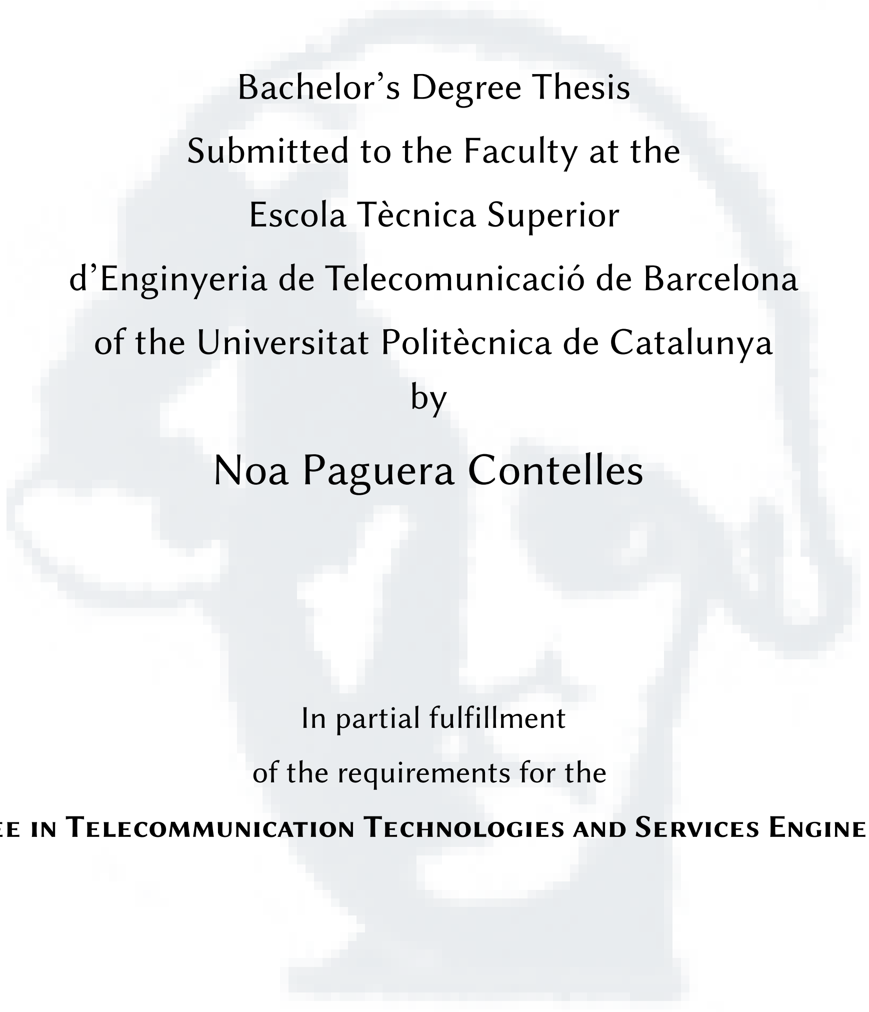
ETSETB – Degree in Telecommunication Technologies and Services Engineering

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

telecos
BCN

AI Distillation for Lightweight Models



Bachelor's Degree Thesis
Submitted to the Faculty at the
Escola Tècnica Superior
d'Enginyeria de Telecomunicació de Barcelona
of the Universitat Politècnica de Catalunya
by

Noa Paguera Contelles

In partial fulfillment
of the requirements for the
Degree in Telecommunication Technologies and Services Engineering

Advisors: Khac-Hoang Ngo
Sergi Bermejo

Linköping, January 2026

Resum

Aquest projecte té com a objectiu millorar l'eficiència i la usabilitat de grans models d'Intel·ligencia Artificial (IA), específicament els models d'aprenentatge profund utilitzats en Visió per Computador quan es despleguen en dispositius perimetrals amb recursos limitats. El nostre objectiu és reduir les demandes computacionals d'aquests models sense sacrificar el rendiment desenvolupant una cadena modular que integra les tècniques de destil·lació de coneixement, l'ajustament de paràmetres amb Low-Rank Adaptation (LoRA) i quantificació. El projecte quantifica els compromisos entre el rendiment i l'eficiència del model en cada etapa de la cadena. Això s'aconsegueix integrant tres metodologies bàsiques per reduir un gran model d'IA (ResNet-18) a una versió compacta i eficient (MobileNetV2).

Resumen

Este proyecto busca mejorar la eficiencia y la usabilidad de grandes modelos de Inteligencia Artificial (IA), en concreto los modelos de aprendizaje profundo utilizados en Visión Artificial al implementarse en dispositivos perimetrales con recursos limitados. Buscamos reducir la demanda computacional de estos modelos sin sacrificar el rendimiento mediante el desarrollo de una canalización modular que integra técnicas de destilación de conocimiento, ajuste de parámetros con Low-Rank Adaptation (LoRA) y cuantificación. El proyecto cuantifica el intercambio entre el rendimiento y la eficiencia del modelo en cada etapa de la canalización. Esto se logra integrando tres metodologías principales para reducir un gran modelo de IA (ResNet-18) a una versión compacta y eficiente (MobileNetV2).

Summary

This project aims to enhance the efficiency and usability of large Artificial Intelligence (AI) models, specifically deep learning models used in Computer Vision when deployed on edge devices with limited resources. We aim to reduce the computational demands of these models with minimal performance degradation by developing a modular pipeline that integrates knowledge distillation, Low-Rank Adaptation (LoRA) fine-tuning, and quantization techniques. The project quantifies the trade-offs between model performance and efficiency at each stage of the pipeline. This is achieved by integrating three core methodologies to shrink a large AI model (ResNet-18) into a compact, efficient version (MobileNetV2).

Acknowledgements

I would like to express my sincere gratitude to Professor Khac-Hoang Ngo, my supervisor, for giving me the amazing opportunity to conduct research at Linköping University and for his continuous guidance, constant availability and support during the months I spent in Linköping working on my bachelor’s thesis.

I want to thank Professor Sergi Bermejo from Universitat Politècnica de Catalunya for his patience and guidance, which helped me throughout the project.

Finally, I would like to express my gratitude for the unwavering support of my parents and friends.

Revision history and approval record

Revision	Date	Author(s)	Description
1.0	22/10/2025	NPC	Document creation
1.1	dd/mm/yyyy	AME, JPV	Error correction
2.0	dd/mm/yyyy	AME, MLO	Revised after review
4.0	dd/mm/yyyy	AME	Final version

DOCUMENT DISTRIBUTION LIST

Role	Surname(s) and Name
Student	Noa Paguera Contelles
Project Supervisor	Sergi Bermejo
Project Supervisor	Khac-Hoang Ngo

Written by:		**Reviewed and approved by:**	
Date	14/01/2026	Date	dd/mm/yyyy
Name	Noa Paguera Contelles	Name	Khac-Hoang Ngo
Position	Project Author	Position	Project Supervisor

Contents

List of Figures

List of Tables

Abbreviations

AI Artificial Intelligence

CNNs Convolutional Neural Networks

ETSETB Escola Tècnica Superior d’Enginyeria de Telecomunicació de Barcelona

FL Federated Learning

GRETST Grau en Enginyeria de Tecnologies i Serveis de Telecomunicació

KD Knowledge Distillation

LLMs Large Language Models

LoRA Low-Rank Adaption

PEFT Parameter-Efficient Fine-Tuning

CHAPTER 1

Introduction

1.1 Motivation

This project aims to enhance the efficiency and usability of large Artificial Intelligence (AI) models, specifically deep learning models used in Computer Vision (CV), when deployed on edge devices with limited resources. The aim is to reduce the computational demands of these models with minimal performance degradation by developing a modular pipeline that integrates knowledge distillation, Low-Rank Adaptation (LoRA) fine-tuning, and quantization techniques. The project systematically quantifies the trade-offs between model performance (accuracy) and efficiency (size, latency, computational cost) at each stage of the pipeline.

Moving AI from the cloud to the device is necessary for several reasons, such as zero latency, privacy and security, and bandwidth efficiency. Applications requiring immediate decisions cannot afford the round-trip delay of sending data to a server.

In practice, simplified models are relevant for personal privacy. In healthcare, wearable devices use compact AI to monitor cardiac rhythms or glucose levels directly on the user's wrist, ensuring that sensitive medical data never leaves the device. This local processing is also important for smart home security, the ability to run AI on "lite" hardware ensures that technology remains functional, private, and energy-efficient in any environment.

Beyond that, the drive toward simpler models is fueled by the need for autonomy and immediacy. In automotive safety, for instance, driver assistance systems must process video feeds to detect pedestrians or obstacles with near-zero latency; waiting for a cloud server to respond is not an option when millisecond delays determine braking distances. By running these models locally, factories avoid the massive bandwidth costs and security risks associated with streaming constant video to external servers.

This is achieved by integrating three core methodologies to shrink a large AI model (ResNet-18) into a compact, efficient version (MobileNetV2):

1. Knowledge Distillation: Train a small AI model, called "student", to mimic a larger,

more complex one, called ”teacher” by transferring the teacher’s knowledge into the student.

2. Low-Rank Adaptation (LoRA) Fine-Tuning: Adjust the small model for specific tasks without significantly increasing its size.

3. Quantization: Simplify how the model handles numbers, cutting memory usage and battery drain.

The project empirically validates this pipeline, providing a clear methodology for developing lightweight, task-specialized AI suitable for edge computing applications. The methodology is grounded in information theory to maximize knowledge transfer.

1.2 Work Goals

The purpose of this project is to address the deployment of large, powerful AI models on resource-constrained edge devices (e.g., smartphones and Internet of Things (IoT) sensors) by designing, implementing, and evaluating a modular, three-stage pipeline. This pipeline systematically combines knowledge distillation, LoRA fine-tuning, and quantization to create a lightweight yet functional version of a standard computer vision model.

Furthermore, the project is grounded in information theory to guide the distillation process, ensuring that the student model learns to compress the input while preserving the maximum information about the teacher’s predictions.

To achieve the main goal of developing and evaluating a compressed student model, the project has the following specific objectives:

1. Hybrid Distillation-LoRA Framework: Develop a two-phase approach where a compact student model is first distilled from a large pre-trained AI model and then fine-tuned using low-rank adaptation to achieve task-specific performance improvements.

   1.1 Develop a mini-model for hardware with limited resources: Create a compact model based on a method to distill knowledge from a large AI into a smaller, lighter model that can run efficiently on T4 Colab.

   1.2 Optimize this Lightweight Version to minimize memory usage for CV by Fine-Tuning with Low-Rank Adaptation: Use techniques like LoRA to further optimize the smaller model for specific tasks, improving its performance without increasing its size significantly.

2. Reduce Computational Costs through Quantization: Apply quantization strategies to shrink the model’s size and lower its energy consumption, making it suitable for real-world applications. In other words, implement quantization-aware distillation methods to reduce model size and computational complexity, ensuring viability for edge deployment.

3. Evaluate the resulting AI model against the original using standard metrics for speed, memory usage, and accuracy:

   3.1 Evaluate Performance: Test the optimized model on practical tasks (e.g., object classification), analyzing how well it balances size, speed, and accuracy.

   3.2 Study Fine-Tuning Dynamics: Investigate how the initial distillation phase impacts the efficiency and effectiveness of subsequent LoRA-based fine-tuning, identifying the most critical elements of the original large AI model knowledge for downstream performance.

   The evaluation is done in parallel with knowledge distillation and fine-tuning.

1.3 Requirements and Specifications

A set of requirements has been defined to ensure that the developed system meets the objectives, covering key points such as the model's capabilities and its evaluation.

Requirements

Requirement	**Description**
Benchmark	The distilled model must retain high accuracy compared to their teacher while being deployable on resource-constrained edge devices.
Efficiency	The lightweight model must demonstrate reduced computational cost and inference time.
Pipeline	The pipeline must integrate knowledge distillation, LoRA fine-tuning and quantization. It must be reproducible and modular for future extensions.
Evaluation	The resulting model must be evaluated on standard CV benchmarks and framed within an information-theoretic perspective.

Table 1.1: *Requirements table.*

Specifications

Specification	Description
Accuracy	The student model must achieve 90% accuracy of the teacher model.
Model size	The model must reduce >75% of the original size.
Time optimization	The model must reduce inference time.
Development platform	The developed system must be compatible with major operating systems such as Linux, macOS, Windows.

Table 1.2: *Specifications table.*

Model Architectures:

The selection of teacher and student models is designed to create a realistic and challenging distillation scenario.

- **Teacher Model: ResNet-18:** This architecture is selected for its known high performance in CV, since it can learn complex, non-linear channel features, establishing a strong upper-bound for accuracy.

- **Student Model: MobileNetV2:** This model is chosen as a prototypical lightweight architecture for edge devices. Its efficiency comes from the use of depthwise separable convolutions, which drastically reduce the parameter count and computational load compared to standard convolutions. The significant architectural differences between ResNet-18 and MobileNetV2 make them excellent test cases for evaluating the effectiveness of different distillation techniques.

1.4 Methods and Procedures

The project builds upon existing state-of-the-art technologies, architectures and methods developed by other authors:

- Teacher model: ResNet-18 (He et al., 2015)

- Student model: MobileNetV2 (Sandler et al., 2018)

- Distillation methods: based on Hinton et al. (2015)

- LoRA fine-tuning: Low-rank adaptation methods (Hu et al., 2021)

The project is conducted independently and does not form part of any department or company research or development project. The UPC supervisor provided the foundational concepts and initial ideas, which were then complemented by the supervisor at Linköping University.

1.5 Work Plan

The project is structured into three overlapping phases, ensuring a clear progression from foundational development to advanced analysis.

Distillation-LoRA Framework & Initial Evaluation. This phase focuses on implementing the core compression pipeline. The key objective is a functional, fine-tuned student model benchmarked against the original teacher and baseline student.

Quantization & Continued Evaluation. This phase builds on Phase 1 by applying the final compression stage, quantization. The objective is the final, fully compressed model, and a complete set of experimental results.

Theoretical Analysis, Final Report, and Submission. This final phase is dedicated to the theoretical framing of the project, in-depth analysis of the results, and writing the thesis document.

Work Packages

- WP1: Distillation-LoRA Framework & Evaluation
  - Tasks:
    - T1.1: Comprehensive literature review on KD, LoRA, and quantization.
    - T1.2: Setup of development environment (PyTorch, Colab).
    - T1.3: Implementation of data loading and preprocessing for CIFAR-100.
    - T1.4: Train and benchmark the ResNet-18 teacher and baseline MobileNetV2 student.
    - T1.5: Implement the knowledge distillation framework and train the distilled student model.
    - T1.6: Implement LoRA fine-tuning on the distilled student model.
    - T1.7: Benchmark all intermediate models (teacher, baseline, distilled, LoRA-tuned).
- WP2: Quantization & Theoretical Framing
  - Tasks:
    - T2.1: Apply post-training INT8 quantization to the LoRA-tuned model.
    - T2.2: Design and execute a final, comprehensive benchmarking script to evaluate all model versions on accuracy, size, latency, and FLOPs.
    - T2.3: Generate tables and visualizations for the results chapter.
- WP3: Project Management & Thesis Writing
  - Tasks:

    * T3.1: Draft Introduction and State of the Art chapters.
    * T3.2: Draft Methodology chapter as WP1 and WP2 tasks are completed.
    * T3.3: Draft Results, Sustainability Analysis, and Conclusion chapters.
    * T3.4: Final review, formatting, and thesis submission.
  - Primary Deliverables:
    * Proposal Presentation and TFG Work Plan. deadline: 05/10/2025
    * Critical Review. deadline: 30/11/2025
    * Final Bachelor’s Thesis document. deadline: 18/01/2026

Below is the Gantt diagram with the timeline of the project.

CHAPTER 2

State of the Art of the Technology Applied in this Thesis

In order to achieve better performance, current deep learning models generally are becoming deeper and wider. However, these heavy models are hard to deploy on resource-constrained devices in practice due to computational and memory resource limitations. We aim to develop an AI model that is smaller, faster, and more energy-efficient while maintaining its prediction capability. In the following sections, we follow the development of deep learning, making a survey and integrating three core methodologies to shrink a large AI model (ResNet-18) [7] into a compact, efficient version (MobileNetV2) [17]: Knowledge Distillation [8], LoRA Fine-Tuning [9] and Quantization.

This project empirically validates this pipeline, providing a clear methodology for developing lightweight, task-specialized AI suitable for edge computing applications. The methodology is grounded in information theory to maximize knowledge transfer.

2.1 Deep Learning

Deep learning is a subset of machine learning that utilizes multilayered neural networks to simulate decision-making of the human brain to perform tasks such as classification, regression and feature learning. Deep neural networks consist of multiple layers of interconnected nodes, a combination of data inputs, weights and bias, which can be expressed as $W^T \cdot X + b$ in linear regression, where W is the weight matrix, X is the input, and b is the bias.

On the one hand, over the past decade, deep learning has revolutionized computer vision in tasks such as image classification, object recognition and image segmentation. Architectures such as ResNet have set new records in benchmark performance. However, these deep learning models are computationally intensive and require large amounts of memory and processing power, which limits their deployment to high-performance servers with dedicated GPUs or TPUs.

On the other hand, edge devices such as smartphones, IoT sensors, drones, and medical wearables, have resource constraints, including limited processing power, memory, and energy. Despite these constraints, edge computing is becoming increasingly important because it reduces latency, lowers dependence on centralized cloud services, and improves privacy by keeping sensitive data local. A key challenge in this field is finding a way to leverage the power of deep learning models while addressing the limitations of edge devices.

2.2 Model Compression

The deployment of advanced AI models on edge devices with limited resources has motivated substantial research into model compression techniques. The central goal of this field is to balance the power of large models with the requirements of efficiency, latency, and privacy. This field has evolved from applying individual optimization techniques in isolation to designing integrated, multi-stage pipelines that synergistically combine methods for maximum efficiency.

There are several ways to make deep learning models more efficient without sacrificing accuracy with the following core components of this project: Knowledge Distillation (KD) shrinks the knowledge base [8], Parameter-Efficient Fine-Tuning (PEFT) [25] with a specific focus on Low-Rank Adaptation (LoRA) fine-tunes for specific tasks [9], and Quantization further compresses the model for deployment.

In this integration, we combine multiple AI model compression techniques in sequence for developing methodologies; instead of applying these methods in isolation, we view them as interconnected steps in a single, integrated process, in which the order and interaction of these techniques are considered crucial for maximizing efficiency and achieving the best compression results, as exemplified by approaches that combine distillation, LoRA, and pruning in a multi-stage process.

Moslem's recent work on speech translation combines knowledge distillation with QLoRA and iterative pruning in a multi-stage process [15]. This highlights the importance of technique order, as distillation can create a more robust student model, aiding subsequent quantization or fine-tuning.

2.3 Knowledge Distillation

Knowledge Distillation is a machine learning technique that aims to transfer the learnings of a large pre-trained model to a smaller and more compact one so that the "student" mimics the behaviour of the "teacher" [8] by matching its predictions. It is usually applied to deep neural networks with many layers and learnable parameters. The main idea is to use soft probabilities of the larger network to supervise the smaller one, which reveal more information than the class labels alone. Soft targets can be estimated by a softmax function as:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

where z are the logits and T is the temperature parameter, a scaling factor applied to logits before softmax.

The loss for the student is then a linear combination of cross entropy loss L_{CE} and knowledge distillation loss L_{KD}:

$$L = (1-\alpha)\, L_{\mathrm{CE}}(y, p_{\mathrm{student}}) + \alpha\, L_{\mathrm{KD}}(p_{\mathrm{teacher}}, p_{\mathrm{student}})$$

where α is a hyperparameter that balances the importance of the teacher's logits versus the hard labels.

We have:

$$L_{CE} = -\sum_{k=1}^{K} y_k \log(p_k)$$

where y_k is the true probability of class k and p_k is the predicted probability of class k.

[vanilla knowledge distillation loss function]

Knowledge Distillation techniques have been employed across multiple fields, including speech recognition, natural language processing, image recognition [7] and object detection. KD is an effective means of transferring the capabilities from a large model or set of models to a single smaller model.

AI model's accuracy and capacity are not enough to make the model useful - it has a limit of time, memory and computational resources. While top performing models are often too large for most applications, small models are faster yet lack the accuracy and knowledge capacity of the first.

A critical analysis of seminal and recent works reveals a clear trajectory in the field. Initially, techniques were developed to solve singular problems, e.g. KD for knowledge transfer and LoRA for tuning efficiency.

Hinton, Vinyals, & Dean focused exclusively on the concept of transferring "knowledge" from a teacher to a student model, establishing the theoretical groundwork for response-based distillation [8]. Although their work showed that a student model could learn a richer structure by training on the teacher's outputs, they did not address the efficiency of adapting the distilled model to new tasks.

2.4 Low-Rank Adaptation

LoRA was originally developed for efficient fine-tuning in large language models (LLMs). LoRA offers an opportunity to fine-tune compact models without enlarging their size.

Hu et al. developed LoRA to address the computational infeasibility of fine-tuning large models, reducing trainable parameters, memory usage and training time by freezing pre-trained weights and training only small, low-rank adapter matrices inserted into existing weight structures, without increasing inference latency [9]. The pre-trained model's original weight matrix is frozen and only the smaller matrices are updated during training. Yet, the LoRA paper did not focus on reducing the size of the base model itself, which remained a significant barrier for edge deployment.

Instead of retraining the whole model, LoRA freezes the original weights and parameters of the model. On top of this original model, it adds a lightweight addition called a low-rank matrix, which is then applied to new inputs to get results specific to the context. The low-rank matrix adjusts for the weights of the original model so that outputs match the desired use case.

Figure 2.1 shows how the additional matrices A and B are updated by using smaller matrices of rank r. Once LoRA training is complete, smaller weights are merged into a new weight matrix, without modifying the original weights W of the pre-trained model and not needing to store a complete copy of the model. In the training process after calculating the loss, the loss is only backpropagated to the LoRA matrices.

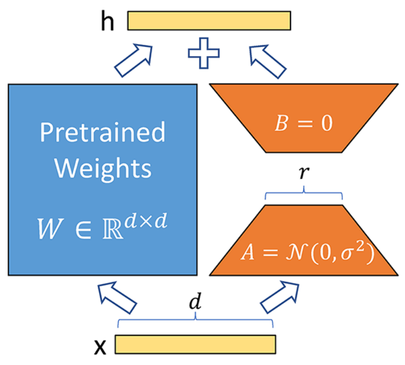

h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 2.1: *Reparametrization. Illustration from Edward J. Hu's work [9]*

Full fine-tuning requires storing optimizer states for all parameters. Since LoRA freezes most of the model's weights, GPU memory usage is lower than full fine-tuning storage efficiency training speed.

2.5 Quantization

The limitations of these isolated approaches naturally led to hybrid solutions such as QLoRA [2], which combines 4-bit quantization and LoRA adapters, enabling the fine-tuning of large models on a single consumer GPU. This unified approach addresses memory issues in both base models and fine-tuning, shifting towards combined compression solutions.

Quantization reduces the precision of model weights and activations (e.g., from 32-bit floating-point to 8-bit integers), which reduces memory and speeds up inference time.

Post-training quantization (PTQ) applies quantization to a fully trained model without additional training. It typically relies on a small calibration dataset to estimate activation ranges and is widely used due to its simplicity and low computational cost. While PTQ can introduce accuracy degradation it is often sufficient for inference-only deployments where retraining is impractical.

In contrast, quantization-aware training (QAT) applies quantization during training, allowing the model to adapt to reduced precision and mitigating accuracy degradation. This approach generally achieves higher accuracy than PTQ but requires access to training data and increased training complexity.

While multi-stage pipelines are emerging, particularly in natural language processing, a systematic, empirical study of a sequential KD, LoRA and Quantization pipeline for computer vision remains a clear research gap.

CHAPTER 3

Methodology / Project Development

This chapter describes the methodology employed to design, train and evaluate the knowledge distillation framework proposed in this study. It outlines the dataset, data preparation, teacher and student model architectures, training procedures, evaluation metrics and experimental setup.

This research follows an experimental design where a teacher model is used to supervise the training of a smaller student model. A series of experiments evaluate the impact the distillation process in the task of image classification.

Image classification serves as a fundamental block of computer vision, aiming to categorize an input image into a single, predefined class. While humans perform this task intuitively, it presents a significant computational challenge, where the machine must interpret raw pixel intensity values as objects. Modern approaches leverage Deep Learning architectures, such as Convolutional Neural Networks (CNNs), to automatically extract features from data. By learning patterns of increasing complexity, these models can generalize across variations in viewpoint and lighting, enabling applications ranging from medical diagnostics to autonomous navigation.

3.1 Data Preparation and Preprocessing

For the initial evaluations of the model, we conducted experiments to evaluate RestNet18 and MobileNetV2 performance using the public available CIFAR-10 and CIFAR-100 datasets. The CIFAR-10 and CIFAR-100 datasets are labeled subsets of the 80 million tiny images dataset. This dataset was chosen because it is an established Computer Vision dataset used for object recognition and it allows to quickly try different algorithms, ideal for benchmarking and preliminary evaluations.

Data preparation is a crucial step that transforms the input data into the format required by the model, ensuring it can be correctly processed and learned from. The dataset is divided in training, validation and test.

The images are normalized to ensure consistency in the data format. Data normalization prevents large values from dominating the learning process by ensuring numerical features are on a similar scale for optimal model performance. It avoids numerical instability, speeds up convergence in gradient-based algorithms and ensures features contribute equally.

3.2 Model Training and Fine-Tuning

Training a model involves teaching it to identify patterns within data so that it can make accurate predictions or decisions when presented with new, unseen inputs. The training process is generally divided into two key stages: pre-training and fine-tuning.

- Pre-training refers to training a model on a large, general-purpose dataset, often using unsupervised or self-supervised learning methods. During this stage, the model develops a foundational understanding of the data by learning to extract features and recognize broad patterns, without focusing on any specific task. The resulting pre-trained models serve as a strong starting point for various downstream applications, significantly reducing both time and computational cost compared to training a model entirely from scratch.
- Fine-tuning, on the other hand, builds upon a pre-trained model by adapting it to a specific task or dataset. This is done by further training the model on a smaller, labeled, and task-specific dataset. Through fine-tuning, the model refines its learned representations and becomes specialized in the target application while still benefiting from the general knowledge acquired during pre-training.

In this work, the ResNet-18 model serves as the teacher network due to its strong performance on image classification tasks. We focus on fine-tuning the ResNet-18 model rather than training it from scratch. This approach is supported by the availability of high-quality pre-trained checkpoints in PyTorch, which have been trained on large-scale datasets. These pre-trained models already capture fundamental visual features and general data representations. However, their learned knowledge remains shallow and tied to the original training data. Fine-tuning enables the model to adapt and develop more task-specific capabilities. In our case, the teacher was pre-trained on ImageNet and fine-tuned on the target CIFAR dataset.

By leveraging the visual representations obtained during pre-training, this method avoids the significant computational expense and environmental impact associated with large-scale model training. Consequently, it aligns with sustainable AI practices. In our implementation, we fine-tune the ResNet-18 model using labeled data to tailor it to the desired task.

In the distillation process, the student model is a MobileNetV2, selected for its compact size and suitability for edge deployment. The student model is trained for 50 epochs using Adam optimizer with learning rate of 0.001. All hyperparameters are tuned based on the validation set. After several runs, the distillation temperature and alpha values chosen are T=4 and $\alpha = 0.3$. It is trained under the supervision of the teacher, using both ground truth labels and soft targets learned from the larger model.

Model performance is evaluated using top-1 accuracy, F1-score, and computational metrics such as inference latency.

To select the best checkpoint, we monitor validation accuracy and choose the checkpoint that achieves the highest score. This helps ensure that the model generalizes well to unseen data and avoids overfitting to the training set. Although fine-tuning continues for additional steps, this process ensures that training stops at an optimal point.

3.3 Quantization

Model quantization converts high-precision floating-point values into lower-precision formats such as integers, resulting in faster inference, lower energy consumption, and reduced storage requirements. In particular, model parameters (weights and activations) are approximated using INT8 instead of the more common 32-bit floating-point format (FP32). This is especially beneficial in tasks such as image processing and neural network inference, where computational efficiency and memory optimization are crucial.

The primary objective was to reduce the computational complexity and memory footprint of the MobileNetV2 model. Post-Training Quantization (PTQ), specifically dynamic quantization, is selected as the primary optimization strategy. This approach was chosen because it allows for reductions in model size without requiring the intensive retraining cycles associated with Quantization-Aware Training (QAT).

Linear quantization is applied to the model weights. The mapping from the floating-point tensors to the quantized integer tensors is governed by the affine transformation:

$$q = round(\lfloor r/S) \rfloor + Z$$

where S is the scale and Z the zero-point, adjusted to ensure that the floating-point value of 0 is exactly represented in the integer space.

The quantization process was executed using the PyTorch torch.quantization backend. The workflow followed a three-step pipeline of model preparation, layer-specific targeting and dynamic conversion. The pre-trained model was loaded and set to evaluation mode (.eval()). The nn.Linear modules were targeted (Fully Connected layers), which account for the vast majority of the model's parameters and memory consumption. Using quantize_dynamic, weights were converted to 8-bit integers (qint8). Activations were kept in floating-point format while at rest and quantized dynamically during the forward pass to minimize the precision loss typically seen in static quantization.

To validate the effectiveness of the quantization, the model size is measured in Megabytes (MB) on disk and inference latency is measured in milliseconds (ms).

3.4 Benchmark

Evaluation is a fundamental component of any research project, as it provides a systematic and objective means of measuring the effectiveness and robustness of a proposed system. A well-designed evaluation framework allows researchers to compare models under consistent conditions, quantify performance using standardized metrics, and identify the strengths and limitations of different approaches.

The quality of the evaluation process depends heavily on the datasets used. These datasets must be both trustworthy and sufficiently diverse to reflect real-world scenarios. Reliable benchmark data ensure that the results are reproducible and not influenced by noise or bias, while variation within the dataset allows for a more comprehensive assessment of the model's ability to generalize beyond the specific examples seen during training.

3.5 Evaluation Framework and Testing

The evaluation framework used in this project is structured to assess the performance and reliability of the MobileNetV2 student model against the teacher. The framework enables consistent comparison of model behavior under different experimental parameters. This structure not only helps verify the model's robustness but also ensures that its performance can be generalized beyond the specific conditions encountered during training.

To conduct evaluation, the model is tested on a dataset that is isolated from the training process, therefore ensuring an unbiased assessment. The testing procedure examines the model's ability to correctly classify inputs and handle variations in data distribution.

Following the evaluation, accuracy is computed as the primary performance metric. Accuracy provides a clear measure of the proportion of correct predictions relative to the total number of samples, making it an appropriate and widely accepted indicator for classification tasks.

However, accuracy is often insufficient for evaluating models trained on imbalanced datasets, where one class substantially outweighs the others. In such cases, a model may achieve deceptively high accuracy simply by consistently predicting the majority class, while failing to detect the minority class altogether. This makes accuracy an unreliable indicator of true model performance.

To address these limitations, additional metrics such as precision, recall, and the F1 score are employed. These metrics provide more informative insights, particularly regarding the model's ability to recognize minority classes and to manage trade-offs between false positives and false negatives.

A confusion matrix is a fundamental diagnostic tool used to evaluate the performance of a classification model. It provides a structured layout of the counts of true versus predicted values.

For binary classification tasks, the matrix is organized into four distinct quadrants: True Positive (TP), where the model correctly predicts the right class; True Negative (TN), where the model correctly predicts the negative class; False Positive (FP), where the model incorrectly predicts a positive outcome; and False Negative (FN), where the model incorrectly predicts a negative outcome.

		Ground truth	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Table 3.1: *Confusion matrix for binary classification*

While the matrix itself provides raw counts, the overall performance is often summarized via Accuracy. This metric represents the ratio of correct predictions to the total number of observations. It is expressed on a scale from 0 to 1, where a value of 1 represents a perfect classification. In practice, a robust model is characterized by high values along the diagonal (TP and TN) and minimal values elsewhere.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive. High precision indicates that the model's positive predictions are reliable.

$$Precision = \frac{TP}{TP + FP}$$

Recall, also known as sensitivity or the true positive rate, quantifies the proportion of actual positive instances that the model successfully identifies. Recall is crucial in scenarios where failing to detect positive cases, producing false negatives, is particularly costly.

$$Recall = \frac{TP}{TP + FN}$$

Precision and recall frequently exhibit a trade-off. Improving one may reduce the other. Therefore, selecting the appropriate balance depends on the specific requirements and risks associated with the task.

The F1 score provides a single metric that balances both precision and recall, making it particularly useful when false positives and false negatives are equally important. It is defined as the harmonic mean of precision and recall:

$$F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

The F1 score ranges from 0 to 1, with higher values indicating stronger overall performance. Achieving an F1 score of 1 would require perfect precision and recall simultaneously, which is a condition rarely met in practice due to the trade-offs between detecting all positive instances and minimizing incorrect positive predictions.

CHAPTER 4

Results

4.1 Experiments and Tests

This chapter provides an overview of the experimental setup used for training and evaluating the model. It specifically addresses the configurations implemented to achieve the results discussed. For further methodological details or prerequisites, please refer to Chapter 3. In this section, the parameters employed for fine-tuning are defined.

Fine-tuning Configuration

Fine-tuning of the MobileNet model is performed using a custom configuration that defines the parameters used in training, optimization, and overall model behavior. This setup ensures the model can adapt to the specific dataset used in our experiments. The configuration remains consistent across all trained models to maintain uniformity.

The model uses a 32-bit precision format (FP32), a storage format for floating-point values suitable for applications where higher precision is not critical. This is particularly advantageous in scenarios like image processing and neural network training, where computational efficiency and memory optimization are crucial.

Fixed random seed values of 42, 518, 1993 are used to ensure reproducibility, enabling consistent results across multiple runs.

To maintain efficiency during training, only the highest-performing checkpoint is retained. Selection of the best checkpoint is based on the validation loss metric, ensuring that the stored model reflects peak performance for future evaluation.

The model is pre-trained on the ImageNet dataset for image classification. Data preprocessing includes normalization to standardize inputs before they are fed into the model. Image augmentation is also applied, introducing random transformations to increase robustness to data variability.

For optimization, we use the Adam optimizer, a widely adopted choice due to its adaptability to changing gradients during training. The model begins training with an initial learning rate of 0.001, chosen to provide stable updates without causing divergence.

4.2 Data Visualization

Figure 4.1 illustrates the baseline accuracy of the student model. These results serve as a primary benchmark for comparison against the distilled model, which is trained using the soft outputs of the teacher model.

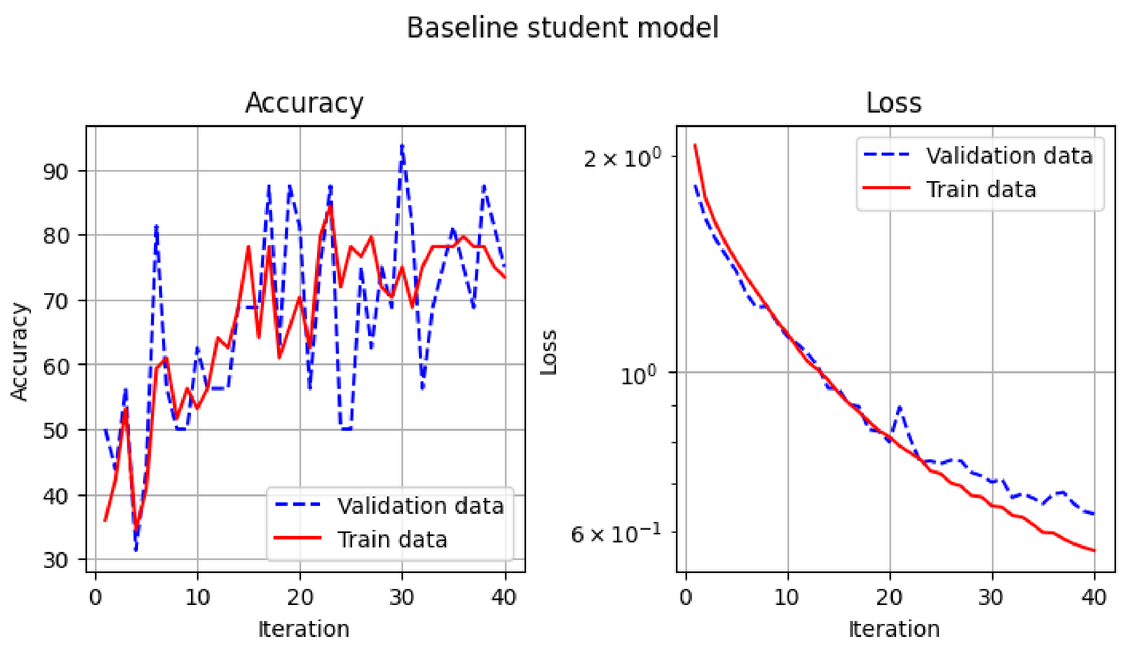


Figure 4.1: *Baseline MobileNetV2 student model*

Knowledge Distillation Results

The implementation of the knowledge distillation phase aimed to transfer the predictions of the ResNet18 to the more compact MobileNetV2. The results in Figure 4.2 indicate that the student model achieved an accuracy of 79.2%, successfully recovering 97% of the teacher's performance. This suggests that the soft targets provided by the teacher were effective in regularizing the student's training, despite the significant reduction in total parameter count.

LoRA & Quantization Trade-offs

To further optimize the model for deployment, LoRA fine-tuning was applied with 8-bit quantization. While quantization typically introduces a quantization error that degrades performance, the LoRA adapters effectively compensated for this loss. The final quantized model reached a compressed size of 79% compared to the baseline, with a minimum accuracy degradation.

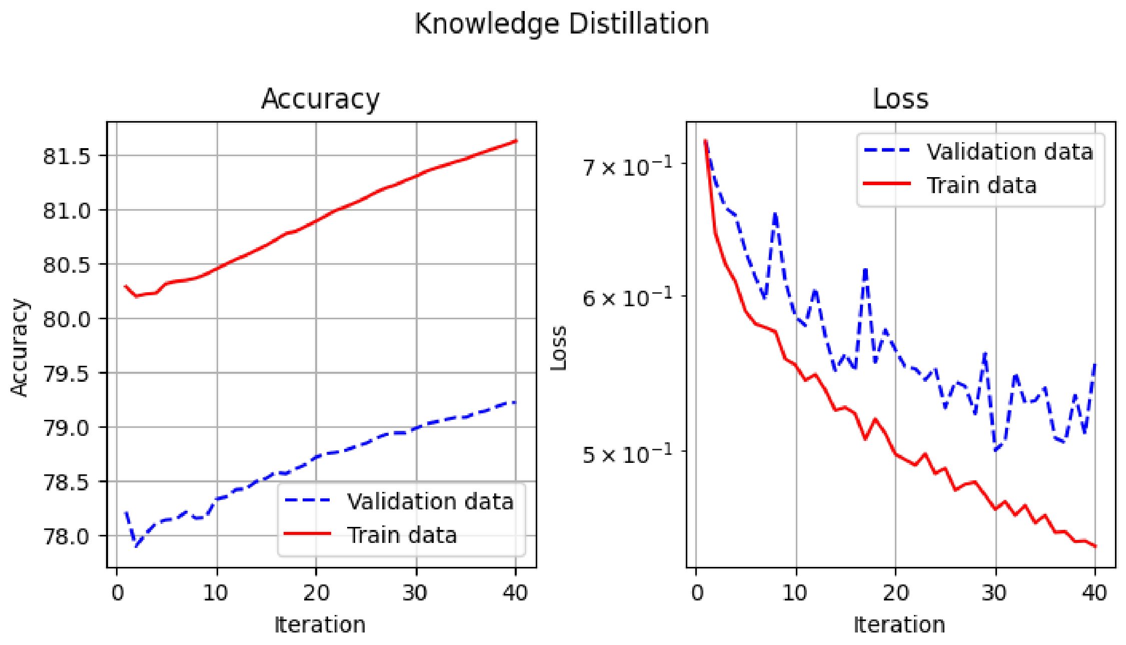


Figure 4.2: *MobileNetV2 student model after distillation*

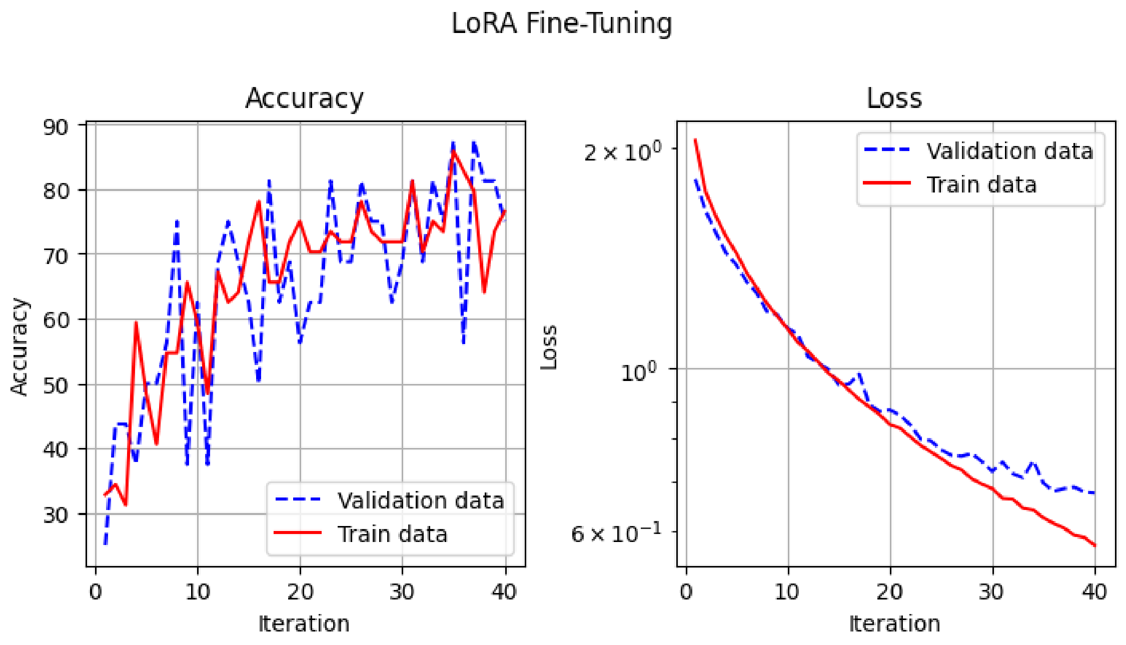


Figure 4.3: *MobileNetV2 student model after LoRA Fine-Tuning*

4.3 Discussion

In summary, this research successfully established a pipeline for model compression. However, several limitations must be acknowledged. First, the reliance on a fixed, balanced dataset, while providing a controlled environment for testing, may mask the model's performance in real-world, imbalanced scenarios.

Secondly, the observed inconsistencies in the LoRA fine-tuning phase suggest a sensitivity to hyperparameter settings that was not fully captured within the current training volume. The inclusion of synthetic data generation could address these variances in future iterations.

Despite these constraints, the results demonstrate that the proposed pipeline achieves >75% student model accuracy, providing a viable framework for further optimization in image classification.

Model stage	Accuracy (%)	Size (MB)	Inference (ms)	FLOPS
Teacher	80.5	44.8	95.1	74.440M
Student without distillation	76.4	9.19	66.2	13.340M
Distilled Student	79.0	9.19	66.2	13.340M
LoRA-tuned Student	79.3	9.22	77.2	13.340M
Quantized Student	79.5	9.18	15.1	13.340M

Table 4.1: *Summary table comparing models' metrics*

CHAPTER 5

Sustainability Analysis and Ethical Implications

The sustainability of technological innovations, particularly those involving artificial intelligence and large-scale data processing, requires careful assessment across environmental, social, economic, and ethical dimensions. This section evaluates the broader impacts of the project and discusses potential societal implications associated with advancing this technology.

The development, training, and deployment of machine learning models require substantial computational workloads, which translate into energy consumption and associated environmental costs. In this project, the processing of visual data during the preparation phase and the subsequent fine-tuning and evaluation of the model contribute to the overall computational demand.

Training machine learning models requires considerable electricity due to the intensive operations involved. The NVIDIA T4 available on Google Colab, used for this project, consumes approximately 70W, emphasizing the importance of efficient resource usage during training.

The T4 GPU was selected because GPUs are optimized for parallel computation, enabling them to conduct multiple operations simultaneously. This significantly reduces training time compared to CPU-only systems and makes GPUs particularly effective for large-scale datasets and complex neural network architectures. While GPUs consume more power than CPUs, the computational efficiency they provide justifies their use in scenarios where model training time and throughput are critical.

Storage requirements also contribute to environmental impact. The project requires approximately [Xbytes] of storage, primarily due to the datasets used in model training. Storage infrastructure has its own energy overhead: maintaining 1 terabyte of data in a data center typically consumes around 100 kWh annually. Consequently, managing [Xbytes] of data would result in an estimated annual energy consumption of [X kWh], excluding the additional energy needed for data processing and cooling. Implementing effective data management and pruning strategies is therefore essential to minimize environmental impact when handling large-scale datasets.

Training AI models contributes to carbon emissions, as the electricity used by computational hardware often originates from carbon-intensive energy sources. Although an exact carbon footprint cannot be calculated without detailed measurements of power usage, hardware efficiency, and local energy mixes, we can estimate the approximate environmental impact associated with the model training process. These estimates help highlight the importance of optimizing training pipelines, adopting energy-efficient hardware, and exploring low-carbon or renewable energy options where available.

CHAPTER 6

Conclusions and Future Work

The proposed pipeline demonstrates that the integration of distillation, low-rank adaptation, and quantization provides a robust framework for model compression. The findings confirm that it is possible to achieve a reduction in memory usage compared to the teacher model while maintaining competitive accuracy levels. This makes the resulting model suitable for resource-constrained environments where the original baseline would be computationally infeasible.

Currently, the compression stages were executed with fixed parameters: a specific rank r for LoRA and a specific bit-width for quantization. Future research could investigate how to automatically find the optimal balance between distillation temperature and quantization thresholds.

A further line of research is to frame the distillation process within an information-theoretic context and explore extending the pipeline to a Federated Learning setting for decentralized, privacy-preserving training.

While this thesis focused on theoretical model size and accuracy, the pipeline could be extended by testing performance on diverse edge devices such as Raspberry Pi or mobile CPUs. This would provide empirical data on ”Inference latency vs. power consumption” that goes beyond static model size.

Bibliography

[1] Jang Hyun Cho and Bharath Hariharan. *On the Efficacy of Knowledge Distillation.* 2019. arXiv: `1910.01348 [cs.LG]`. URL: `https://arxiv.org/abs/1910.01348`.

[2] Tim Dettmers et al. *QLoRA: Efficient Finetuning of Quantized LLMs.* 2023. arXiv: `2305.14314 [cs.LG]`. URL: `https://arxiv.org/abs/2305.14314`.

[3] Albert Einstein. "Zur Elektrodynamik bewegter Körper". In: *Annalen der Physik* 322.10 (1905), pp. 891–921.

[4] Simon Fear. *`booktabs` – Publication quality tables in LaTeX.* 2020. URL: `http://mirrors.ctan.org/macros/latex/contrib/booktabs/booktabs.pdf` (visited on 11/01/2023).

[5] Christian Feuersänger. *Manual for Package PGFPLOTS.* URL: `http://mirrors.ctan.org/graphics/pgf/contrib/pgfplots/doc/pgfplots.pdf` (visited on 11/03/2023).

[6] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics.* Vol. I-III. Reading, Massachusetts: Addison-Wesley, 1963. ISBN: 978-0201021165.

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition.* 2015. arXiv: `1512.03385 [cs.CV]`. URL: `https://arxiv.org/abs/1512.03385`.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network". In: (2015). arXiv: `1503.02531 [stat.ML]`. URL: `https://arxiv.org/abs/1503.02531`.

[9] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models.* 2021. arXiv: `2106.09685 [cs.CL]`. URL: `https://arxiv.org/abs/2106.09685`.

[10] Philip Kime, Moritz Wemheuer, and Philipp Lehman. *The `biblatex` Package.* URL: `https://ctan.org/pkg/biblatex` (visited on 11/06/2023).

[11] Donald Knuth. *Knuth: Computers and Typesetting.* URL: `https://www-cs-faculty.stanford.edu/~knuth/abcde.html`.

[12] Donald E. Knuth. *The TeXbook.* Reading, Massachusetts: Addison-Wesley, 1986. ISBN: 0-201-13448-9.

[13] Leslie Lamport. *LaTeX: A Document Preparation System, 2nd Edition*. Reading, Massachusetts: Addison-Wesley, 1994. ISBN: 0-201-52983-1.

[14] Frank Mittelbach and Ulrike Fischer. *The LaTeX Companion*. 3rd ed. 2 vols. Reading, Massachusetts: Addison-Wesley, 2023.

[15] Yasmin Moslem. "Efficient Speech Translation through Model Compression and Knowledge Distillation". In: *Proceedings of the 22nd International Conference on Spoken Language Translation (IWSLT 2025)*. Association for Computational Linguistics, 2025, pp. 379–388. DOI: 10.18653/v1/2025.iwslt-1.40. URL: http://dx.doi.org/10.18653/v1/2025.iwslt-1.40.

[16] Tobias Oetiker et al. *The Not So Short Introduction to LaTeX*. URL: https://tobi.oetiker.ch/lshort/lshort.pdf (visited on 10/31/2023).

[17] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV]. URL: https://arxiv.org/abs/1801.04381.

[18] Claude E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[19] Wolfgang Skala. *Drawing Gantt Charts in LaTeX with TikZ*. URL: http://osl.ugr.es/CTAN/graphics/pgf/contrib/pgfgantt/pgfgantt.pdf.

[20] Thomas F. Sturm. *The tcolorbox package*. URL: http://mirrors.ctan.org/macros/latex/contrib/tcolorbox/tcolorbox.pdf (visited on 11/05/2023).

[21] Till Tantau. *The TikZ and PGF Packages*. URL: http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf (visited on 11/02/2023).

[22] Chaofei Wang et al. *Efficient Knowledge Distillation from Model Checkpoints*. 2022. arXiv: 2210.06458 [cs.LG]. URL: https://arxiv.org/abs/2210.06458.

[23] Joseph Wright. *siunitx – A comprehensive (SI) units package*. URL: https://ctan.org/pkg/siunitx (visited on 11/08/2023).

[24] Zhiyuan Wu et al. *Knowledge Distillation in Federated Edge Learning: A Survey*. 2024. arXiv: 2301.05849 [cs.LG]. URL: https://arxiv.org/abs/2301.05849.

[25] Dan Zhang et al. *Parameter-Efficient Fine-Tuning for Foundation Models*. 2025. arXiv: 2501.13787 [cs.CL]. URL: https://arxiv.org/abs/2501.13787.

APPENDIX A

An appendix

Appendices may be included in your thesis but it is not a requirement.