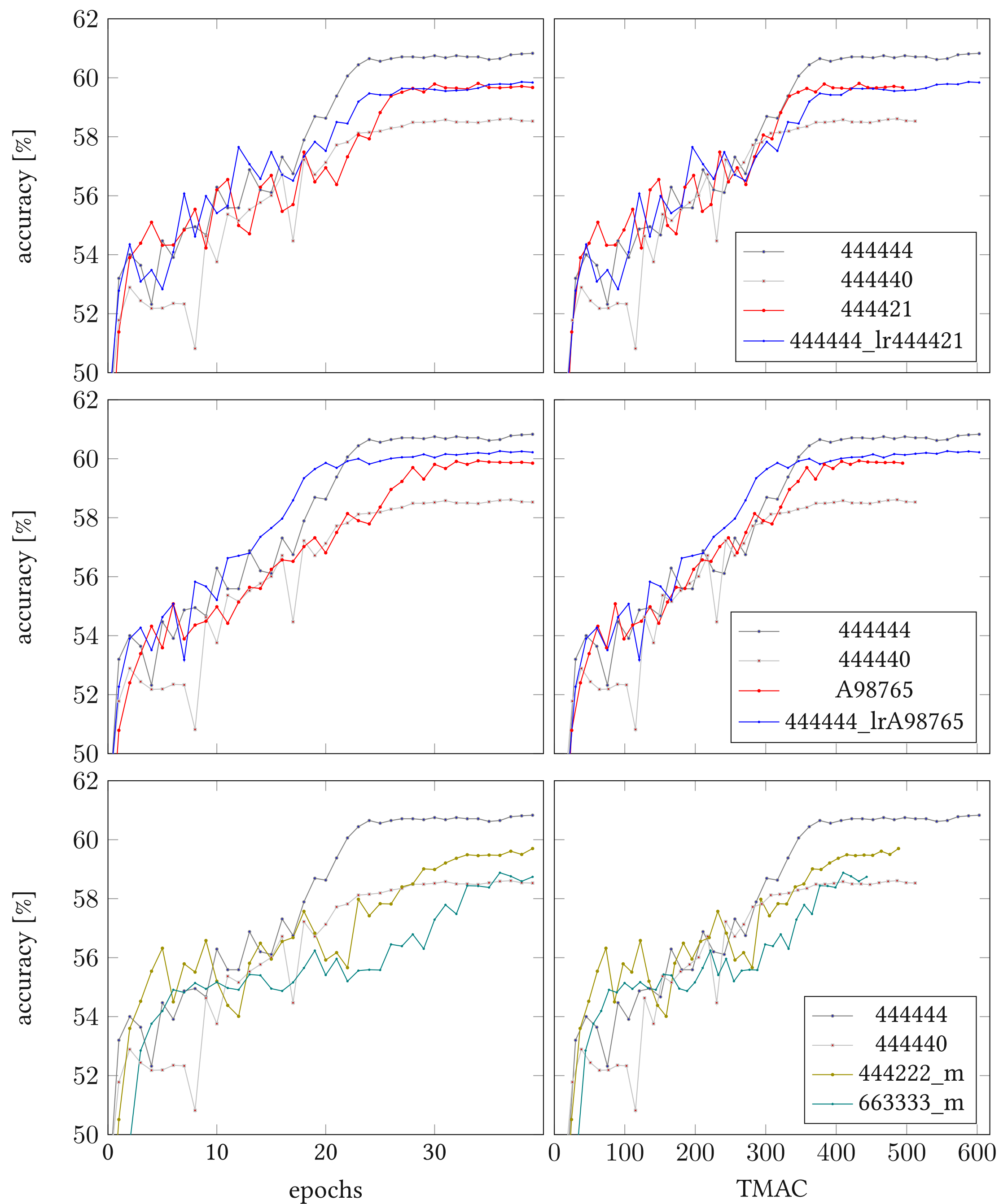


Figure 7.12: Cifar100 Network - Training Performance

Figure 7.13 displays the number of epochs required to reach 58.0% accuracy in relation to the reduction in operations. Up to a 7.5% reduction, no increase in epochs can be noted. From there on, the number of epochs steadily increases. Here, a minor but continues decline in accuracy can be seen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

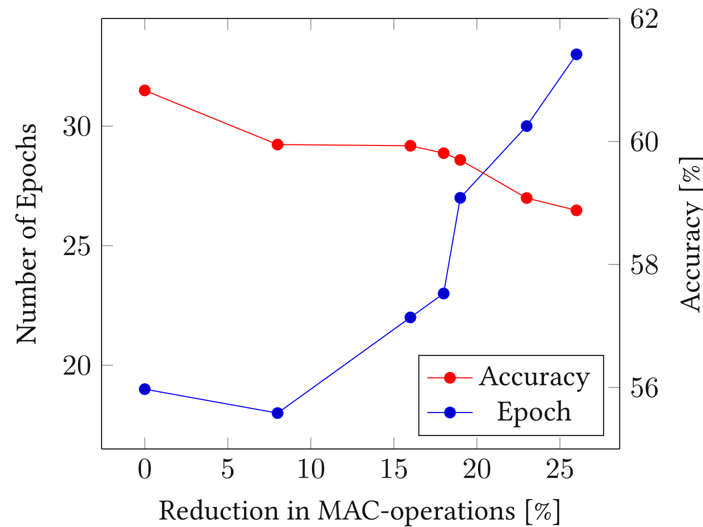


Figure 7.13: Cifar10 Network - Epochs per Operations

7.3.4 Key-Word Performance

Training Conditions

While the Cifar10 and Cifar100 datasets are commonly used in image classification experiments, they are more or less distant from real-life use cases. In contrast, the experiment around the Key-Word scenario uses audio samples for training and evaluation that were recorded by us.

Similar to the Cifar100 experiment, a Plato Learning Rate Schedule was applied here. The learning rate was halved if no improvement in test loss was detected for 50 epochs in a row. The training was executed for 350 epochs.

The Key-Word experiment is the only one applying Dropout Layers. These Dropout Layers significantly improve training performance in this example but lead to non-deterministic training behavior. Therefore, around 150 training runs per investigated training approach were initiated. These training runs were stopped after 20 epochs if no accuracy higher than 30% was achieved by then.

It is important to note that the test dataset for the Key-Word scenario consists of only 160 samples, unlike the 10,000 test samples featured in Cifar10 and Cifar100. Classifying one additional test keyword correctly therefore improves test accuracy by 0.625 percentage points. For clarity, table 7.4 does not provide the test accuracy as a percentage but rather the number of correctly classified test samples.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

schedule	learning rate [E-6]	best acc. [samples]	learning rate range (5 samples) [E-6]	ops per step	rel. ops
4444	766	146	576 – 1899	67.89M	100.00%
4440	2404	130	2186 – 2404	53.15M	78.29%
4444_{lr4442}	1806	142	633 – 1806	67.89M	100.00%
4444_{lr4421}	2734	143	2734 – 2734	67.89M	100.00%
4444_{lr4432}	3200	148	2645 – 3200	67.89M	100.00%
4444_{lr4422}	1122	145	122 – 1806	67.89M	100.00%
4444_{lr4222}	1195	149	1009 – 2985	67.89M	100.00%
4442	1516	145	1516 – 2734	60.52M	89.14%
4421	1520	145	1246 – 2186	48.77M	71.83%
4432	1642	152	1642 – 1642	56.49M	83.20%
4422	1505	149	1031 – 2734	52.45M	77.26%
4422_m	906	144	906 – 2049	53.26M	78.45%
4222	1418	145	1139 – 2680	50.84M	74.89%
4222_m	1322	144	1322 – 1800	50.85M	74.89%
4211_m	908	146	908 – 1449	43.53M	64.12%

Table 7.4: Key Words - Training Performance Summary

Overview

In table 7.4 following can be seen:

- The BPLS-schedule 4432 and its Per-Layer Learning Rate counterpart achieve the highest test accuracy, while 4440 achieves the lowest.
- The network investigated in this experiment does not provide much room for peak-memory optimization but shows the highest potential for reducing peak-operations. While 4422_m, similar to 4440, allows for a negligible reduction in peak-memory (0.7%), 4222_m and 4211_m achieve a reduction of 4.11%. The peak-operations are 78.61% for 4422_m, 95.27% for 4222_m, and 73.88% for 4211_m. The accuracy achieved by these schedules is similar to other training approaches.
- Previous experiments have shown that, typically, the Per-Layer Learning Rate approach allows for the application of higher learning rates than its BPLS counterpart. However, the experiment around the Key-Word scenario does not completely confirm this observation. Here, the Per-Layer Learning Rate approach allows learning rates 25% higher for 4421, 95% higher for 4432, 11% higher for 4222 and 66% higher for 4222_m. Nevertheless, some measurements contradict this observation. For instance, in the case of 4442, the upper bound of the learning rate range of the Per-Layer Learning Rate approach is 34% lower than its BPLS counterpart. In the case of 4422, it is 34% lower, and in the case of 4422_m it is 12% lower. As mentioned, the Key-Word experiment is the only one utilizing Dropout Layers. The resulting nondeterminism could be one possible root cause.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Diagrams

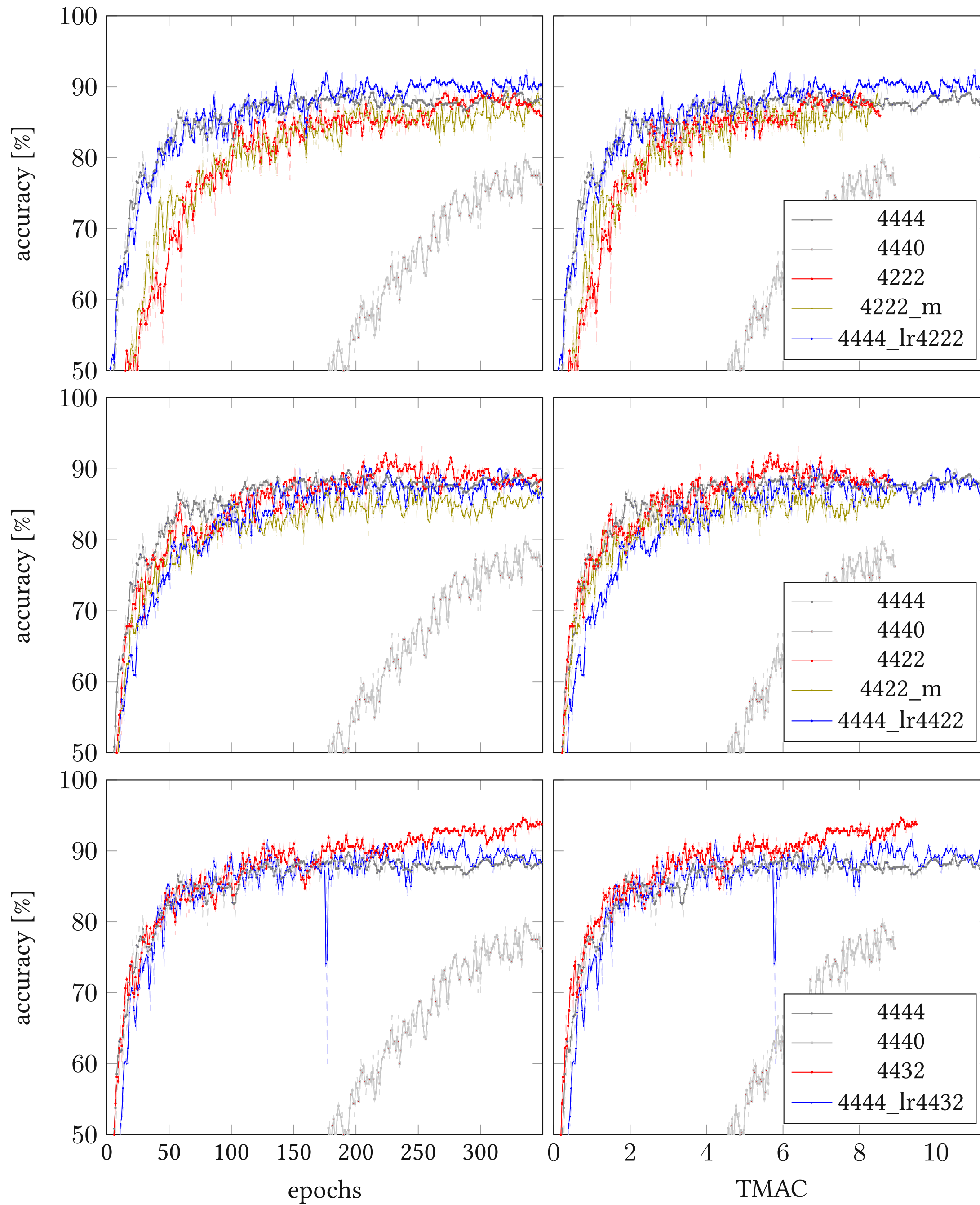


Figure 7.14: KW Network - Training Performance

Figure 7.15 visualizes the training behavior over epochs and number of operations. To make it easier to interpret the plots, a running average filter was applied, slightly flattening the graphs.

As seen in the first row, both 4222 and 4222_m achieve accuracy as high as Conventional Training. However, the BPLS-schedules require significantly more epochs, resulting in slower training, even when considering that both BPLS-schedules perform only 74.89% of the MAC-operations compared to

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

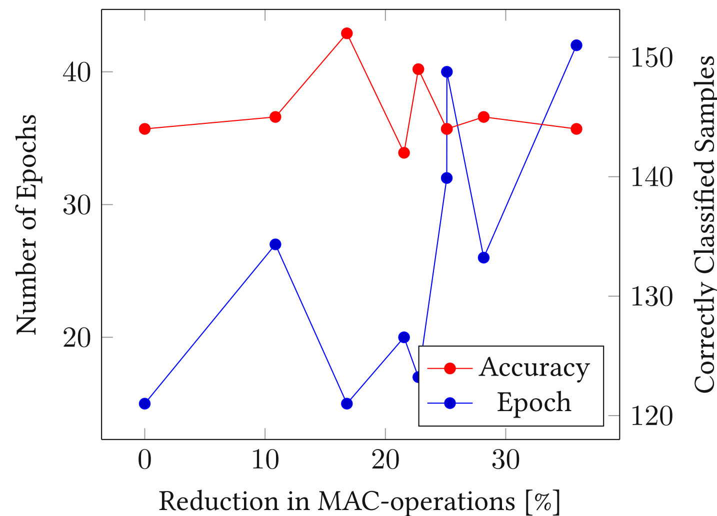


Figure 7.15: KW Network - Epochs per Operations

Conventional Training. In contrast, the Per-Layer Learning Rate counterpart trains as fast as Conventional Training but achieves higher accuracy.

Training just one layer slightly more frequently already reduces the epoch gap between BPLS and the other training approaches, as can be seen in the plots featuring 4422 and 4422_m. In this case, both BPLS-schedules train as fast as Conventional Training considering the number of executed MAC-operations. The accuracy achieved by 4422 is even higher than that achieved by Conventional Training, while the one achieved by 4422_m is slightly lower. The Per-Layer Learning Rate counterpart achieves accuracy as high as Conventional Training but requires more epochs.

Modifying the BPLS-schedule further leads to 4432, featured in the last row. The reduction in the number of MAC-operations per step in the case of 4432 is not as high as in the previously investigated schedules. Nevertheless, 4432 achieves very good results by training faster and achieving higher accuracy compared to other training approaches. In fact, 4432 requires 51.34% fewer operations to reach the best accuracy achieved by Conventional Training. The Per-Layer Learning Rate counterpart of 4432 also exhibits relatively good training behavior. This experiment once more highlights the importance of BPLS configurations being well-fitting. Even slight alterations to the BPLS-schedule can lead to very different training behaviors.

Figure 7.15 displays the number of epochs required to reach 68.0% accuracy in relation to the reduction in operations. Despite the relatively high fluctuation, an increased number of epochs from a specific point on can be detected, similar to Cifar10 and Cifar100. Here, this point is at a reduction of around 22.7%. No decline in accuracy is notable.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

7.3.5 Summary

The Performance section showed that BPLS-schedules can perform as well as conventional and Per-Layer Learning Rate approaches in terms of test accuracy, and possibly even better. In the Key-Word experiment, the BPLS-schedule 4432 achieved an accuracy of 152/160 correctly classified test samples, while the best performing Per-Layer Learning Rate approach achieved only 149/160 and Conventional Training 146/160.

One key idea of BPLS is to reduce the number of executed operations, thereby reducing training time and energy consumption. The number of operations must be considered in combination with the number of epochs required for training. Specifically, in the Key-Word experiment, BPLS-schedule 4222 demonstrated that despite executing only 74.89% of the operations of Conventional Training, BPLS required more operations to complete training because it needed significantly more epochs than the other investigated approaches.

However, as seen with the 4432 BPLS-schedule in the Key-Word experiment, when a well-fitting schedule is applied, no additional epochs need to be carried out. In this case, 4432 required 51.34% fewer operations, and therefore less time and energy to reach the best accuracy achieved by Conventional Training. Overall, the measurements revealed, that an increased number of skipped operations does not affect achieved accuracy significantly but rather increases the number of training epochs.

In addition to reducing execution time and energy consumption, a fundamental idea behind BPLS is to limit the number of operations per training step. This can be useful for meeting real-time criteria or synchronizing with processes. Even though 4211_m in the Key-Word experiment required more time and energy to complete training, it reduced the average number of operations per training step to 64.12% and peak-operations to 73.88%, while achieving accuracy as high as Conventional Training.

Furthermore, it was shown that peak-memory-optimizing BPLS-schedules can significantly reduce peak-memory while still maintaining sufficient training speed and accuracy. This is especially evident in the Cifar100 experiment, where BPLS-schedule 444222_m reduced peak-memory by 22.86% while achieving an accuracy of 59.7%. 663333_m even reduced peak-memory by 49.61% while achieving 58.88% test accuracy. In comparison, 444440 reduced peak-memory by only 7.78% while achieving 58.61% accuracy.

In both scenarios, Cifar10 and Key-Word, BPLS achieved the highest accuracy. This could indicate that BPLS has a positive impact on accuracy, possibly due to a generalizing effect. However, these accuracies vary only slightly from other training approaches and could therefore be purely coincidental. Further work is required to validate this aspect.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Through the experiments, the importance of well-fitting BPLS-schedules for achieving sufficient training performance became evident. While the amount of operations per training step and peak-memory depend only on the network structure, the ideal schedule configuration, achieved accuracy and required number of epochs also depend on the fine-tuning data.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

7.4 Evaluation of Estimations

The previous section (Performance) compared training approaches in terms of test accuracy, the number of executed operations, and peak memory requirements. However, instead of measuring the actual execution time for each specific training approach on a specific platform, we estimated the number of operations using an optimal mathematical model. Accordingly, it is crucial that the mathematical model accurately represents real-world behavior.

This section provides actual execution time measurements and compares them with the estimated number of operations. The measurements were conducted on three devices: Server, Desktop, and MCU. For evaluating peak memory estimations, stack information automatically generated during compilation was used.

Execution time and the number of operations executed cannot be compared directly. Instead, we compare the ratio between the training approaches of interest and Conventional Training. Conventional Training executes the most operations during training and therefore requires the highest amount of time. Additionally, we compare the estimated and measured ratio between backward and forward propagation for Conventional Training.

7.4.1 Server Measurements

Initially, it was planned to measure execution time only on the featured Server CPU using PyTorch. However, modern CPUs are very complex. Advanced processor functionalities such as out-of-order execution, instruction-level parallelism, and multiple layers of cache make predictions quite difficult. Additionally, modern operating systems and machine learning frameworks like PyTorch add an additional layer of complexity.

Due to this increased complexity, the mathematical estimations and measured data differ quite drastically, as can be seen in Figure 7.16 based on the Cifar10 scenario. The estimations are represented by simple bar diagrams, while for the Server measurements, the first quartile (q1), median, and third quartile (q3) are plotted. These are defined as follows:

- **Median**: median(training-approach) / median(conventional-training)
- **q1**: q1(training-approach) / q3(conventional-training)
- **q3**: q3(training-approach) / q1(conventional-training)

It should be noted that these values represent training time, respectively the number of operations executed during training, which means that backward and forward propagation are accumulated. As seen in Figure 7.16, especially for the ratio between 1100 and 1111, the difference between estimation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

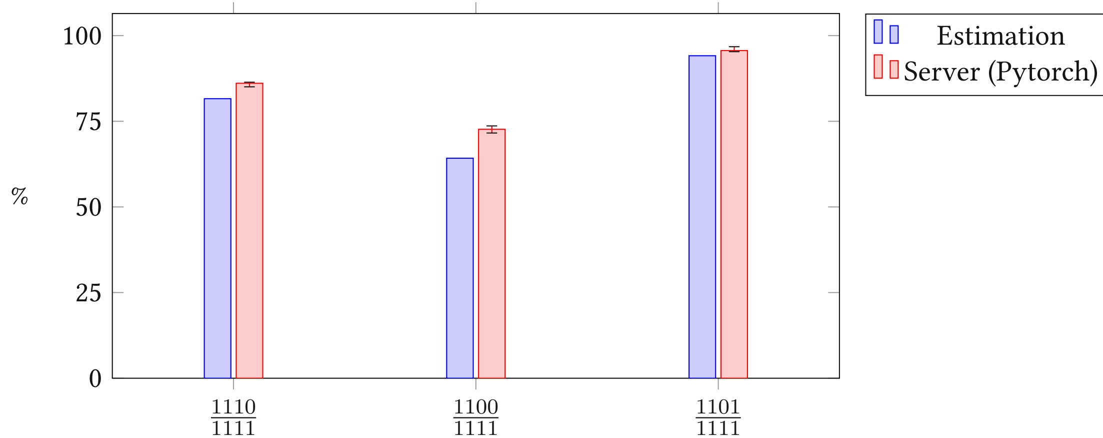


Figure 7.16: Comparison between Server measurement and estimated number of operations (bw + fw)

and measurement is quite high, while the difference for the ratio featuring 1101 is surprisingly low.

Furthermore, as shown in Table 7.5, the estimated backward-to-forward ratio fits the measured one almost perfectly. According to the measurements, the backward path requires 91% of the forward time, while according to the estimations, during backward propagation, 93% of forward path operations are executed. How can it be that the backward path requires less time, respectively executes fewer operations? Simply because forward propagation goes through all layers, while backward propagation only treats unfrozen ones.

	estimation	measurement
bw / fw	0.93	0.91

Table 7.5: Server measurement: bw / fw

To learn more about the Server measurements, Figure 7.17 shows a histogram featuring all measurement samples. In this plot, the backward and forward propagation time is not accumulated, but instead, the raw backward time is shown. While the samples spread over a wide area, especially in the case of 1100, two peaks can clearly be seen. It can be assumed that the left peak represents the ideal case, where data can be fetched from cache.

7.4.2 MCU Measurements

MCUs are much simpler in structure than modern Server CPUs, which makes their behavior much easier to predict. Additionally, we did not use an operating system on the MCU but let it focus completely on executing our training code. Instead of using nontransparent frameworks, we wrote the code from the ground up. This allowed us to modify the mathematical model according to the actual implementation. After minor adjustments, the estimated number of operations aligned very well with the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

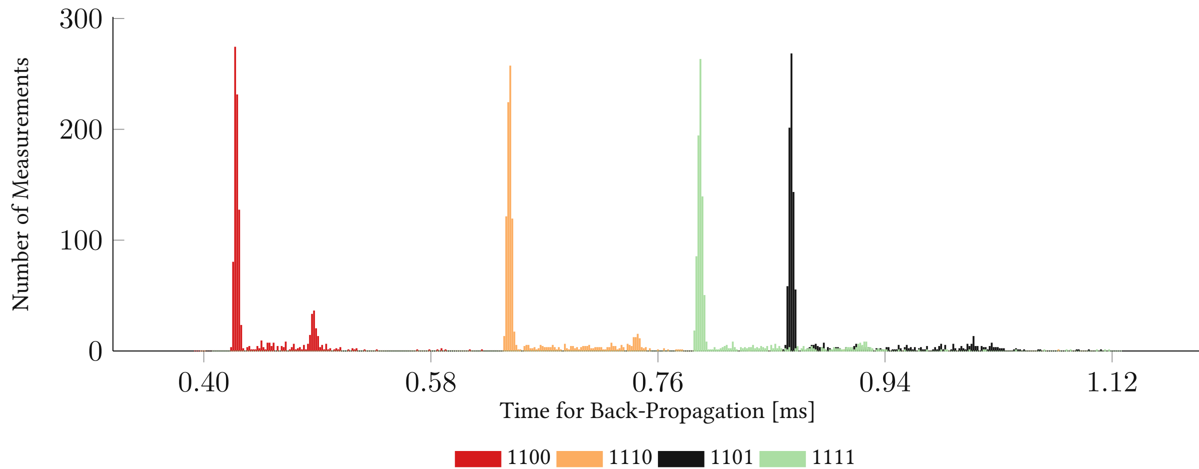


Figure 7.17: Server measurements - Histogram

measurement results. This section describes the applied adjustments and compares the measurements to the estimations.

Due to the limited capacities of the MCU, the experiment is based on the MCU scenario introduced in the Networks and Datasets chapter instead of the Cifar10 scenario used for the Server measurements. The adjusted mathematical model can be found in Table 7.6. The formulas are equivalent to those listed in Tables 2.1 and 2.2 of the Background chapter. However, Table 7.6 includes scaling and the introduction of an offset.

The following figure and table compare the estimated and measured values, similar to previously done with the measurements executed on the Server CPU. The following estimations and measurements are featured:

- **raw-estimation**: represents estimations derived from the pure mathematical model that was used for performance evaluation.
- **adj. estimation**: basically the same model as used for the raw-estimations with adjustments featured in Table 7.6.
- **mcu-Ofast**: measurements performed on MCU. The code was optimized for speed.
- **mcu-Omin**: measurements performed on MCU. The code was optimized to occupy minimal space on flash.
- **Desktop**: MCU code was compiled and executed on a desktop computer.

As seen in Table 7.7, regarding the backward-to-forward ratio, the raw estimations match the measurements performed on the Desktop approximately as well as the Server measurements. The measurements of MCU-Ofast differ significantly. The adjusted estimations match MCU-Omin perfectly. However, it should be kept in mind that the mathematical model was actively adjusted to match the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Layer Type	Number of Ops
bw-Conv-Input-Grad.	$c_{in} \cdot c_{out} \cdot (k_{size} \cdot (i_{size} - 2 \cdot pad) + 2 \cdot (pad \cdot k_{size} \cdot pad^2 + \frac{pad^2 - pad}{2})) \cdot \mathbf{1.3}$
bw-Conv-Weight-Grad.	$c_{in} \cdot c_{out} \cdot (1 + 2 \cdot (pad \cdot o_{size} \cdot pad^2 + \frac{pad^2 - pad}{2})) \cdot \mathbf{1.3}$
bw-Lin-Input-Grad.	$c_{in} \cdot c_{out} \cdot \mathbf{1.3}$
bw-Lin-Weight-Grad.	$c_{in} \cdot c_{out} \cdot \mathbf{1.3}$
fw-ReLU	$c_{in} \cdot i_{width} \cdot i_{height} \cdot \mathbf{1.5}$
Offset	375
bw-Conv-Update	$c_{in} \cdot k_{size} \cdot c_{out} + c_{out}$
bw-Lin-Update	$c_{in} \cdot c_{out} + c_{out}$
bw-ReLU	0
bw-Max-Pool.	$c_{in} \cdot i_{size}$
bw-Gl-Avrg-Pool.	$c_{in} \cdot i_{size}$
fw-Conv	$c_{in} \cdot c_{out} \cdot (k_{size} \cdot (i_{size} - 2 \cdot pad) + 2 \cdot (pad \cdot k_{size} \cdot pad^2 + \frac{pad^2 - pad}{2})) + c_{out} \cdot o_{size}$
fw-Lin	$c_{in} \cdot c_{out} + c_{out}$
fw-Max Pool	$c_{in} \cdot i_{size}$
fw-Gl-Avrg-Pool.	$c_{in} \cdot i_{size}$

Table 7.6: Number of Operations required for different layers

corresponding implementation.

	raw-estimation	adj. estimation	mcu-Omin	mcu-Ofast	desktop (cpp)
bw / fw	2.98	3.44	3.44	2.24	3.03

Table 7.7: MCU measurement: bw / fw

Figure 7.18 shows that especially the adjusted estimations, but also the raw estimations, align quite well with MCU-Omin. However, the measurements performed on the Desktop, and especially the MCU-Ofast measurements, differ more drastically. The reason why MCU-Omin fits so well is likely due to optimizing the code to require the least amount of space, with almost no timing optimization, such as aggressive loop unrolling, just as assumed by the estimations.

Additionally, as can be seen, not only the estimations but also the MCU measurements show only one value per bar, due to zero variance in execution time. On the other hand, for training performed on the Desktop, q1, median, and q3 are provided, just as for the Server CPU measurements earlier.

7.4.3 Evaluation of theoretical Memory Model

The estimation of memory requirements is a complex topic that is just briefly touched upon in this thesis. While compiling the CPP code for MCU and Desktop, a static memory allocation output was automatically generated. According to the simple mathematical memory estimations provided in the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

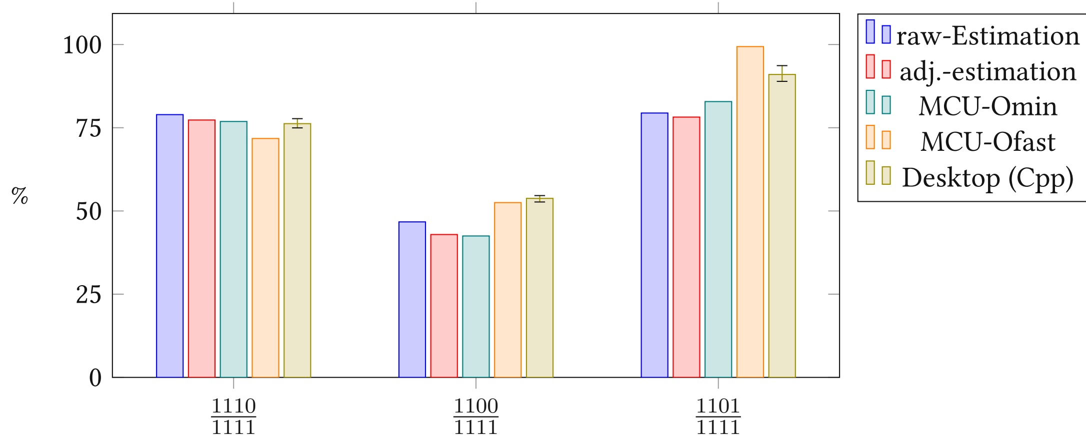


Figure 7.18: Comparison between MCU measurements and estimated number of operations

Background chapter, it is expected that skipping certain layers in the backward path reduces the amount of data required to store. Figure 7.19 compares the estimations with the static memory allocation output for Desktop and MCU. The MCU memory allocation is provided in two variants: with and without Memory Optimization.

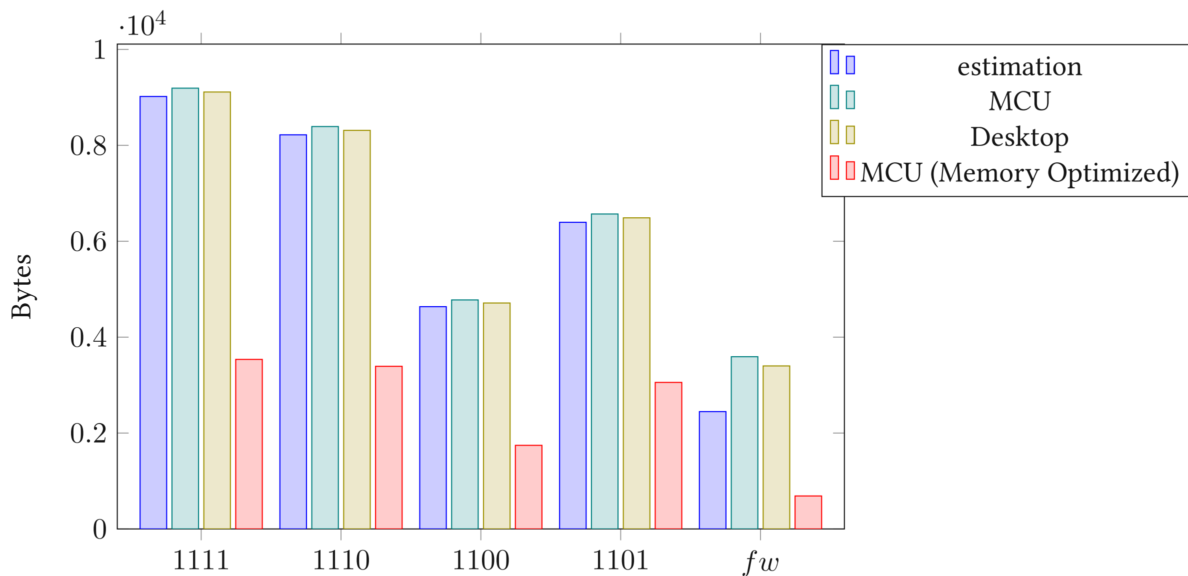


Figure 7.19: Estimated peak memory requirements and static stack information

It should be noted that the estimations ignore the potential for memory optimization, just as the allocation outputs. Only the Memory Optimized allocation output releases memory that is not required anymore to reuse it for new Gradients or Feature Maps.

As shown in Figure 7.19, the mathematical estimations are not perfect but align well with the allocation outputs without Memory Optimization. Training the layers 1110 and 1101 alternately, as BPLS-schedule 4422_m does, reduces peak memory by 8.87% compared to training each unfrozen layer (1111).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Applying a schedule that trains the layers 1010 and 1101 alternately even leads to a peak memory reduction of 26.26% in this specific example. A corresponding BPLS-schedule would be 4222_m.

The Memory Optimized allocation output differs significantly from the one without Memory Optimization. While BPLS can reduce peak memory in both cases, its benefit is slightly lower when Memory Optimization is applied, as shown in Table 7.8.

	1110 / 1111	1101 / 1111	1100 / 1111	fw / 1111
Memory Opt.	95.93%	86.43%	49.32%	19.46%
not Memory Opt.	91.30%	71.45%	48.04%	60.92%

Table 7.8: Peak Memory Reduction - with and without Memory Optimization

7.4.4 Summary

This section has demonstrated that the estimations for both executed operations and peak memory requirements match the measurements quite well. Consequently, these estimations can be used for performance evaluation with confidence.

In more detail, while the timing measured on the Server and Desktop differs, especially the Omin-optimized MCU code aligns well with the estimations. Furthermore, without Memory Optimization, the stack allocation outputs closely match the estimated peak memory. Considering Memory Optimization, a slightly reduced, but still notable optimization potential can be seen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 8

Conclusion and Future Work

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

8.1 Summary

This thesis introduces the so-called Back Propagation Layer Scheduling (BPLS) approach. The key idea behind BPLS is to train neural networks following training schedules. These schedules specify at which training step which layer is trained.

By skipping less critical training steps, the number of executed operations can be reduced. In fine-tuning neural networks, the classifier layer often requires more adaptation than earlier layers. The well-known Per-Layer Learning Rate approach leverages this by applying higher learning rates to later layers while keeping the learning rates low for earlier layers. Consequently, skipping training steps of earlier layers might reduce training time and energy consumption without quality loss. Additionally, defining schedules with a limited number of operations could be used to synchronize training with other processes, meet real-time constraints, or adhere to energy budgets. By configuring training schedules in specific ways, peak-memory requirements can also be minimized.

This thesis primarily focuses on addressing two questions:

1. How does BPLS behave in regard of the Per-Layer Learning Rate approach.
2. Can BPLS improve the efficiency of fine-tuning without affecting training behavior negatively.

To assess the similarity of BPLS with other training approaches and to investigate its performance, three fine-tuning scenarios were defined based on the Cifar10, Cifar100, and a custom-created keyword dataset. More details about these scenarios can be found in the Networks and Datasets chapter.

8.2 Conclusion

8.2.1 Similarity Evaluation

The similarity evaluation revealed a clear correlation between Per-Layer Learning Rate, BPLS, and Random Dropout counterparts. Additionally, a second correlation was noticed between Up-Scaled training and Conventional Training. The investigated training approaches are detailed in the BPLS chapter and briefly summarized here:

- **Per-Layer Learning Rate**: Assigns a specific learning rate to each layer.
- **BPLS**: Skips training steps of certain layers following a schedule.
- **Random Dropout**: Randomly drops gradients with each layer having a specific dropout rate assigned.
- **Up-Scaled**: BPLS, but layer-specific learning rates compensate skipping.

In general, it appears that two training approaches sharing the same configuration at a specific layer tend to exhibit similar patterns of similarity at that layer. However, during the investigations, it became clear that layers influence each other. Even when most BPLS-schedules do not modify the training of the last few layers of a network, these layers continue the similarity patterns observed in the preceding layers.

This influence is especially evident in the plots generated by UMAP, a highly complex Dimensionality Reduction Tool used to represent n-dimensional weights of a layer as 2D point. Throughout the layers, UMAP generates clusters that include BPLS-schedules and their counterparts on one hand, and Up-Scaled variants and Conventional Training on the other. The clustering was mostly as expected.

While Random Dropout is always most similar to its Per-Layer Learning Rate and BPLS counterparts, the similarity is actually relatively low at layers where gradients are dropped, contradicting expectations. One reason could be that due to random dropout, some weights never access certain training samples. In contrast, with Conventional Training and Per-Layer Learning Rate, every layer (and thus every weight) accesses every training sample in each epoch. BPLS uses Dataset Shifting to achieve similar behavior.

At layers where no gradients are dropped, Per-Layer Learning Rate is even more similar to Random Dropout than to BPLS. At these layers, the configurations do not differ from Conventional Training. This suggests that dropping gradients from earlier layers must have a similar effect as adjusting their learning rates accordingly.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The Up-Scaled training approaches showed the highest similarity to Conventional Training, as expected. However, the actual similarity to Conventional Training is relatively low. This could be caused by excessively high learning rates. Especially at layers where only every 4th step is trained, the learning rate is four times the base learning rate. Excessive learning rates can have negative effects on training, which is reflected in reduced accuracy.

8.2.2 Performance Evaluation

The Performance Evaluation compared BPLS to the best-performing Conventional Training and the corresponding Per-Layer Learning Rate approach. The optimal learning rates were determined using Grid Search. The number of operations and peak-memory requirements were obtained using an analytical model. To ensure correctness, one network was validated against hardware using Server, Desktop, and MCU platforms. Several metrics were investigated:

- **Test-Accuracy**: The highest achieved test accuracy.
- **Training Time and Energy**: Reduction in training time and energy consumption.
- **Number of MAC-Operations**: MAC-operations executed by training step.
- **Peak-Memory**: Peak-memory requirements during training.

It was found that BPLS can perform as well as the Conventional and Per-Layer Learning Rate approaches in terms of test accuracy by reducing the number of executed MAC-operations and peak-memory requirements. It has been shown that applying a well-fitting schedule is crucial. Skipping too many training steps can result in an increased number of training epochs required to achieve sufficient accuracy, negating some of the benefits of BPLS. The trade-off can be best explained by investigating the individual metrics.

Test-Accuracy: In the Key-Word experiment, training with BPLS led to 152/160 correctly classified test samples, while reducing the number of operations by 16.8%. The best-performing Per-Layer Learning Rate approach achieved 149/160, and Conventional Training 146/160 correctly classified test-samples. This could indicate that BPLS has an unintended regularizing effect. However, further work is needed to validate this aspect.

At the Cifar10 scenario, BPLS achieved 78.25% accuracy with a 9.22% reduction in operations. Here, Per-Layer Learning Rate achieved 78.13%, and Conventional Training 78.17%. At the Cifar100 scenario, BPLS achieved 60.06% accuracy with a 16.39% reduction in operations, while Per-Layer Learning Rate achieved 60.37% and Conventional Training 60.83%.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Training Time and Energy: Considering the number of training epochs, at the Key-Word experiment BPLS completed training requiring 51.34% less time and energy. At Cifar10, the reduction was 17.91%, and at Cifar100, 7.53%. The accuracies for these runs were 152/160 at the Key-Word experiment, 78.15% at Cifar10, and 59.95% at Cifar100.

When skipping more training steps than a scenario-specific threshold, significantly more epochs and, consequently, more time and energy are required to complete training, with little to no impact on accuracy. These additional epochs may be necessary to achieve the required number of training steps. For CIFAR10, this threshold is at a reduction in operations of 23%, for CIFAR100 at a reduction of 16%, and for the Key-Word experiment at 25%.

Number of MAC-Operations: Particularly in Real-Time-Systems, meeting deadlines is crucial. For example, a deadline could represent the time when new training data arrives. Until then, the previous training data must be processed. Moreover, the time available for individual training steps could be limited to synchronize processes, making the average and maximum number of operations per training step relevant.

At the Key-Word experiment, the lowest maximum operations per training step were 26.12% lower than conventional training, with 35.88% fewer operations on average, while maintaining the same test accuracy as Conventional Training. For Cifar10, a reduction of 17.62% in maximum operations and 18.03% on average was achieved with no loss in accuracy. For Cifar100, reductions of 16.31% in maximum operations and 23.13% on average were achieved with 59.08% accuracy. In all these cases, BPLS resulted in an increased number of training epochs.

Peak-Memory: Peak-memory optimizing BPLS-schedules at the Cifar100 experiment reduced peak-memory by 22.86% while achieving 59.7% accuracy. Schedules that allowed more compromises in terms of training speed and accuracy were capable of achieving a 49.61% reduction in peak-memory with 58.88% accuracy. The potential for peak-memory reduction depends highly on the specific network structure. In the Key-Word scenario, reductions of 4.11%, and at Cifar10, reductions of 14.01% were achieved with practically no loss in accuracy.

8.3 Future Work

In this thesis BPLS was introduced and evaluated in its pure form. Subsequent some of the next promising aspects around BPLS are listed and briefly discussed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

8.3.1 Automatic Schedule Definition

During the experiments, it became clear how important well-defined BPLS-schedules are for achieving effective training. Skipping training steps too frequently results in an increased number of epochs required to complete training, negating some of the benefits of BPLS. Conversely, performing unnecessary training steps wastes optimization potential. Therefore, techniques for finding well-performing schedules for individual training scenarios would be very useful.

One of the achievements of this thesis was to show that training with BPLS behaves similarly to training with the corresponding Per-Layer Learning Rate configuration. As described in the Related Work chapter, methods such as AutoLR [97] allow for the automatic definition of well-performing Per-Layer Learning Rate configurations. An investigation could begin by converting such configurations into BPLS schedules, as described in the BPLS chapter.

Furthermore, the Related Work chapter introduced approaches for dynamically adapting the number of frozen layers during fine-tuning. AutoFreeze [101] is one of these approaches. Similar techniques could be applied to dynamically adapt BPLS schedules during training.

8.3.2 BPLS combined with on-device optimization techniques

BPLS was designed to improve the efficiency of training neural networks on resource-constrained devices. In theory, similar to the Per-Layer Learning Rate approach, BPLS is compatible with most common optimization techniques for training on such devices.

The experiments in this thesis utilized full-precision floating-point values. However, on resource-constrained devices, it is more common to use lower precision or even integer values, a process referred to as quantization. Additionally, networks are usually pruned before being applied to such devices, which involves removing network parameters that contribute little to the network output. More about these techniques can be found in the Background chapter.

It would be interesting to see how BPLS performs when training quantized and pruned networks. While quantization can decrease the achieved network quality, no additional negative effects on BPLS are assumed. On the other hand, skipping training steps could have different effects on pruned networks compared to unpruned ones. While un-pruned networks can have neglectable weights, pruned networks are reduced to the essential ones. When fine-tuning pruned networks, the Per-Layer Learning Rate approach is commonly applied, raising expectations that BPLS can achieve good results in this case as well due to the similarity between these training approaches.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Quantizing floating-point parameters to integer precision leads to operations requiring less execution time and energy. Pruning, on the other hand, reduces the number of parameters in certain layers and therefore the number of operations. Additionally, both techniques can reduce the amount of memory required.

In addition to quantization and pruning, more advanced optimization approaches for efficient training on resource-constrained devices have been recently introduced. Examples include MiniLearn [14] and POET [16]. MiniLearn stores weights and intermediate outputs in integer precision and dequantizes them to floating-point during training. POET considers hardware constraints by searching for the optimal schedule of paging and rematerialization. More about these approaches and other concepts can be found in the Related Work chapter. Investigating the interaction between BPLS and some of these approaches could be a further step.

8.3.3 BPLS in real life experiments

Besides investigating how BPLS behaves in combination with the mentioned optimization techniques, it would be interesting to see how BPLS performs in real-life situations. This would involve its execution on actual MCUs.

This thesis introduced three scenarios. One of them, the Key-Word scenario, is based on real-life data. It features a Robot-Dog pre-trained to understand English keywords. During the fine-tuning experiment, the Robot-Dog learned to understand those keywords in German and Japanese. The corresponding dataset and network can be found in the Networks and Datasets chapter.

It would be interesting to actually build this Robot-Dog and equip it with an MCU running a quantized and pruned variant of the investigated network. This Robot-Dog could then be used for field experiments comparing BPLS to other training approaches.

8.3.4 BPLS on large Neural Networks

At its core, BPLS represents an approach for fine-tuning neural networks fast and efficiently. Consequently, BPLS is platform-independent. This thesis focused on devices relying on resource constraints, especially MCUs. However, energy, timing, and memory optimizations are important for training larger networks, for example at data centers, as well. Such optimizations can help speed up training or allow even larger networks to be trained by reducing memory requirements.

During the final stages of this thesis, the LISA [100] approach was published (as white paper), which follows a relatively similar idea to BPLS but focuses on fine-tuning Large Language Models. LISA

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

randomly freezes layers according to their probability, while the first and last layers of the network remain permanently unfrozen. This allows to speed up training and reduce GPU memory consumption. More about LISA can be found in the Related Work chapter.

In some ways, BPLS can be seen as a more advanced form of LISA. Consequently, it would be interesting to investigate the performance of BPLS on such large networks and directly compare it with LISA. Use cases could include Large Language Models or intensive image classification tasks. For example, the GSM8K dataset (8.5 thousand math word problems) [112] could be applied to the BERT-LARGE network (340 million network parameters) [113], or ImageNet (14 million color images, 469x387 average resolution) [114] could be trained on the VGG19 network (144 million network parameters) [78].

We assume that BPLS will achieve similar performance as LISA, in regard of accuracy and training time. However, because LISA freezes layers randomly, it leads to probabilistic behavior. BPLS utilizes pre-defined schedules. The resulting deterministic execution time and peak-memory requirements can represent a significant advantage.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[1] (2023-10-25) Vgg16 architecture diagram. https://neurohive.io/en/popular-networks/vgg16/.

[2] (2023-10-25) Vgg16 architecture diagram. https://medium.com/machine-learning-for-li/different-convolutional-layers-43dc146f4d0e.

[3] (2023-07-27) Max pool illustration. https://paperswithcode.com/method/max-pooling.

[4] (2023-08-01) Optimizers in machine learning - ema. https://medium.com/nerd-for-tech/optimizers-in-machine-learning-f1a9c549f8b4.

[5] J. Konar, P. Khandelwal, and R. Tripathi, "Comparison of various learning rate scheduling techniques on convolutional neural network," in *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 2020, pp. 1–5.

[6] L. N. Smith, "Cyclical learning rates for training neural networks," 2017.

[7] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf

[8] A. A. Ajani TS, Imoize AL, "An overview of machine learning within embedded and mobile devices – optimizations and applications. sensors," 2021.

[9] D. Liu, W. Cui, K. Jin, Y. Guo, and H. Qu, "Deeptracker: Visualizing the training process of convolutional neural networks," 2018.

[10] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The ucr time

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

series classification archive," October 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[11] S. Wu, H. Fei, L. Qu, W. Ji, and T.-S. Chua, "Next-gpt: Any-to-any multimodal llm," 2023.

[12] T. M. Hoang, S. H. Nam, and K. R. Park, "Enhanced detection and recognition of road markings based on adaptive region of interest and deep learning," *IEEE Access*, vol. 7, pp. 109 817–109 832, 2019.

[13] F. Thabtah and D. Peebles, "A new machine learning model based on induction of rules for autism detection," *Health Informatics Journal*, vol. 26, no. 1, pp. 264–286, 2020, pMID: 30693818. [Online]. Available: https://doi.org/10.1177/1460458218824711

[14] M. A. Christos Profentzas and O. Landsiedel, "Minilearn: On-device learning for low-power iot devices," in *In Proceedings of the 2022 International Conference on Embedded Wireless Systems and Networks (Linz, Austria) (EWSN' 22)*, 2022.

[15] H. Ren, D. Anicic, and T. A. Runkler, "Tinyol: Tinyml with online-learning on microcontrollers," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[16] S. G. Patil, P. Jain, P. Dutta, I. Stoica, and J. Gonzalez, "POET: Training neural networks on tiny devices with integrated rematerialization and paging," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 17 573–17 583. [Online]. Available: https://proceedings.mlr.press/v162/patil22b.html

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[18] OpenAI, :, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph, "Gpt-4 technical report," 2023.

[19] J. Oppenlaender, "The creativity of text-to-image generation," in *Proceedings of the 25th International Academic Mindtrek Conference*. ACM, nov 2022. [Online]. Available: https://doi.org/10.1145%2F3569219.3569352

[20] J. Betker, G. Goh, L. Jing, TimBrooks, J. Wang, L. Li, LongOuyang, JuntangZhuang, JoyceLee, YufeiGuo, WesamManassra, PrafullaDhariwal, CaseyChu, YunxinJiao, and A. Ramesh, "Improving image generation with better captions." [Online]. Available: https://api.semanticscholar.org/CorpusID:264403242

[21] K. Arulkumaran, A. Cully, and J. Togelius, "Alphastar: an evolutionary computation perspective," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 314 – 315. [Online].

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Available: https://doi.org/10.1145/3319619.3321894

[22] A. Agnesina, P. Rajvanshi, T. Yang, G. Pradipta, A. Jiao, B. Keller, B. Khailany, and H. Ren, "Autodmp: Automated dreamplace-based macro placement," in *Proceedings of the 2023 International Symposium on Physical Design*, ser. ISPD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 149 – 157. [Online]. Available: https://doi.org/10.1145/3569052.3578923

[23] S. Disabato and M. Roveri, "Incremental on-device tiny machine learning," in *Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, ser. AIChallengeIoT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 7 – 13. [Online]. Available: https://doi.org/10.1145/3417313.3429378

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[27] V. R. Joseph, "Optimal ratio for data splitting," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 15, no. 4, p. 531 – 538, Apr. 2022. [Online]. Available: http://dx.doi.org/10.1002/sam.11583

[28] (2023-07-25) The cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[30] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

[31] (2023-07-25) Pycharms augmentation api. https://pytorch.org/vision/main/transforms.html.

[32] S. Mallat, "Understanding deep convolutional networks," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150203, 2016. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2015.0203

[33] L. Lu, "Dying relu and initialization: Theory and numerical examples," *Communications in Computational Physics*, vol. 28, no. 5, p. 1671 – 1706, Jun. 2020. [Online]. Available: http://dx.doi.org/10.4208/cicp.OA-2020-0165

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[34] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[35] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013.

[36] R. Girshick, "Fast r-cnn," 2015.

[37] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 2169–2178.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[39] W. Jung, D. Jung, , B. Kim, S. Lee, W. Rhee, and J. H. Ahn, "Restructuring batch normalization to accelerate cnn training," 2019.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929 – 1958, jan 2014.

[41] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf

[42] J. Hron, A. G. de G. Matthews, and Z. Ghahramani, "Variational gaussian dropout is not bayesian," 2017.

[43] S. Mazilu and J. Iria, "L1 vs. l2 regularization in text classification when learning from labeled features," 12 2011.

[44] (2023-07-30) Google about on-device machine learning. https://developers.google.com/learn/topics/on-device-ml.

[45] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," ser. Psychology of Learning and Motivation, G. H. Bower, Ed. Academic Press, 1989, vol. 24, pp. 109–165. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0079742108605368

[46] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?" 2020.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[47] P. Swarup, R. Maas, S. Garimella, S. H. Mallidi, and B. Hoffmeister, “Improving ASR Confidence Scores for Alexa Using Acoustic and Hypothesis Embeddings,” in *Proc. Interspeech 2019*, 2019, pp. 2175–2179.

[48] (2023-07-30) Transmittion time to pluto. https://spacemath.gsfc.nasa.gov/.

[49] V. Růžička, A. Vaughan, D. De Martini, J. Fulton, V. Salvatelli, C. Bridges, G. Mateo-Garcia, and V. Zantedeschi, “Ravæn: unsupervised change detection of extreme events using ml on-board satellites,” *Scientific Reports*, Oct 2022.

[50] A. Ibrahim and M. Valle, “Real-time embedded machine learning for tensorial tactile data processing,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 11, pp. 3897–3906, 2018.

[51] h.-c. Wang, I. Woungang, C.-W. Yao, A. Anpalagan, and M. Obaidat, “Energy-efficient tasks scheduling algorithm for real-time multiprocessor embedded systems,” *The Journal of Supercomputing*, vol. 62, 11 2012.

[52] A. Shrestha and A. Mahmood, “Review of deep learning algorithms and architectures,” *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019.

[53] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, “Ai and ml accelerator survey and trends,” in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1109/HPEC55821.2022.9926331

[54] D. Steinkraus, I. Buck, and P. Simard, “Using gpus for machine learning algorithms,” in *Eighth International Conference on Document Analysis and Recognition (ICDAR’05)*, 2005, pp. 1115–1120 Vol. 2.

[55] (2023-08-06) Atmel. atmel—atmega48p/88p/168p/328p datasheet. https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf.

[56] (2023-08-06) Stmicroelectronics. stm32f215xx stm32f217xx datasheet. https://www.st.com/resource/en/datasheet/stm32f215re.pdf.

[57] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size,” 2016.

[58] (2023-08-06) Keras, a. keras api reference/keras applications. https://keras.io/api/applications/.

[59] M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, “Comparing energy efficiency of cpu, gpu and fpga implementations for vision kernels,” 2019.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[60] C. Bobda, J. M. Mbongue, P. Chow, M. Ewais, N. Tarafdar, J. C. Vega, K. Eguro, D. Koch, S. Handagala, M. Leeser, M. Herbordt, H. Shahzad, P. Hofste, B. Ringlein, J. Szefer, A. Sanaullah, and R. Tessier, "The future of fpga acceleration in datacenters and the cloud," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, no. 3, feb 2022. [Online]. Available: https://doi.org/10.1145/3506713

[61] M. Elnawawy, A. Farhan, A. Al Nabulsi, A. Al-Ali, and A. Sagahyroon, "Role of fpga in internet of things applications," 12 2019, pp. 1–6.

[62] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[63] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware: A review," *IEEE Sensors Journal*, vol. 22, no. 22, p. 21362 – 21390, Nov. 2022. [Online]. Available: http://dx.doi.org/10.1109/JSEN.2022.3210773

[64] A. Ushiroyama, M. Watanabe, N. Watanabe, and A. Nagoya, "Convolutional neural network implementations using vitis ai," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 2022, pp. 0365–0371.

[65] S. Takamaeda-Yamazaki. (2023-11-20) nngen - a fully-customizable hardware synthesis compiler for deep neural network. https://github.com/NNgen/nngen/tree/develop.

[66] D. Mahajan, J. Park, E. Amaro, H. Sharma, A. Yazdanbakhsh, J. K. Kim, and H. Esmaeilzadeh, "Tabla: A unified template-based framework for accelerating statistical machine learning," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016, pp. 14–26.

[67] R. Machupalli, M. Hossain, and M. Mandal, "Review of asic accelerators for deep neural network," *Microprocessors and Microsystems*, vol. 89, p. 104441, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933122000163

[68] (2023-08-06) The ai race expands: Qualcomm reveals "cloud ai 100" family of datacenter ai inference accelerators for 2020. https://www.anandtech.com/show/14187/qualcomm-reveals-cloud-ai-100-family-of-datacenter-ai-inference-accelerators-for-2020.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[69] (2023-08-06) Facebook joins amazon and google in ai chip race. https://www.ft.com/content/1c2aab18-3337-11e9-bd3a-8b2a211d90d5.

[70] (2023-08-06) Samsung and amd will reportedly take on apple's m1 soc later this year. https://arstechnica.com/gadgets/2021/05/report-the-samsung-amd-exynos-soc-will-be-out-for-laptops-this-year/.

[71] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[72] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016.

[73] I. C. Hubara, M. Soudry, D. El-Yaniv, R. Bengio, and Y., "Quantized neural networks: Training neural networks with low precision weights and activations." 2018.

[74] W. Roth, G. Schindler, B. Klein, R. Peharz, S. Tschiatschek, H. Fröning, F. Pernkopf, and Z. Ghahramani, "Resource-efficient neural networks for embedded systems," 2022.

[75] (2023-08-09) Estimating an optimal learning rate for a deep neural network. https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0.

[76] M. M. Pasandi, M. Hajabdollahi, N. Karimi, and S. Samavi, "Modeling of pruning techniques for deep neural networks simplification," *CoRR*, vol. abs/2001.04062, 2020. [Online]. Available: https://arxiv.org/abs/2001.04062

[77] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," 2021.

[78] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[79] O. Rukundo, "Effects of image size on deep learning," *Electronics*, vol. 12, no. 4, p. 985, Feb. 2023. [Online]. Available: http://dx.doi.org/10.3390/electronics12040985

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

[80] D. Hammerstrom, "A vlsi architecture for high-performance, low-cost, on-chip learning," in *1990 IJCNN International Joint Conference on Neural Networks*, 1990, pp. 537–544 vol.2.

[81] B. Moons, K. Goetschalckx, N. Van Berckelaer, and M. Verhelst, "Minimum energy quantized neural networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 1921–1925.

[82] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," 2021.

[83] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," 2016.

[84] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," 2016.

[85] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016.

[86] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2018.

[87] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," 2016.

[88] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020.

[89] Maaten, L.v.d., and G. Hinton, "Visualizing data using t-sne," 2008.

[90] X. Chen, Q. Guan, X. Liang, L.-T. Lo, S. Su, T. Estrada, and J. Ahrens, "Tensorview: visualizing the training of convolutional neural network using paraview," 12 2017, pp. 11–16.

[91] H. Cai, C. Gan, L. Zhu, and S. Han, "Tinytl: Reduce memory, not parameters for efficient on-device learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 11 285–11 297. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/81f7acabd411274fcf65ce2070ed568a-Paper.pdf

[92] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256kb memory," 2022.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[93] S. Zhu, T. Voigt, J. Ko, and F. Rahimian, "On-device training: A first overview on existing systems," 2023.

[94] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84 – 90, may 2017. [Online]. Available: https://doi.org/10.1145/3065386

[95] X. Peng, X. Shi, H. Dai, H. Jin, W. Ma, Q. Xiong, F. Yang, and X. Qian, "Capuchin: Tensor-based gpu memory management for deep learning," 03 2020, pp. 891–905.

[96] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," *CoRR*, vol. abs/1604.06174, 2016. [Online]. Available: http://arxiv.org/abs/1604.06174

[97] Y. Ro and J. Y. Choi, "Autolr: Layer-wise pruning and auto-tuning of learning rates in fine-tuning of deep networks," 2021.

[98] B. Barakat and Q. Huang, "Improving reliability of fine-tuning with block-wise optimisation," 2023.

[99] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196 197–196 211, 2020.

[100] R. Pan, X. Liu, S. Diao, R. Pi, J. Zhang, C. Han, and T. Zhang, "Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning," 2024.

[101] Y. Liu, S. Agarwal, and S. Venkataraman, "Autofreeze: Automatically freezing model blocks to accelerate fine-tuning," *CoRR*, vol. abs/2102.01386, 2021. [Online]. Available: https://arxiv.org/abs/2102.01386

[102] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," 2017.

[103] A. Ardakani, A. Haan, S. Tan, D. T. Popovici, A. Cheung, C. Iancu, and K. Sen, "Slimfit: Memory-efficient fine-tuning of transformer-based models using training dynamics," 2023.

[104] R. N. Wanjiku, L. Nderu, and M. Kimwele, "Dynamic fine-tuning layer selection using kullback – leibler divergence," *Engineering Reports*, vol. 5, no. 5, p. e12595, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/eng2.12595

[105] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012. [Online]. Available: http://jmlr.org/papers/v13/bergstra12a.html

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[106] (2023-11-22) Simple audio recognition: Recognizing keywords (tensorflow tutorial). https://www.tensorflow.org/tutorials/audio/simple_audio.

[107] T. Shibata and K. Wada, "Robot therapy: A new approach for mental healthcare of the elderly. a mini-review," *Gerontology*, vol. 57, pp. 378–86, 06 2011.

[108] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *European Conference on Computer Vision (ECCV)*, 2018.

[109] K. Wang and W. Zhou, "Pedestrian and cyclist detection based on deep neural network fast r-cnn," *International Journal of Advanced Robotic Systems*, vol. 16, 03 2019.

[110] X. Zhu and M. Bain, "B-cnn: Branch convolutional neural network for hierarchical classification," 2017.

[111] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," 2016.

[112] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," 2021.

[113] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[114] J. Deng, K. Li, M. Do, H. Su, and L. Fei-Fei, "Construction and Analysis of a Large Scale Image Ontology." Vision Sciences Society, 2009.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Appendix

While the Result chapter concentrates on the essential insights gained during the experiments carried out as part of the thesis, the Appendix section provides raw measurements and further information. Starting with the definition of each -schedule used in the featured experiments. Further, all archived best test-accuracies with their corresponding base learning rate are listed for each training run in the Grid Search Results section. This is followed by tables showing the similarity measurements.

.1 BPLS-schedules

4442			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-

4432			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	-
n = 1	-	n = 1	-

4422			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
n = 1	-	n = 1	-

4422_m			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
-	n = 1	-	n = 1

4222			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
n = 1	-	n = 1	-
n = 1	-	n = 1	-

4211_m			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
-	n = 1	-	-
-	-	-	n = 1

4222_m (Key-Word Scenario)			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
-	n = 1	-	n = 1
-	n = 1	-	n = 1

4222_m (Cifar10 Scenario)			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
n = 1	-	-	-
-	n = 1	n = 1	n = 1

4421			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
n = 1	-	-	-

Figure 1: -schedules used in Cifar10 and Key-Word scenarios

444442			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-

444421			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
n = 1	-	-	-

A98765									
Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	Step 10
n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	-
n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	n = 1	-	n = 1	-
n = 1	n = 1	n = 1	n = 1	n = 1	-	n = 1	-	n = 1	-
n = 1	n = 1	n = 1	-	n = 1	-	n = 1	-	n = 1	-
n = 1	-	n = 1	-	n = 1	-	n = 1	-	n = 1	-

663333_m					
Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
n = 1	n = 1	n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1	n = 1	n = 1
n = 1	-	-	n = 1	-	n = 1
-	n = 1	-	-	n = 1	-
-	-	n = 1	-	-	n = 1
-	-	n = 1	-	-	n = 1

633111_m					
Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
n = 1	n = 1	n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	-	n = 1	n = 1	-
n = 1	-	-	n = 1	-	-
-	n = 1	-	-	n = 1	-
-	-	n = 1	-	-	n = 1

444222_m			
Step 1	Step 2	Step 3	Step 4
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	n = 1	n = 1	n = 1
n = 1	-	n = 1	-
-	n = 1	-	n = 1
-	n = 1	-	n = 1

Figure 2: -schedules used in Cifar100 scenario

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

.2 Grid Search Results

4444		4440		4444_lr4442		4444_lr4421		4444_lr4432		4444_lr4422		4442		4421		4432		4422		4422_m	
lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc
5669E-6	10.0%	5669E-6	11.25%	5669E-6	10.0%	5669E-6	48.125%	5669E-6	10.0%	5669E-6	11.25%	5669E-6	11.875%	5669E-6	11.25%	5669E-6	10.0%	5669E-6	10.0%	5669E-6	10.0%
5154E-6	10.0%	5154E-6	12.5%	5154E-6	10.0%	5154E-6	73.125%	5154E-6	50.0%	5154E-6	11.875%	5154E-6	11.25%	5154E-6	11.25%	5154E-6	10.0%	5154E-6	11.25%	5154E-6	10.0%
4724E-6	10.0%	4685E-6	13.75%	4724E-6	11.25%	4724E-6	70.0%	4685E-6	12.5%	4724E-6	11.25%	4724E-6	10.0%	4724E-6	10.0%	4685E-6	10.0%	4724E-6	11.25%	4724E-6	10.0%
4685E-6	10.0%	4260E-6	13.125%	4685E-6	11.25%	4685E-6	48.125%	4260E-6	11.25%	4685E-6	34.375%	4685E-6	10.0%	4685E-6	76.25%	4260E-6	10.0%	4685E-6	10.625%	4685E-6	10.0%
4260E-6	11.25%	3872E-6	53.125%	4260E-6	10.0%	4260E-6	86.25%	3872E-6	11.25%	4260E-6	15.0%	4260E-6	10.0%	4260E-6	85.625%	3872E-6	10.0%	4260E-6	11.25%	4260E-6	10.0%
3937E-6	11.25%	3520E-6	46.875%	3937E-6	12.5%	3937E-6	61.875%	3520E-6	83.125%	3937E-6	13.125%	3937E-6	11.25%	3937E-6	11.875%	3520E-6	58.125%	3937E-6	11.25%	3937E-6	11.25%
3872E-6	11.25%	3200E-6	46.25%	3872E-6	11.25%	3872E-6	75.625%	3200E-6	92.5%	3872E-6	11.25%	3872E-6	11.25%	3872E-6	11.25%	3200E-6	11.25%	3872E-6	10.0%	3872E-6	11.25%
3520E-6	11.25%	2909E-6	66.875%	3520E-6	11.25%	3520E-6	38.75%	2909E-6	90.625%	3520E-6	11.25%	3520E-6	11.25%	3520E-6	54.375%	2909E-6	86.875%	3520E-6	11.875%	3520E-6	10.0%
3281E-6	11.25%	2645E-6	75.625%	3281E-6	19.375%	3281E-6	58.75%	2645E-6	91.875%	3281E-6	15.0%	3281E-6	11.25%	3281E-6	11.25%	2645E-6	11.25%	3281E-6	10.0%	3281E-6	10.0%
3200E-6	11.25%	2404E-6	81.25%	3200E-6	30.625%	3200E-6	81.875%	2404E-6	81.875%	3200E-6	46.25%	3200E-6	11.875%	3200E-6	11.25%	2404E-6	11.875%	3200E-6	11.25%	3200E-6	10.0%
2909E-6	85.0%	2186E-6	80.625%	2909E-6	11.875%	2909E-6	65.625%	2186E-6	28.125%	2909E-6	65.0%	2909E-6	47.5%	2909E-6	11.25%	2186E-6	11.875%	2909E-6	11.875%	2909E-6	27.5%
2734E-6	11.25%	1987E-6	76.875%	2734E-6	31.875%	2734E-6	89.375%	1987E-6	85.625%	2734E-6	44.375%	2734E-6	90.0%	2734E-6	22.5%	1987E-6	39.375%	2734E-6	18.125%	2734E-6	32.5%
2645E-6	11.25%	1806E-6	76.25%	2645E-6	43.75%	2645E-6	70.0%	1806E-6	23.75%	2645E-6	30.625%	2645E-6	11.875%	2645E-6	11.875%	1806E-6	11.25%	2645E-6	11.875%	2645E-6	71.875%
2404E-6	41.25%	1642E-6	76.25%	2404E-6	71.875%	2404E-6	48.125%	1642E-6	11.25%	2404E-6	86.25%	2404E-6	55.0%	2404E-6	12.5%	1642E-6	95.0%	2404E-6	11.25%	2404E-6	11.25%
2278E-6	40.0%	1493E-6	80.0%	2278E-6	47.5%	2278E-6	38.75%	1493E-6	78.125%	2278E-6	85.0%	2278E-6	11.875%	2278E-6	13.125%	1493E-6	85.0%	2278E-6	45.625%	2278E-6	31.875%
2186E-6	11.25%	1357E-6	69.375%	2186E-6	51.875%	2186E-6	80.0%	1357E-6	46.25%	2186E-6	11.25%	2186E-6	33.75%	2186E-6	87.5%	1357E-6	89.375%	2186E-6	16.875%	2186E-6	11.875%
1987E-6	12.5%	1234E-6	68.75%	1987E-6	26.875%	1987E-6	51.25%	1234E-6	26.875%	1987E-6	25.0%	1987E-6	11.875%	1987E-6	49.375%	1234E-6	11.25%	1987E-6	11.875%	1987E-6	74.375%
1899E-6	88.75%	1122E-6	71.25%	1899E-6	29.375%	1899E-6	78.125%	1122E-6	38.75%	1899E-6	31.875%	1899E-6	47.5%	1899E-6	23.125%	1122E-6	20.0%	1899E-6	11.25%	1899E-6	38.75%
1806E-6	11.25%	1020E-6	58.75%	1806E-6	88.75%	1806E-6	65.625%	1020E-6	46.875%	1806E-6	88.125%	1806E-6	69.375%	1806E-6	82.5%	1020E-6	90.625%	1806E-6	12.5%	1806E-6	11.25%
1642E-6	90.0%	927E-6	65.625%	1642E-6	11.875%	1642E-6	71.875%	927E-6	23.75%	1642E-6	26.875%	1642E-6	31.875%	1642E-6	12.5%	927E-6	29.375%	1642E-6	29.375%	1642E-6	72.5%
1582E-6	75.625%	843E-6	63.125%	1582E-6	58.75%	1582E-6	58.125%	843E-6	35.625%	1582E-6	80.625%	1516E-6	90.625%	1582E-6	61.875%	843E-6	15.625%	1582E-6	23.75%	1582E-6	35.625%
1493E-6	21.875%	766E-6	58.125%	1493E-6	46.875%	1493E-6	68.125%	766E-6	56.25%	1493E-6	52.5%	1493E-6	14.375%	1493E-6	22.5%	766E-6	91.25%	1493E-6	14.375%	1493E-6	11.875%
1357E-6	75.625%	696E-6	58.125%	1357E-6	18.75%	1357E-6	27.5%	696E-6	40.625%	1357E-6	61.25%	1357E-6	75.625%	1357E-6	23.125%	696E-6	17.5%	1357E-6	20.0%	1357E-6	11.25%
1319E-6	46.875%	633E-6	53.125%	1319E-6	75.0%	1319E-6	30.625%	633E-6	30.0%	1319E-6	69.375%	1319E-6	78.125%	1319E-6	23.75%	633E-6	41.25%	1319E-6	25.0%	1319E-6	11.25%
1234E-6	90.625%	576E-6	53.125%	1234E-6	86.875%	1234E-6	58.125%	576E-6	27.5%	1234E-6	65.625%	1234E-6	11.875%	1234E-6	41.875%	576E-6	28.75%	1234E-6	11.25%	1234E-6	55.625%
1122E-6	51.875%			1122E-6	30.625%	1122E-6	73.125%			1122E-6	90.625%	1122E-6	38.75%	1122E-6	50.625%			1122E-6	46.875%	1122E-6	20.0%
1099E-6	11.25%			1099E-6	88.125%	1099E-6	59.375%			1099E-6	31.25%	1099E-6	51.875%	1099E-6	86.875%			1099E-6	26.875%	1099E-6	67.5%
1020E-6	35.625%			1020E-6	49.375%	1020E-6	60.625%			1020E-6	36.875%	1020E-6	21.25%	1020E-6	70.0%			1020E-6	16.875%	1020E-6	21.25%
927E-6	19.375%			927E-6	28.125%	927E-6	63.125%			927E-6	58.75%	927E-6	45.625%	927E-6	35.625%			927E-6	17.5%	927E-6	31.875%
916E-6	30.625%			916E-6	40.625%	916E-6	73.125%			916E-6	19.375%	916E-6	16.875%	916E-6	12.5%			916E-6	22.5%	916E-6	19.375%
843E-6	35.0%			843E-6	36.25%	843E-6	58.75%			843E-6	36.875%	843E-6	21.25%	843E-6	61.25%			843E-6	19.375%	843E-6	24.375%
766E-6	91.25%			766E-6	30.625%	766E-6	63.75%			766E-6	56.875%	766E-6	43.75%	766E-6	16.875%			766E-6	51.25%	766E-6	23.125%
763E-6	23.125%			763E-6	30.625%	763E-6	70.625%			763E-6	56.25%	763E-6	25.0%	763E-6	52.5%			763E-6	11.25%	763E-6	48.75%
696E-6	26.25%			696E-6	40.625%	696E-6	61.875%			696E-6	23.75%	696E-6	24.375%	696E-6	60.625%			696E-6	15.0%	696E-6	19.375%
636E-6	20.0%			636E-6	59.375%	636E-6	50.0%			636E-6	30.625%	636E-6	18.125%	636E-6	54.375%			636E-6	16.25%	636E-6	56.875%
633E-6	23.75%			633E-6	86.25%	633E-6	61.875%			633E-6	28.75%	633E-6	23.75%	633E-6	54.375%			633E-6	20.625%	633E-6	65.0%
576E-6	90.625%			576E-6	31.25%	576E-6	53.125%			576E-6	41.875%	576E-6	26.875%	576E-6	60.0%			576E-6	16.25%	576E-6	65.0%

Table 1: KW - Network: Grid search result (after 350 Epochs)

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

4444_lr4222		4222		4222_m		4211_m	
lr	acc	lr	acc	lr	acc	lr	acc
3168E-6	90.625%	2680E-6	87.5%	1800E-6	90.0%	2431E-6	83.75%
2985E-6	90.625%	2550E-6	90.0%	1646E-6	85.625%	1449E-6	88.75%
2812E-6	89.375%	2451E-6	86.875%	1322E-6	90.0%	1435E-6	90.0%
2675E-6	91.25%	1836E-6	88.125%			1406E-6	89.375%
2623E-6	90.625%	1782E-6	86.875%			1365E-6	87.5%
2597E-6	92.5%	1765E-6	81.25%			1338E-6	90.0%
2192E-6	92.5%	1662E-6	88.125%			1312E-6	90.0%
2025E-6	81.875%	1535E-6	90.0%			1299E-6	73.125%
1965E-6	86.25%	1418E-6	90.625%			1286E-6	39.375%
1946E-6	91.25%	1139E-6	85.625%			1273E-6	75.625%
1888E-6	92.5%	897E-6	85.625%			1248E-6	76.875%
1833E-6	88.125%	871E-6	86.875%			1224E-6	67.5%
1195E-6	93.125%					1153E-6	88.75%
1009E-6	91.25%					1064E-6	86.25%
						908E-6	91.25%

Table 2: KW - Network: Grid search result (after 350 Epochs)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

4444		4440		4444_lr4442		4444_lr4422		4444_lr4421		4444_lr4432	
lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc
753E-6	77.04%	1476E-6	74.42%	2066E-6	73.92%	248E-6	77.87%	2066E-6	74.96%	248E-6	77.78%
538E-6	76.82%	1054E-6	75.16%	1476E-6	74.91%	225E-6	77.96%	1476E-6	76.22%	225E-6	78.1%
384E-6	77.43%	753E-6	76.28%	1054E-6	75.86%	205E-6	77.83%	1054E-6	76.57%	205E-6	77.96%
274E-6	77.8%	538E-6	76.75%	753E-6	76.73%	186E-6	77.95%	753E-6	76.74%	186E-6	78.08%
196E-6	77.84%	384E-6	77.33%	538E-6	77.17%	169E-6	78.09%	538E-6	77.15%	169E-6	78.1%
140E-6	78.02%	274E-6	77.12%	384E-6	77.38%	154E-6	77.78%	384E-6	77.58%	154E-6	78.04%
100E-6	78.06%	196E-6	77.38%	274E-6	77.32%	140E-6	77.8%	274E-6	77.81%	140E-6	78.0%
71E-6	77.97%	140E-6	77.72%	196E-6	78.09%	127E-6	77.89%	196E-6	77.53%	127E-6	77.85%
51E-6	77.58%	100E-6	77.69%	140E-6	78.11%	116E-6	77.84%	140E-6	77.63%	116E-6	77.86%
161E-6	77.94%	71E-6	77.31%	100E-6	78.05%	105E-6	77.75%	100E-6	77.71%	105E-6	77.78%
146E-6	78.12%	225E-6	77.19%	225E-6	77.73%	96E-6	77.59%	441E-6	77.17%	96E-6	77.7%
133E-6	78.17%	205E-6	77.32%	205E-6	78.05%	538E-6	77.19%	401E-6	77.59%	538E-6	77.41%
121E-6	78.1%	186E-6	77.56%	186E-6	77.98%	489E-6	77.17%	365E-6	77.44%	489E-6	77.46%
110E-6	78.09%	169E-6	77.61%	169E-6	78.06%	444E-6	77.47%	332E-6	77.76%	444E-6	77.42%
91E-6	77.96%	154E-6	77.53%	154E-6	78.13%	404E-6	77.03%	301E-6	77.74%	404E-6	77.63%
83E-6	78.13%	127E-6	77.71%	127E-6	78.07%	367E-6	77.5%	249E-6	77.77%	367E-6	77.85%
36E-6	77.11%	116E-6	77.72%	116E-6	77.86%	334E-6	77.74%	226E-6	77.58%	334E-6	77.76%
26E-6	75.88%	51E-6	76.66%	105E-6	78.03%	304E-6	77.52%	206E-6	77.65%	304E-6	77.71%
19E-6	74.83%	36E-6	75.86%	96E-6	78.02%	276E-6	77.66%	187E-6	77.53%	276E-6	77.68%
13E-6	73.61%	26E-6	74.78%	87E-6	77.97%	251E-6	78.05%			251E-6	77.58%
9E-6	71.88%	19E-6	73.96%			228E-6	77.81%			228E-6	77.94%
		13E-6	72.58%			207E-6	77.76%			207E-6	77.98%

4442		4422		4422_m		4421		4432		4211_m	
lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc
2066E-6	74.68%	225E-6	78.12%	225E-6	78.17%	2066E-6	74.31%	225E-6	77.92%	384E-6	77.16%
1476E-6	76.56%	205E-6	77.92%	205E-6	77.81%	1476E-6	74.98%	205E-6	78.05%	349E-6	77.2%
1054E-6	75.96%	186E-6	78.15%	186E-6	77.88%	1054E-6	76.51%	186E-6	78.04%	317E-6	77.68%
753E-6	76.48%	169E-6	78.04%	169E-6	78.02%	753E-6	77.35%	169E-6	78.09%	289E-6	77.61%
538E-6	77.34%	154E-6	78.0%	154E-6	77.84%	538E-6	77.43%	154E-6	78.2%	262E-6	77.42%
384E-6	77.45%	140E-6	78.01%	140E-6	77.88%	384E-6	77.55%	140E-6	77.94%	239E-6	77.7%
274E-6	77.62%	127E-6	78.0%	127E-6	77.82%	274E-6	78.0%	127E-6	78.04%	217E-6	77.83%
196E-6	78.11%	116E-6	77.91%	116E-6	77.95%	196E-6	78.1%	116E-6	78.1%	197E-6	77.78%
140E-6	78.17%	105E-6	77.7%	105E-6	77.71%	140E-6	77.92%	105E-6	78.07%	179E-6	78.15%
100E-6	78.13%	96E-6	77.78%	96E-6	77.72%	100E-6	77.94%	96E-6	77.99%	163E-6	77.8%
225E-6	77.96%	87E-6	77.65%	87E-6	77.66%	225E-6	77.93%	87E-6	77.95%	148E-6	77.84%
205E-6	77.95%	538E-6	77.14%	300E-6	77.57%	205E-6	77.93%	538E-6	77.12%	135E-6	77.71%
186E-6	78.02%	489E-6	77.29%	273E-6	77.62%	186E-6	78.02%	489E-6	77.14%	122E-6	77.67%
169E-6	78.17%	444E-6	77.17%	248E-6	78.12%	169E-6	78.06%	444E-6	77.32%	111E-6	77.57%
154E-6	78.07%	404E-6	77.57%	538E-6	77.19%	154E-6	77.89%	404E-6	77.56%	101E-6	77.59%
127E-6	78.15%	367E-6	77.75%	489E-6	77.36%	127E-6	77.97%	367E-6	77.83%	92E-6	77.27%
116E-6	78.1%	334E-6	77.21%	444E-6	77.41%	116E-6	78.01%	334E-6	77.44%	84E-6	77.06%
105E-6	78.23%	304E-6	77.5%	404E-6	77.31%	105E-6	77.85%	304E-6	77.59%	76E-6	76.95%
96E-6	78.25%	276E-6	77.78%	367E-6	77.36%	96E-6	77.93%	276E-6	77.82%		
87E-6	78.2%	251E-6	77.86%	334E-6	77.61%	87E-6	77.74%	251E-6	77.83%		
		228E-6	77.89%	304E-6	77.72%			228E-6	77.94%		
		207E-6	78.15%	276E-6	77.6%			207E-6	78.01%		
				251E-6	77.76%						
				228E-6	77.94%						
				207E-6	77.93%						

Table 3: Cifar10 - Network: Grid search result (after 30 Epochs)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

444444		444440		444444_lr444442		444444_lrA98765		444444_lr444421		444442		A98765		444421		444222_m		663333_m		633111_m	
lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc	lr	acc
2893E-6	54.4%	2893E-6	54.19%	2893E-6	55.63%	2893E-6	56.32%	2893E-6	57.05%	2893E-6	52.71%	2893E-6	53.26%	2893E-6	51.49%	2893E-6	52.85%	2893E-6	51.49%	1403E-6	56.48%
2410E-6	56.35%	2410E-6	55.51%	2410E-6	56.82%	2410E-6	59.47%	2410E-6	57.4%	2410E-6	54.46%	2410E-6	54.85%	2410E-6	53.07%	2066E-6	55.17%	2066E-6	53.46%	1275E-6	57.31%
2009E-6	56.85%	2009E-6	56.64%	2009E-6	58.43%	2009E-6	59.07%	2009E-6	59.79%	2009E-6	56.49%	2009E-6	55.99%	2009E-6	54.73%	1476E-6	57.11%	1476E-6	55.88%	1159E-6	56.9%
1674E-6	57.4%	1674E-6	58.15%	1674E-6	57.83%	1674E-6	59.85%	1674E-6	58.87%	1674E-6	57.05%	1674E-6	56.96%	1674E-6	56.26%	1054E-6	58.02%	1054E-6	57.34%	1054E-6	57.62%
1395E-6	59.77%	1395E-6	57.65%	1395E-6	60.08%	1395E-6	59.91%	1395E-6	59.57%	1395E-6	58.96%	1395E-6	58.33%	1395E-6	57.09%	753E-6	59.15%	753E-6	57.81%	958E-6	57.81%
1162E-6	60.09%	1162E-6	58.52%	1162E-6	59.63%	1162E-6	60.26%	1162E-6	59.48%	1162E-6	59.37%	1162E-6	58.13%	1162E-6	59.63%	538E-6	59.5%	538E-6	57.87%	871E-6	57.63%
969E-6	60.47%	969E-6	58.02%	969E-6	59.76%	969E-6	59.24%	969E-6	59.47%	969E-6	59.67%	969E-6	60.06%	969E-6	59.07%	958E-6	58.48%	958E-6	57.05%	792E-6	56.95%
807E-6	60.32%	807E-6	58.17%	807E-6	59.83%	807E-6	59.58%	807E-6	59.2%	807E-6	59.95%	807E-6	59.93%	807E-6	59.81%	871E-6	58.59%	871E-6	57.7%	720E-6	59.08%
673E-6	60.25%	673E-6	57.75%	673E-6	59.84%	673E-6	58.81%	673E-6	58.6%	673E-6	59.92%	673E-6	59.83%	673E-6	58.97%	792E-6	58.24%	792E-6	57.75%	654E-6	57.74%
561E-6	59.64%	561E-6	58.06%	561E-6	58.79%	561E-6	58.19%	561E-6	57.65%	561E-6	59.37%	561E-6	59.34%	561E-6	58.6%	720E-6	59.7%	720E-6	57.35%	595E-6	58.63%
467E-6	59.57%	467E-6	56.72%	467E-6	58.74%	467E-6	57.48%	467E-6	57.33%	467E-6	58.95%	467E-6	59.12%	467E-6	58.43%	654E-6	59.35%	654E-6	57.57%	541E-6	58.75%
1268E-6	59.33%	389E-6	56.15%	389E-6	58.0%	389E-6	56.9%	2630E-6	56.63%	389E-6	58.39%	389E-6	58.87%	389E-6	58.01%	595E-6	59.44%	595E-6	58.82%	492E-6	58.34%
1153E-6	60.05%	1522E-6	58.12%	1826E-6	58.07%	1522E-6	60.26%	2391E-6	57.3%	1056E-6	59.29%	1268E-6	58.07%	1056E-6	58.64%	541E-6	58.96%	541E-6	58.88%	447E-6	58.03%
1048E-6	60.83%	1383E-6	57.98%	1660E-6	58.6%	1383E-6	60.01%	2174E-6	58.59%	960E-6	59.82%	1153E-6	58.86%	960E-6	59.52%	492E-6	58.51%	492E-6	58.51%	406E-6	58.1%
953E-6	60.24%	1258E-6	58.53%	1509E-6	59.81%	1258E-6	59.7%	1976E-6	58.67%	873E-6	59.02%	1048E-6	59.74%	873E-6	59.01%	447E-6	58.54%	447E-6	58.55%	369E-6	58.43%
866E-6	59.56%	1143E-6	58.39%	1372E-6	59.62%	1143E-6	60.2%	1796E-6	59.5%	794E-6	59.73%	953E-6	59.84%	794E-6	59.42%	406E-6	58.5%	406E-6	58.13%	336E-6	57.06%
787E-6	60.56%	1039E-6	58.5%	1247E-6	60.24%	1039E-6	59.51%	1633E-6	59.6%	722E-6	59.51%	866E-6	59.99%	722E-6	59.11%	369E-6	57.5%	369E-6	57.5%	305E-6	56.7%
716E-6	60.46%	945E-6	58.61%	1134E-6	60.37%	945E-6	59.52%	1485E-6	59.86%	656E-6	59.83%	787E-6	60.04%	656E-6	59.01%	336E-6	57.1%	336E-6	57.17%	278E-6	56.58%
		859E-6	57.75%	1031E-6	60.19%	859E-6	59.17%			596E-6	59.84%	716E-6	59.62%	596E-6	58.7%	305E-6	57.21%	305E-6	56.39%		
																278E-6	56.61%	278E-6	56.73%		
																		541E-6	58.88%		

Table 4: Cifar100 - Network: Grid search result (after 40 Epochs)

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

.3 Cifar10 - Similarity Measurements

	4444	4444 $lr4421$	4444 $lr4442$	4444 $lr4422$	4421	4442	4422	4422 $_m$	4421 $_{rdo}$	4442 $_{rdo}$	4422 $_{rdo}$	4421 $lr4444$	4442 $lr4444$	4422 $lr4444$	4422 $m\,lr4444$
4444	0.0	2.342	1.542	1.679	2.328	1.563	1.7	1.742	2.57	1.902	2.026	1.777	1.109	1.492	1.597
4444 $lr4421$	2.159	0.0	0.918	0.864	0.4	0.971	0.969	1.022	1.037	1.426	1.44	2.481	2.268	2.286	2.37
4444 $lr4442$	1.372	0.889	0.0	0.622	0.91	0.469	0.71	0.797	1.341	1.121	1.273	1.9	1.558	1.65	1.756
4444 $lr4422$	1.582	0.889	0.677	0.0	0.962	0.717	0.619	0.687	1.367	1.287	1.196	2.033	1.699	1.816	1.899
4421	2.623	0.52	1.18	1.148	0.0	1.198	1.108	1.328	1.272	1.693	1.729	2.882	2.731	2.701	2.862
4442	1.341	0.91	0.444	0.631	0.898	0.0	0.646	0.713	1.333	1.128	1.241	1.761	1.386	1.54	1.613
4422	1.798	1.117	0.886	0.719	1.032	0.856	0.0	0.967	1.561	1.446	1.416	2.099	1.862	1.85	2.121
4422 $_m$	1.791	1.148	0.953	0.766	1.201	0.907	0.931	0.0	1.58	1.478	1.441	2.232	1.866	2.045	1.865
4421 $_{rdo}$	2.867	1.204	1.64	1.573	1.272	1.674	1.658	1.714	0.0	2.05	1.564	3.202	2.987	3.002	3.092
4442 $_{rdo}$	2.12	1.683	1.323	1.486	1.693	1.371	1.543	1.615	2.05	0.0	1.972	2.651	2.299	2.433	2.536
4422 $_{rdo}$	2.026	1.537	1.398	1.251	1.575	1.411	1.363	1.413	1.409	1.798	0.0	2.428	2.133	2.244	2.319
4421 $lr4444$	1.777	2.689	2.119	2.149	2.565	2.031	1.99	2.182	2.873	2.372	2.428	0.0	1.606	1.361	1.958
4442 $lr4444$	1.031	2.334	1.654	1.699	2.299	1.52	1.658	1.704	2.544	1.967	2.026	1.514	0.0	1.316	1.385
4422 $lr4444$	1.955	3.047	2.328	2.423	2.942	2.265	2.209	2.499	3.255	2.651	2.746	1.766	1.879	0.0	2.384
4422 $m\,lr4444$	1.32	2.235	1.692	1.724	2.213	1.613	1.721	1.556	2.428	1.979	2.016	1.657	1.23	1.537	0.0

Figure 3: Cifar10 Network - Conv4 - Distance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	4444	4444 $lr4421$	4444 $lr4442$	4444 $lr4422$	4421	4442	4422	4422 $_m$	4421 $_{rdo}$	4442 $_{rdo}$	4422 $_{rdo}$	4421 $lr4444$	4442 $lr4444$	4422 $lr4444$	4422 $m_l r4444$
4444	0.0	2.5	1.466	2.365	2.541	1.576	2.44	2.509	3.045	1.526	2.935	2.569	1.51	2.386	2.503
4444 $lr4421$	2.28	0.0	2.1	0.616	0.993	2.098	1.031	1.073	1.733	2.108	1.813	2.648	2.266	2.626	2.704
4444 $lr4442$	1.236	2.036	0.0	2.017	2.132	1.063	2.125	2.165	2.609	0.929	2.597	2.344	1.527	2.196	2.27
4444 $lr4422$	2.227	0.643	2.144	0.0	1.105	2.14	1.004	1.047	1.833	2.151	1.785	2.659	2.212	2.636	2.717
4421	2.907	1.297	2.723	1.381	0.0	2.652	1.01	1.664	2.314	2.702	2.374	2.947	2.832	2.977	3.365
4442	1.27	1.968	1.013	1.948	2.005	0.0	2.008	2.062	2.522	0.914	2.513	2.108	1.143	2.011	2.071
4422	2.58	1.237	2.519	1.158	0.907	2.463	0.0	1.506	2.213	2.499	2.177	2.662	2.503	2.605	3.092
4422 $_m$	2.578	1.243	2.503	1.164	1.476	2.475	1.448	0.0	2.171	2.52	2.163	3.011	2.565	3.008	2.644
4421 $_{rdo}$	3.441	2.056	3.263	2.135	2.314	3.248	2.355	2.375	0.0	3.263	1.105	3.814	3.433	3.805	3.876
4442 $_{rdo}$	1.843	2.641	1.356	2.609	2.702	1.407	2.695	2.799	3.263	0.0	3.243	3.071	2.04	2.892	3.035
4422 $_{rdo}$	2.935	1.945	2.881	1.87	2.127	2.874	2.074	2.112	0.926	2.886	0.0	3.308	2.931	3.295	3.398
4421 $lr4444$	2.569	2.895	2.687	2.82	2.555	2.529	2.513	2.931	3.371	2.624	3.308	0.0	2.362	1.798	2.988
4442 $lr4444$	1.391	2.338	1.659	2.212	2.329	1.306	2.228	2.347	2.876	1.57	2.778	2.204	0.0	2.089	2.265
4422 $lr4444$	3.128	3.629	3.294	3.534	3.285	3.181	3.115	3.671	4.143	3.234	4.051	2.456	2.977	0.0	3.794
4422 $mlr4444$	2.076	2.534	2.176	2.462	2.531	2.071	2.504	2.195	2.986	2.171	2.946	2.504	2.013	2.488	0.0

Figure 4: Cifar10 Network - Conv5 - Distance

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	4444	4444 $lr4421$	4444 $lr4442$	4444 $lr4422$	4421	4442	4422	4422 $_m$	4421 $_{rdo}$	4442 $_{rdo}$	4422 $_{rdo}$	4421 $lr4444$	4442 $lr4444$	4422 $lr4444$	4422 m_lr4444
4444	0.0	3.069	1.813	2.788	3.126	1.977	2.925	3.009	3.168	1.879	2.878	3.114	1.954	2.877	2.812
4444 $lr4421$	2.728	0.0	2.358	1.227	1.866	2.296	1.926	1.853	1.306	2.342	1.628	3.175	2.671	3.071	3.009
4444 $lr4442$	1.486	2.26	0.0	2.154	2.454	1.411	2.386	2.428	2.386	1.197	2.281	2.761	1.903	2.587	2.497
4444 $lr4422$	2.573	1.286	2.346	0.0	2.034	2.271	1.866	1.815	1.638	2.328	1.422	3.137	2.504	3.02	2.957
4421	3.709	2.526	3.401	2.642	0.0	3.17	1.938	3.119	2.744	3.295	2.863	3.56	3.51	3.531	4.067
4442	1.548	2.099	1.341	1.98	2.155	0.0	2.103	2.203	2.202	1.141	2.097	2.424	1.37	2.305	2.215
4422	3.147	2.352	3.009	2.195	1.729	2.805	0.0	2.758	2.564	2.913	2.398	3.075	2.924	2.885	3.578
4422 $_m$	3.121	2.192	2.958	2.05	2.717	2.839	2.658	0.0	2.36	2.95	2.321	3.744	3.098	3.684	2.907
4421 $_{rdo}$	3.769	1.884	3.361	2.199	2.744	3.288	2.86	2.748	0.0	3.35	1.858	4.322	3.737	4.218	4.146
4442 $_{rdo}$	2.324	3.162	1.802	3.036	3.295	1.84	3.233	3.402	3.35	0.0	3.222	3.737	2.589	3.523	3.516
4422 $_{rdo}$	2.878	1.869	2.697	1.564	2.377	2.618	2.215	2.233	1.515	2.677	0.0	3.474	2.833	3.354	3.349
4421 $lr4444$	3.114	3.558	3.243	3.385	2.987	3.0	2.85	3.62	3.66	3.127	3.474	0.0	2.778	2.036	3.363
4442 $lr4444$	1.781	2.787	2.09	2.504	2.717	1.605	2.498	2.76	2.888	1.927	2.61	2.552	0.0	2.408	2.534
4422 $lr4444$	3.936	4.59	4.133	4.377	4.001	3.922	3.654	4.721	4.749	4.004	4.5	2.913	3.645	0.0	4.508
4422 $mlr4444$	2.243	2.764	2.38	2.594	2.827	2.215	2.731	2.243	2.841	2.336	2.724	2.74	2.2	2.727	0.0

Figure 5: Cifar10 Network - Conv6 - Distance

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	4444	4444 $lr4421$	4444 $lr4442$	4444 $lr4422$	4421	4442	4422	4422 $_m$	4421 $_{rdo}$	4442 $_{rdo}$	4422 $_{rdo}$	4421 $lr4444$	4442 $lr4444$	4422 $lr4444$	4422 $m_l r4444$
4444	0.0	1.977	1.048	1.761	2.023	1.168	1.873	1.965	2.024	1.111	1.821	2.09	1.174	1.867	1.834
4444 $lr4421$	1.751	0.0	1.465	0.693	1.16	1.455	1.163	1.167	0.717	1.494	0.949	2.151	1.729	2.047	2.038
4444 $lr4442$	0.855	1.402	0.0	1.323	1.55	0.805	1.495	1.562	1.46	0.666	1.401	1.843	1.151	1.667	1.637
4444 $lr4422$	1.621	0.727	1.446	0.0	1.287	1.428	1.136	1.121	0.942	1.475	0.789	2.131	1.593	2.018	1.987
4421	2.407	1.58	2.175	1.673	0.0	2.022	1.161	2.028	1.691	2.125	1.78	2.368	2.273	2.339	2.747
4442	0.91	1.329	0.763	1.245	1.367	0.0	1.324	1.439	1.378	0.648	1.306	1.619	0.817	1.493	1.462
4422	2.021	1.437	1.91	1.343	1.033	1.775	0.0	1.757	1.567	1.861	1.459	2.049	1.865	1.892	2.403
4422 $_m$	2.04	1.379	1.909	1.266	1.771	1.847	1.691	0.0	1.467	1.941	1.44	2.543	2.036	2.493	1.928
4421 $_{rdo}$	2.422	1.045	2.093	1.271	1.691	2.072	1.752	1.705	0.0	2.129	1.063	2.919	2.407	2.813	2.788
4442 $_{rdo}$	1.378	2.03	1.017	1.932	2.125	1.065	2.07	2.232	2.129	0.0	2.043	2.521	1.6	2.316	2.344
4422 $_{rdo}$	1.821	1.093	1.671	0.87	1.479	1.637	1.346	1.386	0.865	1.686	0.0	2.357	1.802	2.231	2.251
4421 $lr4444$	2.09	2.413	2.17	2.3	1.985	2.006	1.899	2.459	2.47	2.11	2.357	0.0	1.842	1.299	2.251
4442 $lr4444$	1.067	1.803	1.27	1.593	1.761	0.956	1.591	1.812	1.856	1.181	1.657	1.693	0.0	1.541	1.664
4422 $lr4444$	2.558	3.075	2.695	2.936	2.645	2.562	2.393	3.178	3.167	2.636	3.003	1.845	2.347	0.0	3.005
4422 $mlr4444$	1.463	1.87	1.56	1.743	1.904	1.462	1.827	1.486	1.91	1.568	1.828	1.838	1.451	1.838	0.0

Figure 6: Cifar10 Network - Lin1 - Distance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	4444	4444 $lr4421$	4444 $lr4442$	4444 $lr4422$	4421	4442	4422	4422_m	4421 $_{rdo}$	4442 $_{rdo}$	4422 $_{rdo}$	4421 $lr4444$	4442 $lr4444$	4422 $lr4444$	4422_m $lr4444$
4444	0.0	29.55	17.97	24.15	30.74	20.35	25.28	27.51	48.07	33.26	36.56	29.0	18.46	25.25	26.52
4444 $lr4421$	28.95	0.0	21.63	14.45	18.96	22.11	20.68	21.08	41.07	35.58	33.14	34.39	28.89	33.17	32.7
4444 $lr4442$	17.12	21.38	0.0	18.96	24.27	14.2	21.48	23.64	45.32	31.31	34.9	29.35	20.79	26.19	26.9
4444 $lr4422$	23.66	14.64	19.55	0.0	21.86	20.47	17.53	19.1	42.46	34.31	30.81	31.32	24.13	29.54	29.44
4421	31.67	20.25	26.09	22.82	0.0	24.56	19.33	27.13	42.46	36.54	35.55	30.34	30.0	31.62	34.66
4442	19.05	21.65	13.87	19.6	22.69	0.0	19.94	21.69	45.43	32.41	35.01	27.23	16.65	24.77	24.61
4422	25.75	21.74	23.02	18.47	18.97	22.0	0.0	24.38	43.66	35.14	33.54	27.54	24.16	25.39	30.65
4422 $_m$	27.63	21.91	24.87	19.78	26.42	23.5	24.05	0.0	44.24	36.52	34.79	33.74	26.67	32.99	25.5
4421 $_{rdo}$	47.55	39.88	44.39	41.46	42.46	44.25	43.4	43.8	0.0	50.53	34.07	49.8	47.24	49.47	49.44
4442 $_{rdo}$	32.93	35.16	30.02	33.94	36.54	31.1	35.2	36.61	50.53	0.0	43.19	39.74	34.47	38.49	39.07
4422 $_{rdo}$	36.56	32.97	34.8	30.76	35.94	35.07	33.74	34.86	34.05	43.74	0.0	41.41	36.96	40.51	40.51
4421 $lr4444$	29.0	34.71	29.92	31.62	29.26	28.08	27.09	33.69	50.28	39.83	41.41	0.0	25.88	21.9	31.55
4442 $lr4444$	18.04	28.99	21.19	24.13	28.91	17.31	23.22	25.88	48.03	35.18	36.98	25.66	0.0	22.99	23.84
4422 $lr4444$	27.3	35.36	29.07	31.94	32.33	28.3	27.05	34.67	49.42	38.74	40.83	23.5	25.72	0.0	32.66
4422$_{mlr4444}$	25.06	32.32	26.55	28.75	32.82	24.61	29.04	24.02	50.02	39.09	40.31	30.5	23.05	29.09	0.0

Figure 7: Cifar10 Network - Conv4 - Cosine Similarity

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	4444	4444 $lr4421$	4444 $lr4442$	4444 $lr4422$	4421	4442	4422	4422_m	4421 $_{rdo}$	4442 $_{rdo}$	4422 $_{rdo}$	4421 $lr4444$	4442 $lr4444$	4422 $lr4444$	4422_m $lr4444$
4444	0.0	31.47	18.99	28.02	33.06	20.37	30.64	32.32	43.53	19.75	41.63	33.22	19.64	30.66	31.82
4444 $lr4421$	30.31	0.0	24.79	13.59	21.41	24.34	22.47	23.28	34.14	24.85	36.14	36.49	29.76	35.6	36.01
4444 $lr4442$	17.58	24.45	0.0	23.31	27.98	14.99	27.39	28.61	39.87	13.11	39.63	33.18	21.74	30.94	31.65
4444 $lr4422$	27.24	13.84	24.04	0.0	23.39	23.49	21.44	22.22	35.5	24.07	34.75	34.97	26.63	33.97	34.33
4421	34.69	23.65	30.48	25.31	0.0	28.72	18.21	30.06	38.95	29.94	40.16	33.03	33.0	33.05	39.01
4442	18.49	23.61	14.66	22.35	25.8	0.0	25.37	27.0	39.26	13.21	39.08	30.53	16.61	28.99	29.55
4422	31.32	24.11	29.23	22.65	17.34	27.64	0.0	29.02	39.51	28.67	38.96	30.56	29.43	28.75	37.35
4422 $_m$	32.61	24.6	30.26	23.08	28.89	29.28	28.52	0.0	39.5	30.48	39.52	39.06	32.08	38.53	29.64
4421 $_{rdo}$	44.57	34.44	41.23	35.95	38.95	40.75	39.81	40.04	0.0	41.16	17.55	48.72	44.23	48.17	48.33
4442 $_{rdo}$	21.55	27.72	15.73	26.67	29.94	16.3	29.52	31.84	41.16	0.0	40.83	35.69	23.84	33.4	34.74
4422 $_{rdo}$	41.63	36.28	40.11	34.77	39.71	39.75	38.76	39.43	16.19	40.1	0.0	46.5	41.39	45.91	46.69
4421 $lr4444$	33.22	37.61	34.63	35.74	31.02	32.46	29.79	38.8	47.57	33.77	46.5	0.0	30.43	22.69	37.77
4442 $lr4444$	18.94	30.13	22.5	26.63	30.74	17.62	28.08	31.03	42.82	21.26	41.03	29.72	0.0	28.03	30.09
4422 $lr4444$	33.72	39.86	35.43	38.0	34.38	34.11	31.11	41.07	48.83	34.74	47.48	25.95	31.98	0.0	40.06
4422 $mlr4444$	29.8	35.22	31.15	33.08	35.79	29.55	34.87	27.32	46.39	31.05	45.63	35.77	28.85	35.43	0.0

Figure 8: Cifar10 Network - Conv5 - Cosine Similarity

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	4444	4444 $lr4421$	4444 $lr4442$	4444 $lr4422$	4421	4442	4422	4422_m	4421 $_{rdo}$	4442 $_{rdo}$	4422 $_{rdo}$	4421 $lr4444$	4442 $lr4444$	4422 $lr4444$	4422_m $lr4444$
4444	0.0	33.48	19.97	30.4	34.04	21.76	31.92	32.91	34.38	20.7	31.29	34.74	21.5	32.11	31.34
4444 $lr4421$	31.89	0.0	27.36	13.95	21.2	26.61	21.97	21.16	14.75	27.18	18.5	37.29	31.09	36.1	35.34
4444 $lr4442$	18.07	26.89	0.0	25.68	29.17	17.06	28.51	29.07	28.26	14.49	27.14	33.91	23.1	31.8	30.65
4444 $lr4422$	29.36	14.27	26.63	0.0	22.62	25.74	20.82	20.27	18.12	26.43	15.79	36.0	28.45	34.69	33.92
4421	36.5	24.26	33.27	25.43	0.0	30.91	18.57	30.12	26.25	32.2	27.5	34.94	34.3	34.71	40.18
4442	19.34	25.59	16.66	24.18	26.2	0.0	25.7	27.02	26.72	14.17	25.56	30.45	17.02	29.01	27.85
4422	32.9	24.03	31.33	22.44	17.57	29.12	0.0	28.31	26.08	30.3	24.47	32.1	30.37	30.12	37.6
4422 $_{m}$	33.39	22.87	31.5	21.41	28.4	30.17	27.86	0.0	24.51	31.42	24.21	40.3	32.99	39.72	31.05
4421 $_{rdo}$	36.94	17.96	32.72	21.02	26.25	31.95	27.44	26.35	0.0	32.61	17.68	42.59	36.44	41.58	40.82
4442 $_{rdo}$	23.01	30.93	17.74	29.73	32.2	18.09	31.68	33.41	32.61	0.0	31.46	37.37	25.54	35.23	35.13
4422 $_{rdo}$	31.29	19.83	29.19	16.6	25.25	28.28	23.58	23.79	15.96	28.97	0.0	38.01	30.67	36.72	36.63
4421 $lr4444$	34.74	39.0	36.1	37.12	32.44	33.27	31.06	39.83	39.94	34.78	38.01	0.0	30.74	22.55	37.66
4442 $lr4444$	20.58	31.67	24.1	28.45	30.78	18.43	28.37	31.47	32.68	22.2	29.59	29.63	0.0	27.99	29.47
4422 $lr4444$	36.65	42.33	38.43	40.35	36.61	36.33	33.43	43.66	43.56	37.17	41.32	26.87	33.64	0.0	42.19
4422 $mlr4444$	28.38	34.14	30.02	32.07	34.85	27.85	33.82	27.65	34.94	29.46	33.64	34.8	27.7	34.74	0.0

Figure 9: Cifar10 Network - Conv6 - Cosine Similarity

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	4444	4444 $lr4421$	4444 $lr4442$	4444 $lr4422$	4421	4442	4422	4422_m	4421 $_{rdo}$	4442 $_{rdo}$	4422 $_{rdo}$	4421 $lr4444$	4442 $lr4444$	4422 $lr4444$	4422_m $lr4444$
4444	0.0	25.24	13.61	22.61	26.11	15.19	24.28	25.42	25.92	14.41	23.4	27.84	15.35	24.72	24.24
4444 $lr4421$	23.91	0.0	19.86	9.26	15.57	19.72	15.66	15.69	9.56	20.25	12.72	29.74	23.6	28.19	28.05
4444 $lr4442$	12.23	19.46	0.0	18.53	21.77	11.46	21.13	22.02	20.34	9.47	19.64	26.79	16.5	24.12	23.66
4444 $lr4422$	21.76	9.5	19.31	0.0	17.01	19.07	15.05	14.83	12.35	19.69	10.39	29.01	21.37	27.35	26.9
4421	28.12	17.97	25.25	19.16	0.0	23.47	13.36	23.41	19.3	24.64	20.41	27.96	26.53	27.51	32.41
4442	13.37	18.91	11.14	17.86	19.65	0.0	19.15	20.77	19.66	9.45	18.77	24.09	11.99	22.14	21.66
4422	25.12	17.36	23.61	16.34	12.59	21.95	0.0	21.55	19.0	22.97	17.77	25.72	23.16	23.63	30.17
4422 $_{m}$	25.84	16.99	24.05	15.7	22.08	23.27	21.17	0.0	18.13	24.43	17.87	32.73	25.79	31.94	24.5
4421 $_{rdo}$	28.06	11.79	24.09	14.43	19.3	23.86	20.06	19.49	0.0	24.48	12.06	34.41	27.89	32.98	32.62
4442 $_{rdo}$	16.08	23.26	11.8	22.27	24.64	12.37	24.08	25.93	24.48	0.0	23.56	30.06	18.7	27.42	27.72
4422 $_{rdo}$	23.4	13.69	21.37	10.97	18.72	20.93	17.08	17.57	10.84	21.54	0.0	30.75	23.15	28.97	29.22
4421 $lr4444$	27.84	31.25	28.74	29.98	25.8	26.54	24.83	32.3	32.09	27.87	30.75	0.0	24.49	17.36	30.28
4442 $lr4444$	14.62	24.05	17.33	21.37	23.73	13.01	21.52	24.51	24.84	16.07	22.27	23.56	0.0	21.34	23.05
4422 $lr4444$	28.31	33.44	29.71	32.08	29.0	28.25	26.27	35.11	34.57	28.98	32.83	20.71	25.95	0.0	33.75
4422 $mlr4444$	21.83	26.99	23.12	25.33	27.76	21.66	26.79	21.58	27.64	23.21	26.62	27.8	21.64	27.71	0.0

Figure 10: Cifar10 Network - Lin1 - Cosine Similarity

TU Bibliothek Your knowledge hub WIEN Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

.3.1 Cifar100 - Similarity Measurements

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	6.411	10.964	5.252	3.774	4.482	3.838	3.479	4.301	3.291	3.999	4.703	4.022
444442 $lr444444$	6.41	0.0	10.243	6.315	7.596	8.191	7.338	7.5	8.127	7.204	7.752	8.347	7.554
444421 $lr444444$	10.962	10.243	0.0	10.253	11.832	12.329	11.588	11.798	12.322	11.6	11.962	12.465	11.82
A98765 $lr444444$	5.241	6.301	10.238	0.0	6.2	6.808	5.987	6.097	6.761	5.925	6.413	7.023	6.341
444442	3.756	7.568	11.804	6.193	0.0	2.451	2.855	2.047	2.252	2.448	2.845	2.951	3.362
444421	4.478	8.188	12.326	6.819	2.46	0.0	3.276	2.344	1.64	2.953	3.059	2.516	3.737
A98765	3.83	7.327	11.576	5.991	2.861	3.273	0.0	2.905	3.248	2.975	3.513	3.768	3.763
444444 $lr444442$	3.468	7.483	11.781	6.097	2.051	2.339	2.902	0.0	1.927	1.978	2.415	2.716	3.056
444444 $lr444421$	4.294	8.118	12.313	6.767	2.259	1.638	3.248	1.93	0.0	2.556	2.757	2.043	3.443
444444 $lrA98765$	3.272	7.171	11.567	5.914	2.445	2.938	2.963	1.971	2.544	0.0	2.798	3.186	2.815
444442 rdo	3.978	7.72	11.931	6.403	2.844	3.046	3.503	2.412	2.75	2.799	0.0	3.346	3.639
444421 rdo	4.67	8.296	12.415	7.001	2.945	2.506	3.751	2.707	2.038	3.18	3.342	0.0	3.921
A98765 rdo	4.012	7.538	11.804	6.341	3.368	3.73	3.759	3.056	3.44	2.821	3.645	3.936	0.0

Figure 11: Cifar100 Network - Conv7 - Distance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	5.589	9.017	6.283	5.415	5.908	5.477	4.812	5.2	4.447	4.796	5.859	5.293
444442 $lr444444$	5.588	0.0	8.423	6.365	6.738	6.405	6.285	6.497	5.953	5.715	6.492	6.543	6.386
444421 $lr444444$	9.015	8.423	0.0	9.024	9.681	9.607	9.417	9.551	9.351	9.144	9.56	9.73	9.594
A98765 $lr444444$	6.267	6.35	9.008	0.0	7.224	7.496	7.055	6.993	7.124	6.716	6.977	7.62	7.286
444442	5.387	6.707	9.649	7.213	0.0	5.6	5.217	4.672	4.995	4.641	4.689	5.689	5.463
444421	5.903	6.401	9.603	7.51	5.624	0.0	4.915	5.62	3.741	4.788	5.609	4.611	5.527
A98765	5.465	6.275	9.405	7.06	5.232	4.909	0.0	5.337	4.69	4.702	5.333	5.419	5.485
444444 $lr444442$	4.796	6.479	9.532	6.993	4.682	5.607	5.332	0.0	4.643	3.959	3.533	5.375	4.929
444444 $lr444421$	5.19	5.944	9.342	7.131	5.012	3.737	4.692	4.649	0.0	3.632	4.639	3.262	4.594
444444 $lrA98765$	4.418	5.682	9.109	6.701	4.636	4.761	4.683	3.947	3.613	0.0	3.971	4.537	3.853
444442 rdo	4.761	6.452	9.52	6.962	4.686	5.576	5.314	3.524	4.619	3.973	0.0	5.358	4.937
444421 rdo	5.814	6.498	9.682	7.593	5.675	4.584	5.391	5.352	3.248	4.528	5.347	0.0	5.314
A98765 rdo	5.277	6.368	9.575	7.286	5.473	5.514	5.48	4.929	4.587	3.864	4.949	5.339	0.0

Figure 12: Cifar100 Network - Conv8 - Distance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	6.066	6.951	6.377	6.2	8.799	6.596	5.493	7.516	5.236	5.46	7.527	5.86
444442 $lr444444$	6.064	0.0	6.01	6.233	7.452	9.6	7.335	7.153	8.784	6.578	7.134	8.798	7.067
444421 $lr444444$	6.949	6.01	0.0	6.918	7.906	9.913	7.712	7.711	9.303	7.122	7.711	9.314	7.588
A98765 $lr444444$	6.356	6.216	6.901	0.0	7.504	9.565	7.502	7.237	8.778	6.948	7.214	8.792	7.401
444442	6.162	7.412	7.866	7.49	0.0	8.007	6.295	5.476	6.885	5.484	5.464	6.901	6.103
444421	8.79	9.594	9.907	9.587	8.044	0.0	7.894	8.155	7.552	8.074	8.094	7.523	8.425
A98765	6.58	7.32	7.698	7.51	6.316	7.883	0.0	6.452	7.416	6.078	6.434	7.429	6.597
444444 $lr444442$	5.472	7.13	7.688	7.237	5.489	8.135	6.445	0.0	6.357	4.647	4.138	6.374	5.41
444444 $lr444421$	7.498	8.768	9.287	8.792	6.913	7.544	7.418	6.367	0.0	6.152	6.314	3.829	6.675
444444 $lrA98765$	5.196	6.532	7.075	6.926	5.477	8.026	6.049	4.631	6.117	0.0	4.641	6.18	4.259
444442 rdo	5.415	7.085	7.661	7.195	5.459	8.047	6.407	4.125	6.282	4.643	0.0	6.312	5.387
444421 rdo	7.45	8.715	9.232	8.742	6.874	7.458	7.374	6.333	3.795	6.163	6.292	0.0	6.661
A98765 rdo	5.837	7.042	7.563	7.401	6.117	8.403	6.589	5.41	6.663	4.276	5.404	6.704	0.0

Figure 13: Cifar100 Network - Conv9 - Distance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	0.85	1.06	0.853	0.883	1.447	1.058	0.771	1.136	0.776	0.764	1.144	0.94
444442 $lr444444$	0.849	0.0	0.851	0.813	1.108	1.622	1.201	1.062	1.402	1.036	1.058	1.398	1.166
444421 $lr444444$	1.06	0.851	0.0	0.931	1.266	1.763	1.348	1.246	1.586	1.234	1.239	1.586	1.344
A98765 $lr444444$	0.848	0.81	0.928	0.0	1.064	1.558	1.174	1.02	1.35	1.036	1.014	1.35	1.161
444442	0.876	1.1	1.256	1.061	0.0	1.294	1.002	0.777	1.034	0.814	0.769	1.043	0.977
444421	1.445	1.621	1.762	1.564	1.301	0.0	1.325	1.326	1.195	1.334	1.308	1.192	1.428
A98765	1.055	1.198	1.345	1.176	1.007	1.323	0.0	1.036	1.186	1.005	1.029	1.191	1.133
444444 $lr444442$	0.768	1.057	1.241	1.02	0.78	1.322	1.034	0.0	0.935	0.665	0.559	0.942	0.857
444444 $lr444421$	1.133	1.399	1.582	1.353	1.039	1.193	1.187	0.937	0.0	0.913	0.929	0.531	1.054
444444 $lrA98765$	0.767	1.025	1.221	1.031	0.812	1.324	0.998	0.662	0.907	0.0	0.667	0.92	0.715
444442 rdo	0.756	1.049	1.227	1.01	0.768	1.299	1.023	0.556	0.923	0.667	0.0	0.933	0.856
444421 rdo	1.129	1.382	1.565	1.34	1.038	1.179	1.18	0.934	0.524	0.918	0.929	0.0	1.051
A98765 rdo	0.935	1.161	1.337	1.161	0.98	1.423	1.131	0.857	1.052	0.718	0.859	1.059	0.0

Figure 14: Cifar100 Network - Lin1 - Distance

	444444	444442 lr444444	444421 lr444444	A98765 lr444444	444442	444421	A98765	444444 lr444442	444444 lr444421	444444 lrA98765	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	1.128	1.495	1.129	1.117	1.805	1.542	0.976	1.47	1.134	0.973	1.474	1.198
444442 lr444444	1.127	0.0	1.198	1.027	1.458	2.078	1.796	1.4	1.853	1.549	1.397	1.849	1.597
444421 lr444444	1.494	1.198	0.0	1.176	1.738	2.343	2.095	1.728	2.182	1.931	1.719	2.184	1.971
A98765 lr444444	1.12	1.02	1.17	0.0	1.381	1.997	1.752	1.34	1.802	1.567	1.341	1.803	1.605
444442	1.106	1.445	1.723	1.376	0.0	1.595	1.391	0.975	1.318	1.124	0.969	1.329	1.194
444421	1.802	2.076	2.341	2.005	1.606	0.0	1.697	1.632	1.467	1.67	1.615	1.46	1.698
A98765	1.535	1.79	2.088	1.756	1.399	1.694	0.0	1.455	1.581	1.456	1.45	1.587	1.496
444444 lr444442	0.969	1.392	1.719	1.34	0.979	1.626	1.452	0.0	1.184	0.929	0.699	1.188	1.013
444444 lr444421	1.464	1.847	2.176	1.807	1.326	1.464	1.582	1.188	0.0	1.15	1.182	0.637	1.207
444444 lrA98765	1.117	1.531	1.908	1.557	1.121	1.657	1.446	0.923	1.14	0.0	0.931	1.156	0.761
444442 rdo	0.958	1.382	1.701	1.334	0.967	1.602	1.44	0.694	1.172	0.932	0.0	1.183	1.014
444421 rdo	1.45	1.824	2.153	1.785	1.32	1.443	1.57	1.175	0.626	1.151	1.177	0.0	1.197
A98765 rdo	1.188	1.587	1.959	1.605	1.2	1.692	1.493	1.013	1.203	0.767	1.021	1.209	0.0

Figure 15: Cifar100 Network - Lin2 - Distance

TU Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	1.497	2.059	1.589	1.469	2.406	2.331	1.262	1.972	1.757	1.247	1.975	1.775
444442 $lr444444$	1.496	0.0	1.652	1.379	1.959	2.818	2.747	1.893	2.526	2.4	1.877	2.533	2.409
444421 $lr444444$	2.058	1.652	0.0	1.448	2.379	3.21	3.247	2.38	3.031	3.036	2.363	3.032	3.046
A98765 $lr444444$	1.578	1.371	1.441	0.0	1.932	2.8	2.819	1.906	2.586	2.607	1.893	2.588	2.606
444442	1.456	1.945	2.36	1.926	0.0	2.087	2.04	1.227	1.717	1.688	1.226	1.73	1.708
444421	2.403	2.815	3.206	2.81	2.099	0.0	2.263	2.143	1.853	2.273	2.121	1.833	2.243
A98765	2.322	2.739	3.235	2.825	2.05	2.259	0.0	2.151	2.184	2.12	2.148	2.202	2.108
444444 $lr444442$	1.253	1.884	2.368	1.906	1.232	2.136	2.146	0.0	1.535	1.42	0.847	1.547	1.457
444444 $lr444421$	1.965	2.52	3.023	2.592	1.727	1.85	2.185	1.539	0.0	1.523	1.534	0.735	1.523
444444 $lrA98765$	1.737	2.377	3.003	2.592	1.684	2.259	2.108	1.412	1.513	0.0	1.423	1.535	0.915
444442 rdo	1.231	1.86	2.339	1.884	1.224	2.108	2.135	0.841	1.523	1.424	0.0	1.534	1.457
444421 rdo	1.948	2.502	2.991	2.564	1.72	1.815	2.181	1.533	0.723	1.53	1.527	0.0	1.521
A98765 rdo	1.762	2.395	3.028	2.606	1.715	2.236	2.104	1.457	1.519	0.921	1.466	1.535	0.0

Figure 16: Cifar100 Network - Lin3 - Distance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	44.13	55.34	42.18	42.37	52.76	43.72	36.24	45.2	36.1	46.43	58.43	45.78
444442 $lr444444$	44.13	0.0	51.09	43.05	53.97	61.08	53.01	51.23	56.71	49.81	57.63	65.98	56.0
444421 $lr444444$	55.34	51.09	0.0	51.37	60.9	66.13	60.36	59.36	63.54	59.18	64.2	70.64	63.66
A98765 $lr444444$	42.18	43.05	51.37	0.0	50.42	57.7	49.81	47.55	53.58	47.91	54.89	63.91	54.44
444442	42.37	53.96	60.9	50.42	0.0	45.71	40.49	35.68	39.71	38.09	46.17	55.23	47.23
444421	52.76	61.08	66.13	57.71	45.68	0.0	47.08	46.09	41.28	48.31	53.47	55.95	54.64
A98765	43.72	53.0	60.36	49.82	40.48	47.08	0.0	41.24	44.23	41.29	50.06	57.92	49.52
444444 $lr444442$	36.23	51.22	59.35	47.55	35.7	46.1	41.23	0.0	34.81	30.01	39.23	52.55	41.94
444444 $lr444421$	45.19	56.71	63.53	53.59	39.72	41.29	44.22	34.81	0.0	35.88	45.37	45.64	45.56
444444 $lrA98765$	36.09	49.79	59.18	47.91	38.09	48.32	41.28	30.0	35.86	0.0	42.53	53.26	37.87
444442 rdo	46.44	57.65	64.22	54.91	46.18	53.5	50.09	39.27	45.4	42.52	0.0	58.65	50.73
444421 rdo	58.5	66.02	70.69	63.96	55.28	56.08	57.98	52.62	45.77	53.29	58.7	0.0	58.87
A98765 rdo	45.8	56.01	63.68	54.44	47.22	54.65	49.52	41.94	45.56	37.84	50.69	58.79	0.0

Figure 17: Cifar100 Network - Conv7 - Cosine Similarity

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	49.22	62.23	47.49	44.93	57.14	49.0	40.0	49.24	40.09	39.8	55.61	46.89
444442 $lr444444$	49.22	0.0	57.62	48.7	58.11	65.78	58.67	56.32	61.64	54.96	56.18	65.77	59.14
444421 $lr444444$	62.23	57.62	0.0	58.67	66.84	72.27	67.18	66.02	69.99	65.68	66.04	72.72	68.55
A98765 $lr444444$	47.48	48.7	58.67	0.0	54.64	62.57	55.57	52.96	58.56	53.14	52.8	63.23	57.47
444442	44.91	58.09	66.82	54.64	0.0	50.5	44.75	37.9	43.13	39.84	38.02	51.07	46.81
444421	57.14	65.78	72.27	62.58	50.53	0.0	50.08	51.28	45.45	52.3	51.01	52.57	56.41
A98765	48.99	58.66	67.18	55.58	44.76	50.07	0.0	46.22	48.13	45.82	46.13	54.83	51.35
444444 $lr444442$	39.99	56.31	66.01	52.96	37.91	51.27	46.22	0.0	39.5	33.52	28.56	48.28	42.22
444444 $lr444421$	49.23	61.63	69.99	58.57	43.15	45.44	48.13	39.51	0.0	39.18	39.28	37.17	45.94
444444 $lrA98765$	40.06	54.94	65.66	53.13	39.83	52.27	45.81	33.51	39.15	0.0	33.69	48.27	36.73
444442 rdo	39.76	56.14	66.02	52.79	38.02	50.98	46.12	28.55	39.25	33.69	0.0	48.16	42.36
444421 rdo	55.62	65.77	72.72	63.23	51.09	52.61	54.85	48.3	37.23	48.28	48.17	0.0	53.27
A98765 rdo	46.89	59.14	68.54	57.47	46.8	56.4	51.35	42.22	45.93	36.72	42.36	53.24	0.0

Figure 18: Cifar100 Network - Conv8 - Cosine Similarity

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	49.55	61.83	48.53	45.98	57.53	50.46	40.9	50.21	41.17	40.62	50.17	44.94
444442 $lr444444$	49.54	0.0	56.88	49.22	58.38	65.67	59.17	56.48	61.9	55.21	56.28	61.86	57.45
444421 $lr444444$	61.83	56.88	0.0	58.56	66.69	71.86	67.35	65.74	69.94	65.45	65.69	69.83	67.07
A98765 $lr444444$	48.52	49.21	58.55	0.0	55.42	62.8	56.71	53.81	59.34	54.14	53.64	59.32	56.41
444442	45.95	58.35	66.67	55.41	0.0	50.73	46.11	39.37	44.84	40.97	39.29	44.85	44.97
444421	57.52	65.67	71.86	62.82	50.76	0.0	50.16	51.79	44.96	52.23	51.33	44.72	54.33
A98765	50.45	59.16	67.34	56.72	46.12	50.15	0.0	47.66	49.05	46.97	47.52	49.03	49.91
444444 $lr444442$	40.88	56.47	65.73	53.81	39.39	51.77	47.65	0.0	41.13	34.74	29.71	41.15	39.95
444444 $lr444421$	50.19	61.89	69.93	59.35	44.86	44.95	49.06	41.14	0.0	39.9	40.76	23.0	43.69
444444 $lrA98765$	41.13	55.18	65.43	54.13	40.97	52.19	46.95	34.72	39.87	0.0	34.81	40.22	32.86
444442 rdo	40.57	56.25	65.66	53.62	39.29	51.29	47.5	29.7	40.73	34.82	0.0	40.84	39.9
444421 rdo	50.1	61.8	69.79	59.29	44.83	44.67	48.99	41.12	22.96	40.2	40.82	0.0	43.84
A98765 rdo	44.93	57.44	67.06	56.41	44.98	54.31	49.9	39.95	43.68	32.87	39.91	43.87	0.0

Figure 19: Cifar100 Network - Conv9 - Cosine Similarity

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	31.36	40.37	31.18	29.42	40.44	34.97	25.54	32.5	25.94	25.41	32.72	30.57
444442 $lr444444$	31.37	0.0	36.3	31.82	38.23	46.65	40.92	36.46	41.75	35.82	36.48	41.53	39.23
444421 $lr444444$	40.39	36.3	0.0	37.84	44.72	51.68	47.23	43.85	48.31	43.9	43.78	48.24	46.54
A98765 $lr444444$	31.3	31.94	38.0	0.0	36.67	44.72	40.01	35.06	40.33	36.03	34.97	40.28	39.24
444442	29.59	38.46	45.01	36.74	0.0	35.94	32.51	25.42	29.92	26.77	25.2	30.15	31.56
444421	40.47	46.68	51.72	44.61	35.8	0.0	36.54	36.6	31.94	36.85	36.03	31.85	39.73
A98765	35.06	41.03	47.36	39.96	32.4	36.59	0.0	33.4	34.48	32.52	33.24	34.58	36.1
444444 $lr444442$	25.64	36.6	44.05	35.06	25.38	36.67	33.43	0.0	26.56	21.72	18.17	26.72	27.48
444444 $lr444421$	32.56	41.83	48.42	40.27	29.81	31.97	34.47	26.53	0.0	25.66	26.22	14.59	30.28
444444 $lrA98765$	26.13	36.12	44.31	36.19	26.81	37.04	32.67	21.8	25.78	0.0	22.02	26.18	23.09
444442 rdo	25.54	36.73	44.12	35.09	25.23	36.17	33.39	18.22	26.33	22.01	0.0	26.61	27.69
444421 rdo	33.01	41.96	48.78	40.53	30.27	32.06	34.84	26.86	14.64	26.24	26.68	0.0	30.71
A98765 rdo	30.69	39.39	46.74	39.24	31.49	39.82	36.13	27.48	30.32	23.0	27.58	30.51	0.0

Figure 20: Cifar100 Network - Lin1 - Cosine Similarity

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	18.69	24.09	18.56	17.69	25.86	22.66	15.31	20.79	16.5	15.3	20.83	17.42
444442 $lr444444$	18.69	0.0	21.04	18.33	23.29	29.98	26.43	22.13	26.55	22.77	22.15	26.45	23.43
444421 $lr444444$	24.09	21.04	0.0	21.56	26.96	33.12	29.97	26.51	30.52	27.59	26.43	30.5	28.16
A98765 $lr444444$	18.54	18.32	21.56	0.0	21.86	28.42	25.51	20.94	25.43	22.88	21.03	25.4	23.4
444442	17.68	23.31	27.0	21.86	0.0	22.99	20.69	15.35	18.98	16.89	15.28	19.12	17.91
444421	25.87	29.99	33.13	28.4	22.96	0.0	24.18	23.38	20.69	23.9	23.1	20.59	24.28
A98765	22.67	26.45	30.0	25.5	20.67	24.19	0.0	21.55	22.75	21.35	21.47	22.82	21.91
444444 $lr444442$	15.28	22.13	26.52	20.94	15.36	23.39	21.56	0.0	16.83	13.78	10.94	16.85	15.0
444444 $lr444421$	20.79	26.56	30.53	25.43	18.98	20.69	22.75	16.83	0.0	16.5	16.71	9.03	17.34
444444 $lrA98765$	16.48	22.8	27.64	22.9	16.89	23.96	21.4	13.78	16.52	0.0	13.89	16.73	11.31
444442 rdo	15.24	22.16	26.45	21.04	15.28	23.13	21.5	10.92	16.71	13.89	0.0	16.86	15.1
444421 rdo	20.84	26.52	30.58	25.43	19.14	20.64	22.89	16.86	8.98	16.74	16.86	0.0	17.41
A98765 rdo	17.41	23.44	28.18	23.4	17.91	24.3	21.91	15.0	17.34	11.32	15.11	17.4	0.0

Figure 21: Cifar100 Network - Lin2 - Cosine Similarity

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	444444	444442 $lr444444$	444421 $lr444444$	A98765 $lr444444$	444442	444421	A98765	444444 $lr444442$	444444 $lr444421$	444444 $lrA98765$	444442 rdo	444421 rdo	A98765 rdo
444444	0.0	17.91	24.11	17.73	16.94	24.97	22.47	14.32	19.94	15.85	14.18	20.02	16.02
444442 $lr444444$	17.91	0.0	21.08	17.53	22.83	29.48	26.6	21.66	25.93	22.6	21.52	26.07	22.66
444421 $lr444444$	24.11	21.08	0.0	20.95	27.17	33.08	30.86	26.63	30.65	28.4	26.49	30.75	28.48
A98765 $lr444444$	17.74	17.52	20.96	0.0	21.33	27.87	25.44	20.38	24.92	22.92	20.33	25.03	22.84
444442	16.96	22.89	27.24	21.35	0.0	21.71	19.8	14.04	17.59	16.06	14.04	17.76	16.29
444421	24.98	29.49	33.1	27.84	21.66	0.0	22.5	22.22	18.91	22.81	21.97	18.72	22.49
A98765	22.49	26.63	30.89	25.43	19.78	22.52	0.0	21.09	21.78	20.93	21.02	21.97	20.8
444444 $lr444442$	14.31	21.69	26.66	20.38	14.04	22.24	21.09	0.0	15.52	12.96	9.61	15.69	13.38
444444 $lr444421$	19.95	25.95	30.67	24.9	17.57	18.92	21.78	15.52	0.0	15.22	15.42	7.51	15.21
444444 $lrA98765$	15.85	22.67	28.5	22.95	16.07	22.9	20.97	12.97	15.25	0.0	13.03	15.47	9.12
444442 rdo	14.17	21.58	26.57	20.34	14.04	22.03	21.06	9.59	15.43	13.03	0.0	15.6	13.43
444421 rdo	20.08	26.19	30.9	25.09	17.79	18.8	22.03	15.71	7.48	15.49	15.61	0.0	15.4
A98765 rdo	16.02	22.69	28.52	22.84	16.28	22.52	20.8	13.38	15.21	9.13	13.43	15.37	0.0

Figure 22: Cifar100 Network - Lin3 - Cosine Similarity