Improving Knowledge Distillation For Smaller Networks Via Reducing Regularization

By

Mubarek Mohammed Yesuf

Submitted to the School of Information Technology and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science in Artificial Intelligence

Addis Ababa Institute of Technology

Addis Ababa University

Addis Ababa, Ethiopia

May 2023

Approval

This is to certify that this thesis titled ***Improving Knowledge Distillation For Smaller Networks Via Reducing Regularization*** is prepared by ***Mubarek Mohammed Yesuf*** and submitted in partial fulfillment of the thesis-option requirements for the Degree of Master of Science in Artificial Intelligence at School of Information Technology & Engineering, Addis Ababa Institute of Technology.

Advisor

Name	Signature	Date

Co-Advisor

Name	Signature	Date

Examiner

Name	Signature	Date

Examiner

Name	Signature	Date

Abstract

Knowledge Distillation (KD) is one of the numerous model compression methods that help reduce the size of models to address problems that come with large models. In KD a bigger model termed the teacher, transfers its knowledge, referred to as the Dark Knowledge (DK), to a smaller network usually termed the student network. The key part of the mechanism is a Distillation Loss added in the loss term that plays a dual role: one as a regularizer and one as a carrier of the categorical information to be transferred from the teacher to the student which is sometimes termed DK [1]. It is known that the conventional KD does not produce high compression rates. Existing works focus on improving the general mechanism of KD and neglect the strong regularization entangled with the DK in the KD mechanism. The impact of reducing the regularization effect that comes entangled with DK remained unexplored. This research proposes a novel approach, which we termed Dark Knowledge Pruning (DKP), to lower this regularization effect in the form of a newly added term on the Distillation Loss. Experiments done across representative and benchmark datasets and models demonstrate the effectiveness of the proposed mechanism. We find that it can help improve the performance of a student against the baseline KD even in extreme compression, a phenomenon normally considered not well suited for KD. An increment of 3% is achieved in performance with a less regularized network on CIFAR 10 dataset with ResNet teacher and student models against the baseline KD. It also improves the current reported smallest result on ResNET 8 on the CIFAR-100 dataset from 61.82% to 62.4%. To the best of our knowledge, we are also the first to study the effect of reducing the regularizing nature of the distillation loss in KD when distilling into very small students. Beyond bridging Pruning and KD in an entirely new way, the proposed approach improves the understanding of the knowledge transfer, helps achieve better performance out of very small students via KD, and poses questions for further research in the areas of model efficiency and knowledge transfer. Furthermore, it is model agnostic and showed interesting properties, and can potentially be extended for other interesting research such as quantifying DK.

Key words: *Deep Learning*, *Neural Networks*, *Model Compression*

Acknowledgements

First of all, I would like to start my acknowledgment from Addis Ababa University. I am extremely grateful to the leadership of Addis Ababa University Institute of Technology, including everyone on the staff involved, for taking the initiative to make this graduate program in Artificial Intelligence a reality. I would like to express my deepest gratitude to all current and past instructors in the department from whom I have learned a lot throughout the program.

I would like to extend my acknowledgment to my advisor Beakal Gizachew(Ph.D.) for his follow-ups, mentorship, and constructive feedback. I have a special appreciation for Dr. Elefelious Getachew, and Dr. Henock Mulugeta, for supporting me through this program since day 1 in every way possible. I would also like to express my sincere appreciation to the thesis examiners Dr. Sileshi Demesie, Dr. Adane Mamuye, and Dr. Fantahun Bogale for dedicating their time and providing valuable feedback during this thesis work.

My acknowledgment will not be complete without mentioning my office and colleagues, who supported me in all ways possible while I am engaged in this program. And of course to all my friends and classmates: Amanuel Negash, Dereje Tadesse, and Tesfaye Mengistu, who have kept supporting me through the final days of the thesis. Especially, Amanuel Negash, undoubtedly one of a kind AI researcher in the country, has been of great help through his insightful comments and questions.

Being a human, I might forget to mention people who played a significant positive role in favor of me. I would like to ask for forgiveness and extend this acknowledgment to them as well.

Lastly, I would like to acknowledge the help of the open and sharing internet community for platforming access to important resources.

Dedication

I dedicate this to my family. Each and every one of them. I am indebted to them for supporting me throughout this program and **in life**. I would have given up long ago if it weren't for their support. They kept on giving and giving without expecting anything in return. I LOVE YOU ALL, **GADZANA**.

This is especially dedicated to, Abaye, my Mom, and Gashe(AY), our father, who would appreciate this more than anybody else in this world had he still been here in this world. And to all our family members who left before and after him (AY).

I would like to extend my dedication to all those wonderful people in my life that fit the description of the word 'family' just as much.

And of course to THEE FAMILY, THEE LINE.

Contents

List of Tables

List of Figures

List of Abbreviations

AI Artificial Intelligence.

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

CPU Central processing unit.

DK Dark Knowledge.

DKP Dark Knowledge Pruning.

DL Deep Learning.

DNN Deep Neural Networks.

GAN Generative Adversarial Network.

GNN Graph Neural Networks.

GPU Graphical Processing Unit.

KD Knowledge Distillation.

ML Machine Learning.

MLP Multi Layer Perceptron.

NAS Neural Architecture Search.

NLP Natural Language Processing.

NN Neural Networks.

QAT Quantization-Aware Training.

RL Reinforcement Learning.

RNN Recurrent Neural Networks.

1 Introduction

1.1 Background

1.1.1 Artificial Intelligence, Machine Learning (ML), and Data

Artificial Intelligence (AI) can be broadly seen as the collective effort of certain disciplines, with the help of computer science, to mimic human intelligence [9]. It tries to achieve it in terms of *intelligent* capabilities such as vision, language, knowledge representation, planning, uncertainty, etc [10].

Machine Learning (ML) is a general term given to methods or systems that learn, mostly from data, using some kind of learning algorithm rather than being explicitly programmed [10] and it is the reason prominent advances in AI are made starting by [11]. Typically, the fundamental task of these learning algorithms is to generalize beyond the training samples they were trained with [12]. Customized for their own setup, these learning algorithms have three important components to achieve their goal: data, representation, evaluation, and optimization [12]. Then, the task of 'learning' can be thought of as the process of adjusting numbers, known as weights and biases, using the optimizer using the evaluation [12]. Therefore, a common Machine Learning (ML) requires an algorithm, data, and computation [12]. The main types of Machine Learning (ML) algorithms are **Supervised Machine Learning (ML)**, an approach where both the learning data and the 'supervision' or 'mentorship' for the learning come from the training data, **Unsupervised Machine Learning (ML)**, an approach where the algorithm learns from the data without directly, and **Reinforcement Learning (RL)**, human-like learning that happens via the interaction of the agent, the algorithm, and its environment [9].

Neural Networks (NN), type of ML algorithm inspired by how biological neurons signal to one another in the human brain, are behind the success of ML [13]. They are used as building blocks of NN and almost all real-world tasks can be modeled by some arrangement of these neurons[14]. Then, the question of how to arrange these neurons is again answered by looking at the brain which represents concepts in hierarchical ways, with multiple levels of

abstraction and processes information through stages of transformation and representation [15]. Inspired by this architectural depth, eventually, a class of algorithms that uses a stack of artificial neurons, termed Deep Learning (DL)algorithms, emerged and frankly as of this writing are state of the art in the field of AI [16] and are behind almost every breakthrough heard in the news since 2012 [11].

Data is a core reason why ML are fruitful [12]. Thus, the critical pre-training stage of an ML is related to data as it directly influences the choice of the model and the learning algorithm. Commonly, representativeness, quality, type of encoding, dimensionality, size, and most importantly whether or not it is labeled (annotated) or not are key characteristics considered for analysis [12]. The Machine Learning (ML) stage where all these points are addressed is called the pre-processing stage [12]. Data can be collected and prepared to create a *Dataset* by taking the above points into consideration [9].

A model is the output of a ML algorithm in the form of a mathematical or computational representation of a system, process, or problem domain [12], [16]. A typical ML task is the process of optimizing the model, basically a collection of numbers, the parameters which are organized into a carefully designed architecture or data structure that depends on the algorithm chosen [12]. In the context of the broader picture of AI, the model represents the brain of the agent that we are trying to make *intelligent* and use for predictions, generate outputs, or understand patterns in data [12].

Training a model is the process of adjusting its parameters, such as weights and biases so that it can accurately predict the output for a given input. This is done by what is known as the Back Propagation Algorithm [17]. By iteratively adjusting the model's parameters, such as weights and biases, the training seeks to minimize both training and validation errors, achieving a well-fitted model that captures the essential patterns in the data while avoiding excessive complexity or oversimplification [10].

Overfitting and Underfitting. Training a model can also be looked at from the perspective of finding the optimal balance between overfitting and underfitting [18]. Overfitting occurs when the network excessively adapts to the training data, capturing noise and irrel-

evant patterns, leading to a poor generalization of unseen data [19]. On the other hand, underfitting arises when the network fails to capture the underlying patterns in the data, resulting in high training and test errors [18]. The training process aims to strike a delicate equilibrium, where the network learns to generalize well without over-relying on specific instances. Since most NNs are large by design [20], in most cases the problem usually is overfitting which is mostly solved by Regularization [17] which is a fundamental concept in Deep Learning (DL) that helps prevent overfitting and improve the generalization performance of models. Overfitting occurs when a model becomes too complex to the point where it starts to memorize the training examples instead of learning the underlying patterns and relationships in the data [17]. L1 and L2 regularization [21], Early stopping [22], Data Augmentation [23], and Label Smoothing [24] most well known regularizers.

Different types of models and datasets exist in Artificial Intelligence (AI), depending on the specific task or problem they are designed to address [20]. Since these models and datasets are used as a landmark for progress in the AI literature, they are given names and serve as benchmarks for comparison [20]. MNIST [25], CIFAR-10 and CIFAR-100 [8]and Imagenet [26] are examples of benchmark datasets. Analogously, Alexnet [11], LeNet[25], and VGG16 [27] are examples of such benchmark models. The image below shows the evolution of these benchmark models.

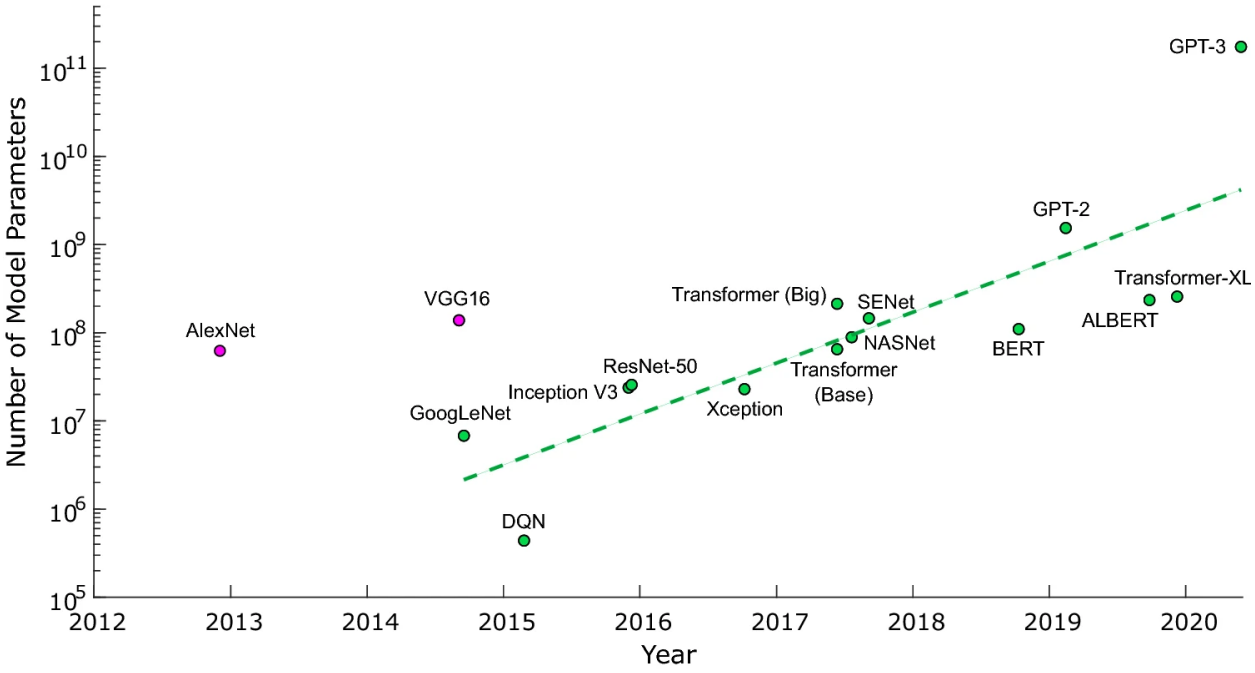

Figure 1: Size of benchmark vision and language models over the past years [2].

1.1.2 Problems of large models

The success of Deep Learning algorithms, as the name 'deep' indicated, is highly tied to their size even though bigger models are not necessarily related to performance [20]. Thus, model size is a consequence of an effort to increase a model's performance [18]. It did help to increase accuracy but on the other hand, bigger model size comes with its own disadvantages [28]. Challenges of giant models include:

- Time-consuming training, inference, and iteration of models [29]
- Hard to deploy on resource-constrained and edge devices
- Economic and environmental impact [28]
- Complexity [30]

Model compression is a method to make big models small without losing performance [31] in order to achieve a good balance between performance and efficiency.

1.2 Motivation

The exact number and arrangement of parameters in a ANN based model is still a hyperparameter, a decision of the designer. Bigger models are not necessarily related to performance [20], yet the current approach is to design them big and then optimize them mostly via model compression. Therefore, the first apparent motivation for anyone to study the area is the resource intensiveness of big models and the problems that come with the efficiency issues: Efficiency in terms of time, computation, data, etc. However, there are other contexts that motivate researchers to explore and study model compression. The following are some of them:

- To get a better understanding of how NNs work [18].
- Parallel growth of other technologies and their use cases that require small-scale models [9]. For example IoT for agriculture use.
- Growing environmental concerns [28]
- To use them as a tool for a task beyond size reduction such as to increase the performance of models [19]
- Inclusiveness of access to models [29].
- Re-purpose models for other tasks [29].

All the above-mentioned motivation played their role in pushing us to pursue this work.

1.3 Statement of the Problem

Knowledge Distillation [1] is a model compression method where the knowledge of a big model is transferred to a smaller model, usually called the student, in classification tasks. The key part of the mechanism is a Distillation Loss added in the loss term [1] as can be seen in equation 1, and it plays a dual role: as a strong regularizer [32] and as a carrier of the categorical information to be transferred from the teacher to the student which is sometimes termed as Dark Knowledge [1]. Even though, the inefficacy of the conventional KD to transfer knowledge from a teacher to a significantly small size student is known [33], existing theoretical developments focus on improving the conventional KD by decoupling the distillation loss into the target class and non-target class [34], replacing the outputs with other designed outputs [35], manipulating the outputs of the teacher [36] before feeding it to the student, adding noise to the outputs of the teacher [37], and exploring what makes KD work in general, [32], [38]. The effect of the strong regularization that comes entangled with the Dark Knowledge in the distillation loss remained unexplored in literature, even by studies that specifically attempted to make distillation effective in cases where there is a high size gap between a student and a teacher network [6], [39]. [6] studied the subject from the perspective of the growing size of teachers holding the student size fixed, to suggest that an extra new teacher network can fill the size difference. [39] introduced a new loss term abandoning the original KL divergence.

$$KD\ Loss\ =\ Hard\ Loss(output, Hard\ Label) + Distillation\ Loss(output, Soft\ Label) \quad (1)$$

This research studies the reduction of the regularization that comes entangled within the Dark Knowledge in KD by applying a novel approach termed DKP. It adapts the concept of pruning on the logits of the teacher. Pruning is another model compression method where the parameters of a model are removed or set to zero for efficiency [40]. But, in the context of the proposed approach, pruning is applied to the logits, the predictions, of the teacher model by setting them to zero.

1.4 Research Questions

Considering the above statement of the problem, the main research questions are:

RQ1 How can the regularization in Knowledge Distillation be reduced from the distillation loss for very small students?

RQ2 What happens to the performance of different-sized student networks if Pruning is applied to the teacher's predictions in the Knowledge Distillation framework?

RQ3 How do different hyper-parameters perform when Dark Knowledge Pruning is applied to a student?

1.5 Objectives of the study

1.5.1 General objective

The general objective of the research is to study the effect of pruning the logits of a teacher network as a means to shrink the regularization in the distillation loss of the Dark Knowledge.

1.5.2 Specific objectives

- Propose a mechanism to reduce the regularization in KD
- Design an experiment set up to apply the proposed concept.
- Apply optimization techniques to increase the performance of the DKP framework.
- Experiment with the proposed approach across different hyper-parameters to check performance when Dark Knowledge pruning is applied.
- Compare the accuracy of compressed networks with and without pruned Dark Knowledge in KD across different student network sizes

1.6 Scope

In general, the limitations emerge out of the lack of enough computation access. The following points describe the scope of the thesis.

- This thesis studies the effect of applying Pruning teacher predictions (logits) on conventional Knowledge Distillation. There are many variants of both Pruning and Knowledge Distillation [33] but here only magnitude pruning and random pruning, along with vanilla Knowledge Distillation are studied.
- The conventional KD [1] comes with its own hyperparameters such as temperature and weight for distillation loss. Only the temperature is varied, in this experiment, we kept the latter fixed.
- The experiments done are on computer vision tasks. Since the underlying principles of training are similar, they can be extended for other tasks with minor modifications.
- Here, ResNet and LeNet are used along with CIFAR-10 and MNIST as they are used commonly in literature [20]. Due to computational limitations, experiments on ImageNet are not considered in this study.

1.7 Significance of the study

In general, studying Dark Knowledge in KD will be valuable in several sub-domains of Artificial Intelligence (AI). Especially, in the model efficiency and knowledge transfer domains, it will have direct benefits. The following are some of the significance of this study in the efficiency and knowledge transfer realms.

- A novel approach that can be used in the model efficiency and knowledge transfer sub-domains of AI
- Small but performant models are required to be used in an energy-effective way. This study will allow a better transfer of knowledge in KD.

- Quantifying the Dark Knowledge can be attempted as an extension of this research and it will be a significant development to take full advantage of KD.
- It will, when extended, help debug the knowledge of a network in the context of compression or otherwise which is beneficial for mechanistic interpretability, an emerging and critical area of study.
- Pruning, in this context, is removing the contribution of the softmax outputs. Initially, it has the effect of generalizing the conventional Knowledge Distillation (KD).
- It also for the first time links Pruning and KD in an entirely new perspective. There has been very little work in applying KD to help recover the generalization of pruned networks. But, applying the idea of pruning to help Knowledge Distillation is being applied for the first time. This is an extendable work as both methods are widely researched and applied and one can easily extend the idea to their variants. For example, pruning in feature-based KD instead of Response based Knowledge Distillation.
- Initiates a whole line of research questions as discussed in the future works section 5.1.

1.8 Contribution

The following are the contributions of the paper.

- Proposed a new approach in the form of a loss term, the third term in equation 2, that will allow better control of the flow of knowledge in Knowledge Distillation (KD) and also generalize it.

$$Loss = \sum_{i=1}^{C} q_i * log(p_i) + \sum_{i}^{C} q_i * log(q_i/p_i) - \underline{\sum_{i=1}^{s} q_i * log(q_i/p_i)} \tag{2}$$

  where s is the percentage of logits to prune or set to zero.
- Performance improvement. If one network is compressed in the conventional KD and another network is compressed in the proposed approach, with selected hyperparam-

eters, the latter can outperform the former as demonstrated in the experiments and results section 4.

- A model agnostic way of reducing the effect of regularization in KD.
- It will, when extended, help debug the knowledge of a network in the context of compression or otherwise which is beneficial for mechanistic interpretability, an emerging and critical area of study.
- It initiates research questions at the intersection of Pruning and KD. Different variants can be explored for a deeper analysis of efficient knowledge transfer.
- To the best of our knowledge for the first time study the regularization effect in KD for the inefficacy of KD for extreme compression, compressing networks to a size that is very low compared to the teacher. Demonstrated cases where extreme compression disentanglement of regularization and Dark Knowledge in the case where a less regularized network outperforms the regulated contrary to conclusions made earlier.

1.9 Thesis Organization

The remainder of this document is structured as follows. The theoretical backgrounds of AI, statement of the problem, research question, and objectives of the study respectively have been discussed in this first chapter. The next chapter, chapter 2 2, will be about a literature review on Model Compression in general and related works that are closely related to the proposed work in this thesis. Chapter 3 discusses the methodology of the thesis. It contains important details about the proposed solution for the problem statement, its mathematical interpretation, the datasets used, and the justification as to how the proposed method solves the problem. Chapter 4 entails the experimental setup and results followed by the conclusion and future works in chapter 5.

2 Literature Review

2.1 Deep Neural Networks (DNN) and their variants

The huge success of Artificial Intelligence comes from a type of Machine Learning algorithms fundamentally based on a biologically inspired unit termed the Perceptron that mimics a neuron in a human brain [13]. A neuron is a special biological cell that accepts, processes, and transmits information through electrical and chemical signals [41]. They are fundamental building blocks for Deep Learning (DL) algorithms which are made up of multiple layers of these artificial neurons [16]. Each layer of neurons performs a specific task, and the layers are connected to each other in a way that allows the model to learn complex patterns from the data [41]. The simplest and most basic form of a DL network arrangement is the **Multi Layer Perceptron (MLP)** [41]. MLPs are composed of multiple layers of these interconnected artificial neurons. Non-linearity is applied at each layer to transform the input data into a representation space [10].

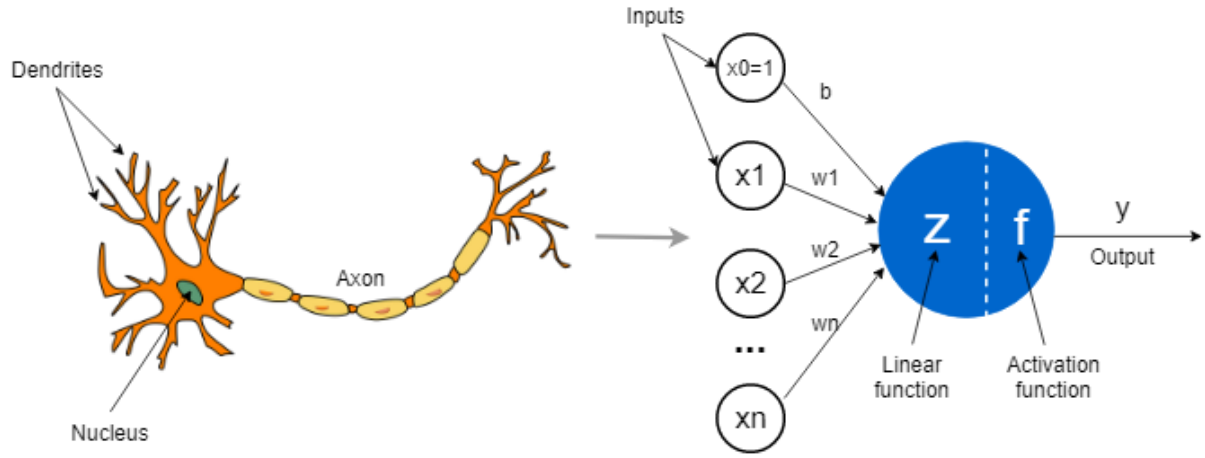

Figure 2: The perceptron [3]

There are different variants of Neural Networks. The following are the most important ones.

- **Convolutional Neural Network (CNN)** s are a type of NNs that try to mimic a

human vision with a design that enables them to pick up spatial 'perception' characteristics. Commonly, they are made up of building blocks, such as convolution layers, pooling layers, and fully connected layers which can be considered primitives of the CNN architecture. They are one of the most successful NN architectures for vision-related problems in AI [11]. Prominent examples from these architectures are LeNet [25] and ResNet [42]. LeNet is a model constructed for the recognition of pictures of handwritten digits [25]. The network is a mix of a Convolutional Neural Network for feature extraction followed by an MLP network for classification. The convolutional layers that make up the Convolutional Neural Network part entail convolution, pooling, and nonlinear activation function(tanh) activation functions sections. The other model the study is conducted with is a family of network architectures referred to as ResNet [42]. These networks are designed to make deeper networks possible by a specifically designed architectural trick termed the Residual module or connection as described in 4. In networks before ReSNET, the deeper the network, the larger the error they make. This problem is called 'the vanishing gradient' problem [10] and it happens when the parameters earlier in the network don't get enough gradient information to learn from [42]. ResNet is a name for the family of architectures that use such residual connection and there are multiple instances of this family that are used in literature for benchmark purposes [20].

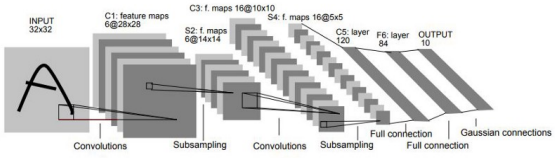

Figure 3: LENET

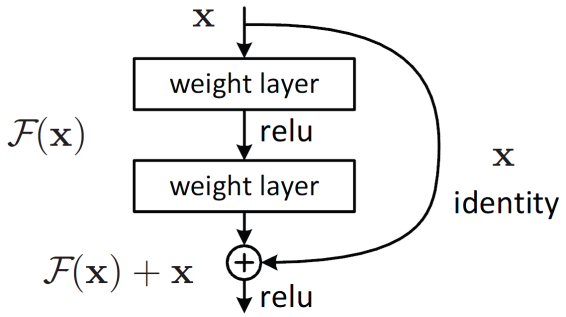

Figure 4: Residual connection

- **Recurrent Neural Networks (RNN)** are a type of NN commonly used for process-

ing sequential data, where the order of elements in the sequence matters [43]. RNNs have a recurrent connection that allows information to be passed from one step to the next, enabling them to capture temporal dependencies and context [43]. The fundamental building block of an RNN is the recurrent neuron or cell, which takes input at each time step and produces an output and a hidden state which serves as the memory of the network, allowing it to maintain information about past inputs [10] . The hidden state at each time step is influenced by both the current input and the previous hidden state, creating a feedback loop. They are widely used in various natural language processing tasks, such as language translation, text generation, sentiment analysis, and speech recognition [10]. They can also be applied to time series analysis, including forecasting, anomaly detection, and signal processing [10].

- **Long Short-Term Memory (LSTM)** and **Gated Recurrent Unit (GRU)** are introduced to solve the problem of The vanishing or exploding gradient problem is one of the challenges of training Recurrent Neural Networkss where the gradients become extremely small or large as they propagate backward through time which can lead to difficulties in capturing long-term dependencies [44]. Different These variants incorporate gating mechanisms that control the flow of information, enabling better handling of long-term dependencies and mitigating the gradient problem [44].

- **Generative Adversarial Network (GAN)**s are a family of complex networks that use a combination of two or more fundamental NNs to generate synthetic data based on real ones [45]. They have at least two components: generative and discriminative [10]. The generative part is a network whose output is data that one wishes to mimic for example an image. The output of the generator is fed into the discriminator which is a separate network that takes in the generated data, and classifies it according to a label [10]. The generator is good enough when the discriminator, which was trained on the real samples of data to be generated, starts to label the generated data as the right class. These networks have the potential to be used for both good and evil intentions as they can simulate any data distribution [45].

- **Autoencoders** are NN that learn to compress and reconstruct unlabeled data to be used for unsupervised learning, dimensionality reduction, anomaly detection, image denoising, and inpainting, and feature extraction [46].

- Graph Neural Networks (GNN)s are NNs that learn from graph-structured data [47]. They do this by aggregating information from neighboring nodes and iteratively updating the node representation to capture both local and global patterns and dependencies within the graph [47].

- **Transformers** are general-purpose architectures based on the attention mechanism, which allows them to learn long-range dependencies in sequences [48]. As of this writing, they are state-of-the-art in Natural Language Processing tasks such as machine translation, text summarization, and question-answering [20].

2.2 Model Compression

Model Compression is a set of techniques [31] for reducing the size of large models without a notable performance loss. Its necessity is increasing in parallel with the advancement and complexity of models. Even though their recent attempts to train smaller networks from the beginning [49], mostly, Model Compression is applied after training a bigger model as the model needs much fewer parameters for inference than for learning. Model Compression methods in literature can be classified into four major parts: Pruning, Knowledge Distillation (KD), Quantization, and other methods. Pruning is removing an unwanted structure from a trained network. KD is a mechanism to pass along the knowledge of a bigger model to a smaller model. Quantization is reducing the number of bits of model parts required to be represented, similar to approximating a number [50]. For convenience, the rest of the Model Compression methods other than Pruning, KD, and Quantization can be organized in one section and are referred to as Other methods.

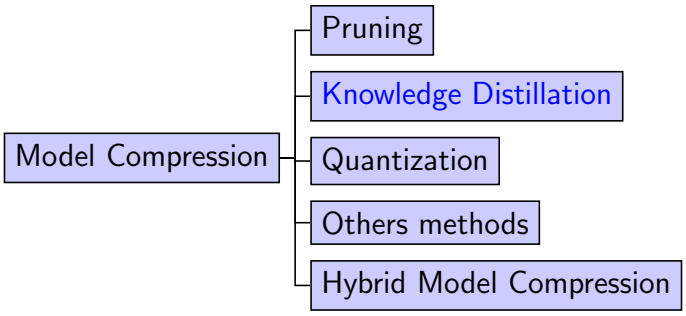

Figure 5: The taxonomy and categories of Deep Learning model compression methods.

2.2.1 Pruning

The oldest, most studied, and intuitive method of reducing model size is pruning [51]. It is literally removing the weight of a node from a network based on how important it is. The process introduces its own hyper-parameters. These include the compression rate [40], magnitude to Prune with when it is magnitude weight, type of saliency metric to use, etc. The approach towards these hyper-parameters categorizes the Pruning methods in the literature. The earliest methods were mainly: Saliency based, where node or weight is checked for importance, penalty based [19], [51], where the penalty term is added on the loss(objective) function to induce weight decay [52], magnitude-based pruning, simply removing the smallest link(weight) in each iteration.

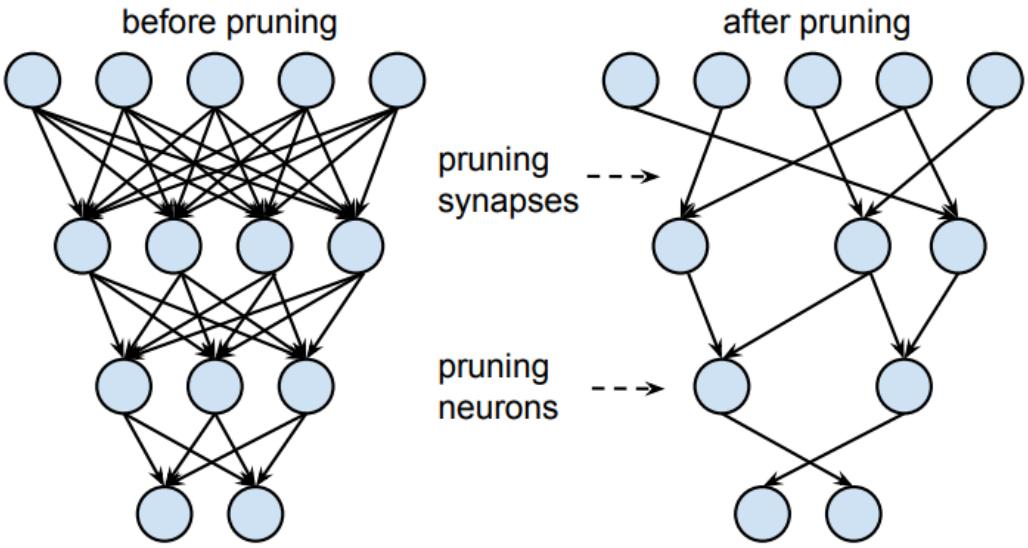

Figure 6: Pruning a network

A more 'educated' way of dubbing a parameter as unimportant based on interpretability or explainability [53]. Explainability is the notion of trying to explain the 'decision' of a model. This is equivalent to asking the question of what part of the input cause a specific prediction. Currently, that is a hot topic as applications of Artificial Intelligence (AI) prevail in more and more sectors including high stake areas. There are well-established methods out there and a couple of them have been used in the context of pruning. An area of growing interest is what is being popularized as Interpretable Pruning, an attempt to know what is actually being pruned. Layer-wise Relevance Propagation (LRP) algorithm is proposed in [53] to assign relevance scores to each and every unit in the network. The work in [54] demonstrated a Neuron Importance Score Propagation (NISP) algorithm to propagate the importance scores of final responses to every neuron in the network and then prune the CNN filter with the least importance. Interpretability based Filter Pruning (IFP) is introduced in [55] that utilizes Activation Maximization, a form of feature visualization method after each activation layer of a CNN, to identify non-essential filters. The method identified filters that learn redundant features and pruned them. An attempt to dynamically determine what neurons to skip at inference time is demonstrated by [56]. This method incorporated a

decision unit before every CNN layer which takes the input and gives output an indication of the most probable channel. In convolutional neural networks, in addition to nodes and weights, now there are channel pruning methods ([57], [58]) and filter pruning [59] pruning methods. Such methods came to be known as structured pruning, removing structures, as opposed to unstructured pruning, including removing weights or nodes as in the legacy approaches.

Modern pruning-based compression approaches vary not just in the Saliency metrics they use or other approach, but also in the overall pipeline of pruning. The 'lottery ticket hypothesis' [49], whose stronger version was supported in [60], states that random initialization of a network contains a small subnetwork (" the winning tickets") that, when trained in isolation, can compete with the performance of the original network. A rather bold finding by [61] states that there is a randomly initialized sub-network that even without training can perform as well as the original. The recent work [62], termed pruning from scratch, questions the need for training the network and proposes a different pipeline as random initialization, pruning, and then training. The work in [63], raises the question of why train a big model to convergence when it is possible to attain better performance by training it for free iterations and pruning it.

A new perspective in pruning is treating the task as a search in the space of sub-networks. This is a consequence of advances in Neural Architecture Search, an emerging area of study in the study of neural networks. The researchers in [64], applied neural architecture search to search directly for a network with flexible channel and layer sizes where the number of the channels/layers is learned by minimizing the loss of the pruned networks.

Pruning methods rely on manual design to get optimal performance. An emerging approach is to automate the process of pruning using Reinforcement Learning methods following advances in the area. RL based pruning methods formulate the problem of removing ad node or weight as a Markov Decision Process . These methods of pruning differ in their design of the RL system which essentially requires defining a state space, an action space, and a reward. The action and state space are obvious enough as they are almost always the

set of all 'removing' and the set of all possible sub-networks respectively. The rewards for pruning can be a dense [65], appearing after every action is taken, or sparse[66], coming or appearing to the agent after a certain sequence of actions. In [65] a magnitude is determined in each step to remove or leave a layer. Almost all RL based methods incorporate structured pruning for the actual pruning, which can be saliency or magnitude based. This is because RL methods are by nature iteration intensive and if individual parameters are considered for each action-state pair, the complexity will be exponential.

The most important gap common in all the pruning methods is that it is hard to compare different pruning variants due to a lack of benchmark metrics. The only benchmark that exists Shrinkbench [67], a framework that framework aiming to provide pruning performance comparisons.

There is a critical difference between unstructured [40] and structured pruning g [59]. Unstructured pruning is implemented by making individual parameters be pruned to zero [40]. This makes unstructured pruning almost surgical and more accurate in identifying what to remove. But this implementation doesn't actually reduce the computation time since the GPU will do the operation anyway. On the other hand, structured pruning will remove a complete layer or channel, or filter. And then the rest of the connections will be connected afterward. This increases efficiency in both pruning time and implementation time. But this too has its own consequences as it is not surgical. The layer or channel to be removed might still have important weights that are needed for inference.

Another common limitation of pruning methods, which can be caused by the lack of a common benchmark, is that they are architecture-dependent [59]. Albeit efforts in modern approaches[49], almost all of them involve at least some amount of fine-tuning or retraining which requires having original data to train the network is trained on which in some cases might not be available [40]. Considering randomly dropping weights or magnitude-based pruning [40], it is justified to say that pruning is the fastest compression method.

2.2.2 Knowledge Distillation

Knowledge Distillation is a relatively new compression procedure originally introduced in [31] and reinvented in [1]. A bigger model and well-performing model, which is termed the teacher model, provides labels for a smaller network to train with. For each observation, data label pair, out of the dataset the original network was trained on, the smaller network will use the predictions of the bigger network as a label along with the observed label. These predictions of the teacher model are termed soft labels since they are not ones and zeros like hard labels are. These soft labels are termed **Dark Knowledge**. The intuition is an image of a dog has a larger chance of being mistaken as a cat than as a car [1]. This is the categorical information being transferred from the teacher to the student.

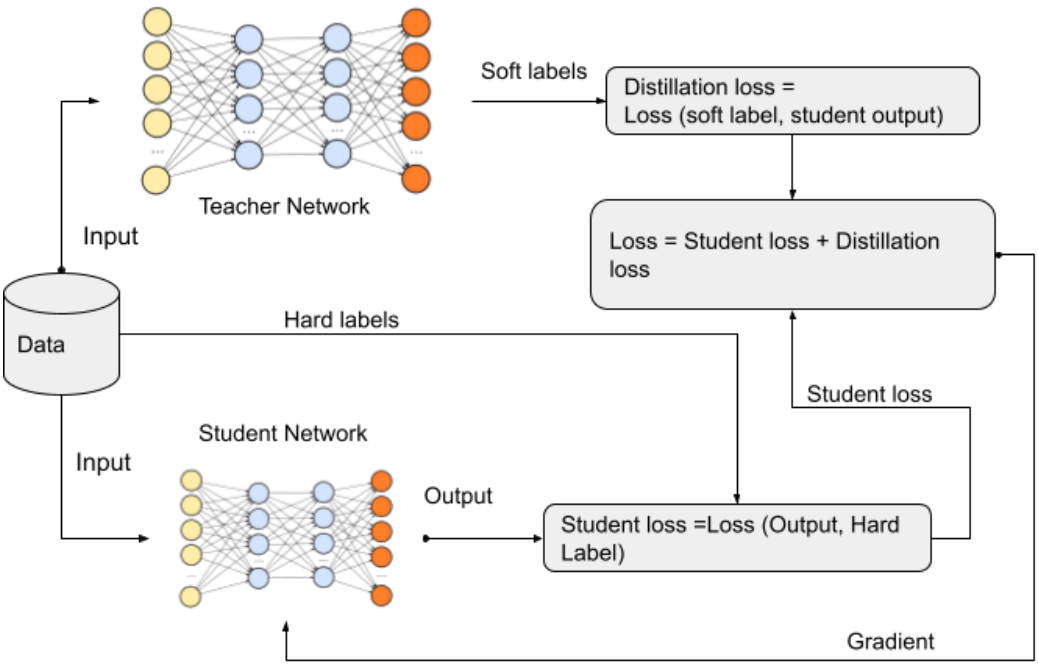

Figure 7: Knowledge Distillation (KD) [4].

KD uses a constant value named Temperature, T, which magnifies the small probability

values in the prediction(the soft labels) which represent. Each output of the i-th class in training, with logit z and a temperature T, is computed as:

$$p(z_i, T) = \frac{exp(z_i/T)}{\sum exp(z_j/T)} \tag{3}$$

A powerful adaptation of KD, Fitnets [68], trains a student, deeper and thinner than the teacher which is unlike [1], using the intermediate representations, features, learned by the teacher as hints to improve the training process and final generalizing capacity of the student. This work opened a whole new sub-category of KD termed as Feature Distillation making the original to be called response-based distillation [69]. Response-based distillation is also limited to supervised learning as the logits, the soft labels, are probability distributions. Relationship distillation extends feature distillation by using outputs of specific layers in the teacher model and exploring the relationships between different layers or data samples [70].

Knowledge Distillation methods actually construct a brand new smaller model that tries to mimic the bigger model [32]. This means training a new model from scratch, and that will take as much time as the training time for the teacher. The loss has to be a softmax loss function whose output is a probability distribution [1]. methods, for compressing a NN, don't give a bigger size reduction [33]. Some might even need a teacher assistant to catch up to the teacher[71].

At the core level, KD is a knowledge transfer mechanism [72]. Thus, it is extremely versatile and model agnostic, any model can be used as a teacher or a student. Thus, it has been used for other purposes including faster optimization [73], defense against adversarial attacks[74], explainable networks [75], and even to improve the original network itself [76].

2.2.3 Quantization

Quantization, in simple terms discretization, is rounding up parameters of a network to reduce the number of bits required for representation. For example, reducing from 32 bits floating point representations to limited precision, commonly in 8 bits or lower representations. The basic method of quantization is to train a model under a normal setup and

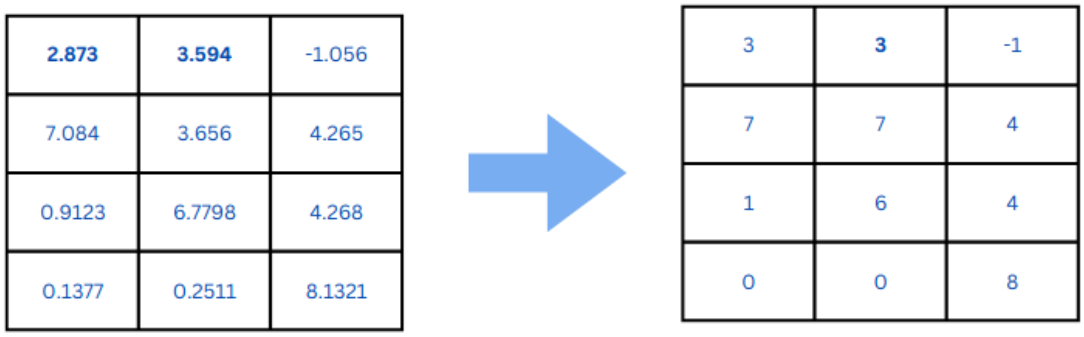

Figure 8: Quantization

then to quantize each parameter, activation, or even input, using some sort of function or mapping, which is sometimes referred to as a quantization operator [77]. When fine-tuning is needed after quantization, Quantization Aware Training is used where a re-training is done with floating point propagation but the weight is Quantized back after each update. Post Training Quantization is quantization without retraining [78], [79]. It is preferred when QAT is too complex to implement.

Normally, what is known as the clipping range, a range to make sure the individual weights will lie in, is determined beforehand. The common formula is given:

$$Q(r) = Int(\frac{r}{S}) - Z \tag{4}$$

where Q is the quantization operator, r is a real-valued number, since the clipping range is not necessarily input (activation or weight), S is a real-valued scaling factor, and Z is an integer. These variables are part of the hyper-parameters introduced by Quantization and different applications' adaptations of them create different variants of Quantization. Int is the integer function that maps its input to the integer. A quantized number can be recovered with reverse quantization, Dequantization.

The ideal Quantization is Binarization, making the parameters of a network binary [80] for memory and compute efficiency. In fact, the most appealing feature of this method is that it can potentially get rid of the need for multiplication at inference time by replacing the dot product in the network with bitwise operators [81] to train networks with binary

weights. In [82], the researchers showed CNNs with only binary weights and demonstrated a binary quantized version of Alexnet [11], with 32 times smaller, with comparable accuracy as the original version. Beyond binarization, an alternative network is a ternary network, where the weights of the network are limited to three values instead of two as in binary. Neither binary nor ternary Quantization methods are trainable with back-propagation with gradient descent. Recent advances introduce Differentiable Quantization in order to learn the hyper-parameters of Quantization [83].

2.2.4 Low-Rank Tensor Decomposition

Low-Rank Approximation [84] is a mathematical technique used to approximate a given matrix by a matrix of lower rank. Particularly, Singular Value Decomposition (SVD) is a widely used method for low-rank approximation in mathematics, as it decomposes a matrix into three components: U, Σ, and V. By selecting the most significant singular values and their corresponding singular vectors, a lower-rank approximation of the weight tensor can be obtained.

In Neural Networks (NN)s context, the weight tensors represent the parameters that define the model's behavior and low-rank approximation techniques aim to find a lower-rank representation of the weight tensors that can approximate the original tensors accurately [85]. They provide a way to reduce the complexity of these weight tensors while preserving important information [85]. Especially, they are beneficial in Convolutional Neural Network (CNN) based architectures that normally consist of convolutional layers with a large number of weight tensors [85]. Applying low-rank approximation to these weight tensors allows for a substantial reduction in computational and memory requirements without significant loss in performance [86]. This is especially beneficial in scenarios where large-scale CNN models are deployed on resource-limited devices or where efficient training and inference times are critical. Thus, Convolutional Neural Network (CNN)s can be improved significantly [87], [85], [88], [86].

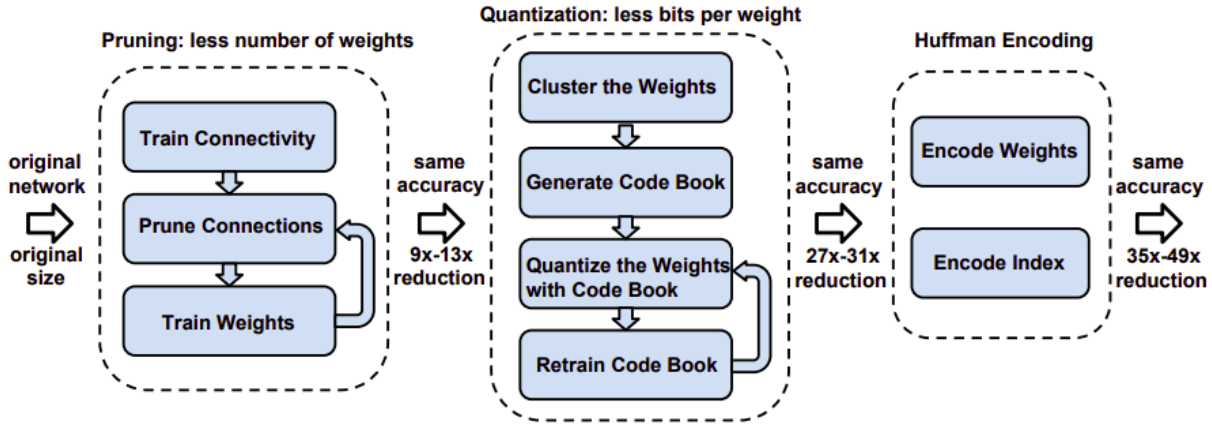

Figure 9: An example of hybrid Compression is the Deep Compression framework [5].

2.2.5 Hybrid Model Compression Methods

Fortunately, Model Compression methods are not disjoint: they can be combined [5]. The resulting process would be more complex than the individual ones, but combining hybrid methods is beneficial in cases where that is not an issue [5]. In fact, a symbiotic relationship where the problems of one are addressed by the other can be achieved [89]. This section highlights prominent works that combine compression methods.

Pruning and Quantization Pruning and Quantization are the most related ones especially unstructured Pruning which is simply setting a parameter zero [80]. The most prominent work not just in the hybrid paradigm but also in Model Compression is the Deep Compression framework [5] where iterative Pruning, Quantization, and Encoding are applied one after the other. Its Pruning part is the same as [40].

The intuitive relationship between Pruning and Quantization is backed by recent advances in Quantization that study differentiable hyperparameters in Quantization as a means to unify them [90], [91].

Pruning and Knowledge Distillation There is also a positive relationship between Pruning and Knowledge Distillation and it has been recognized relatively early in [92] where Pruning is applied on distilled student networks on the task of Neural Machine Translation.

Recent works are more generic than applied. A work demonstrated in [93] showed, a student can learn better from the pruned teacher which serves as a better regularizer than otherwise in the context of KD. The converse of it has also been shown in [94] where the authors argue that the accuracy of a pruned model is well restored when the student is fine-tuned with KD than vanilla training. A good example of a symbiotic relationship between the two is [89] where unprunable parameters are compressed with KD and potentially redundancy is addressed by Pruning[89].

Quantization and Knowledge Distillation

An early major work in the combination of Quantization and Knowledge Distillation is the application of Quantization for the choice of a student model for KD [95] where a the knowledge of a Quantized teacher model is transferred to a Quantized student. In this setting, KD helps Quantization restore better performance. Conversely, Quantization also benefits KD as in [96] where it has been used to enable on-device KD by introducing noise that mimics the existence of a bigger teacher model. This addresses the problem of missing teacher on edge devices to do vanilla KD and might even put the need of the teacher in the first place in question. Their bond is strengthened by Quantization Aware KD [97] that aims to solve the problem of performance degradation in Quantized distillation [95].

2.2.6 Other Model Compression Methods

In general, the goal of compression is intuitive: reduce size, and maintain accuracy. Thus, people have tried whatever they think would work to achieve that goal including information theoretics-based approaches [98], and Weight sharing and efficient architectures [99], Inception [100], MobileNet [101] and SqueezNet [102].

2.2.7 Summary

The main model compression methods, Pruning [40], Quantization [50], and Tensor Decomposition, in general, model compression can be seen in two broad categories. The methods in the first category transform a trained model into a small model using a certain technique.

Table 1: Summary of Hybrid Model Compression.

Hybrid Compression	**Remark**
Pruning and Quantization	can be applied one after the other easily because spare networks can be quantized easily [5].
Pruning and Knowledge Distillation	Mutually beneficial result. One solves the challenges of the other.[89], [93] ,[92]
Quantization and Knowledge Distillation	Distillation restores performance damaged by Quantization [95]. Quantization helps mimic a non-existent teacher [96]

Pruning [40], Quantization [50], and Tensor Decomposition methods can be seen as model transformation methods. They start with a trained model and transom it into an optimized version of itself [40]. The methods in the second category try to create a new but smaller model from scratch [1]. Knowledge distillation [1] can be seen as creating a model from scratch, in the second category. Any other design choices that can reduce model size can be categorized in the second category [98]. The following table summarizes these methods.

Type	Summary
Pruning	Removing unnecessary structures from a network
Knowledge Distillation	Transfer knowledge of a bigger model to a smaller
Quantization	Reducing the bits required to represent parameters
Low-Rank Tensor Decomposition	Weight metrics decomposition for CNNs
Hybrid methods	Methods that combine two or more of them
Other methods	Efficient design, dynamic NNs, etc

Table 2: Summary on Model Compression methods

2.3 Related Work

As shown in [1] and also discussed in the previous section of this paper, KD works better than a normal training mechanism. For a given network with logits z_i, the softmax is computed as:

$$p_i(z_i, T) = \frac{exp(z_i/T)}{\sum exp(z_j/T)} \tag{5}$$

Then the distillation loss becomes :

$$Loss = \sum_{i=1}^{C} q_i * log(p_i) + \sum_{i=1}^{C} q_i * log(q_i/p_i) \tag{6}$$

Where $p = [p_1, p_2, ..., p_C], q = [q_1, q_2, ...q_C] \in R^1C$, are probability distributions from the student and the teacher respectively. The second term in 6 is what is considered to contain useful information from the teacher [1]. It is known as the Dark Knowledge. It is also known that it serves as a regularize.

Since the inception of Knowledge Distillation, there have been several attempts to understand better the underlying mechanism of how it exactly works [38]. In general, they can be seen in two broad categories: those that consider the method in general and those that try to understand the mechanics around the loss function and the logits of the teacher and student. Due to the former class of works, [72] argued that the conventional KD [1] can be unified with the theory of learning with privileged information [103]. Unlike KD, the information proposed in [103] is not necessarily predictions on a data point as a label, but it can be different such as identifying the easy examples from the hard ones. In [38], they tried to understand the exact knowledge transferred from the student to a teacher network by using linear and non-linear polynomial classifiers and determined that the data geometry, optimization bias, and strong monotonicity of the training set, how increasing the training set also increases the performance, as key factors for the success of distillation. A relatively recent work [32], raised the concept of *fidelity*, the capacity of the student to alight with the outputs of the teacher. They claim that matching the teacher network can be a non-desirable property to have for the student network, and the knowledge transfer from the

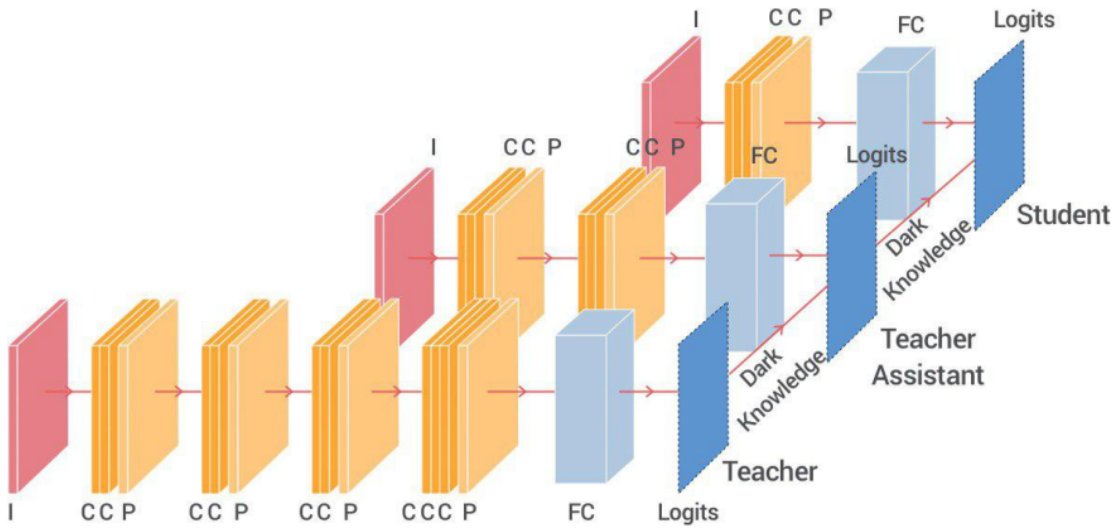

Figure 10: Distillation with teacher assistant [6]

teacher to the student model in KD is very limited. What seems like an opposite result is found in [104], where they argue that a good student network is the one that matches a well-performing teacher network the least. But this can be explained by the fact that a large and highly accurate teacher network induces give outputs that are close to hard labels. [104] proposed to apply early stopping, another regularization technique, as a solution to the performance and knowledge transfer trade-off. Overall, these contrasting perspectives highlight that the degree of fidelity in KD and the optimal strategy for knowledge transfer can depend on factors such as the complexity of the teacher network, the nature of the data, and the performance goals for the student model [33].

Tampering with the mechanics around the loss function and the logits of the teacher and student is becoming the trend to learn more about the distillation mechanism [37]. Complete reliance on having a well-performing teacher network is washing away as time as the understanding of the mechanism increases [105]. This line of work is partially inspired by exploring a non-performing teacher network and trying to understand the dark knowledge separate from its regularization effect. A good example of such works is [76] where the researchers tried to swap the non-argmax logits of the teacher in order to understand the value of the dark knowledge. With this regard, [71] is the closest related work in terms of

studying the size gap between a student and the teacher network. Their concern was to enable teachers to get bigger and bigger rather than studying when the size of the students as they get smaller and smaller a case where the regularization part of distillation can be a concern [71].

It has been shown that two students can learn mutually without having a teacher network to provide the dark knowledge [105] for themselves. In cases where a teacher is not performing well another network that sits between the student and teacher model can be used to enhance the outputs of the teacher before going to the student [36]. In another attempt to understand and exploit the nature of the dark knowledge from a well-performing teacher, [34] treated the loss in the conventional KD as a separate sum of two terms: the target class which is a binary distribution of the prediction for the target class in both the student and teacher outputs, and a loss term for the predictions both the student and the teacher for the classes that are not target classes [34]. Both of the two terms are controlled with a weight. This line of work, playing with the mechanics of the teacher logits and loss function goes as far as removing the need for a teacher network itself. [106] proposed a teacher-free framework to do regularization by designing outputs inspired by label smoothing.

One known limitation of KD is that it does not give a high compression rate [33]. Existing works neglected this issue as there is only one work that studied this phenomenon [6]. the core contribution of the conventional Knowledge Distillation (KD) lies in the second term of the loss function in the training of the student network 6. This second term appears in the form of a regularizer, indeed a strong one [1]. Therefore, there is an overlap between the service of Distillation Loss as a regularizer [32] and its service as purely valuable categorical knowledge [1]. This entanglement of the regularization and Dark Knowledge in the distillation is not explored in literature. The following table summarizes the different kinds of loss functions discussed in related work.

Table 7 summarizes related work on KD where the Distillation Loss is tampered with. It also includes conventional label smoothing and confidence penalty that play a similar role to the distillation loss plays in KD. The proposed method is applied at the end to make

Type	**Added Loss Terms**	**Reference**
Knowledge Distillation (KD)	$\sum_{i=1}^{C} q_i * log(q_i/p_i)$	[1]
Decoupled KD	$\alpha(q_i * log(q_i/p_i)) + (1-\alpha)\sum_{i=1}^{C} q_i * log(q_i/p_i)$	[34]
Teacher-Free KD	$\sum_{i=1}^{C} q_i^d * log(p_i) + \sum_{i=1}^{C} q_i^d * log(q_i^d/p_i)$	[106]
Confidence penalty	$\sum_{i=1}^{C} p_i * log(p_i)$	[107]
Label smoothing	$(1-\alpha) * y_i + \alpha/C$	[24]
Proposed	$\sum_{i}^{C} q_i * log(q_i/p_i) - \sum_{i=1}^{s} q_i * log(q_i/p_i)$	1

Table 3: Summary on penalty terms discussed in this section

comparison easier.

3 Methodology

This section describes the methodology of the research. It starts with the research design. Then continues with the proposed approach which is termed Dark Knowledge Pruning, the intuition behind it, and the algorithm proposed which is the core part of the thesis where a slightly modified version of the conventional Knowledge Distillation (KD) is proposed. After that, a brief description of the datasets and models used for the experiment is mentioned. The two datasets, MNIST [7] and CIFAR-10 [8], are chosen as they are used in literature for similar tasks [20]. The expected analysis directions are described i.e. how a network is impacted by the proposed approach mathematically. Finally, the evaluation section is presented.

3.1 Research Design

Design Science Research (DSR) is a problem-solving paradigm in studies such as information systems, software engineering, and AI that seeks to create innovative artifacts to solve problems and improve the environment in which they are instantiated [108]. We adapted the standard DSR procedure in [109] to our work into the following processes :

- **Problem Identification**: we clearly defined the problem we are trying to solve in 1.3 as the inefficacy of Knowledge Distillation to produce high compression rates. The problem lies in the umbrella of model compression and thus implicitly contains the motivation lies embedded. Then subsequent research questions were provided in 1.4.
- **Objectives**: We set out a more general and encompassing objective that is decomposed into smaller objectives to serve as a guide to address the research questions we set out.
- **Design and Development :** We proposed an adaptation to an existing mechanism that is theoretically sound and will allow us to successfully reduce regularization in KD. To ensure theoretical soundness we analyzed the mathematics and checked it against an already existing formulation of KD in terms of grounded theoretical frameworks [6], [34].

- **Experimentation**: A series of experiments will be designed and conducted on well-suited datasets to evaluate the performance of the proposed algorithm enhancement. The datasets will be randomly divided into appropriate-sized training and test sets as allocated by the designer of the datasets as described in 3.6. The training sets will be used to train the new algorithm enhancement and the baseline KD. The test sets will be used to evaluate the performance of the new algorithm enhancement and the baseline KD.

- **Evaluation**: We set an appropriate evaluation for the problem set up which is described in section 3.8.

Figure 11: Research Design

Figure 11 shows the overall process followed throughout the research. It started out with the problem identification process. Then continue to the objective formulation for the problem identified which will shape the design and development of the approach we are proposing. Then, the experiment stage follows which is carried out to validate our proposal. Then, an evaluation section is followed.

3.2 Proposed approach

The knowledge that comes from the teacher comes entangled with the regularization effect. Proposed here is a mechanism to reduce this effect by adapting the concept of magnitude pruning. **Magnitude pruning** is setting zero values less than a certain threshold as opposed to removing the structure itself as a whole as in [87]. We call this threshold **pruning rate**. The word Pruning is adapted from the Pruning in model compression vocabulary to allow for a principled way of controlling it rather than just thresholding. In simple terms, the proposed approach is similar to putting an activation function at the softmax layer.

3.3 Intuition

Training NNs can be seen as finding the right place in the spectrum of overfitting and underfitting [12]. Regularization, as discussed in 2.3, is a solution proposed for networks that are likely to overfit [19] by constraining the network from memorizing or over-training. But in model compression the student, the focus is on smaller networks and smaller networks are by design constrained and don't have enough parameters to memorize with. But Knowledge Distillation, as described in 2 section, works by adding what is known as a Distillation Loss in the conventional training loss which serves both as a regularizer and as a categorical knowledge carrier [1]. In fact, the regularization effect of the Distillation Loss in KD is very strong [1], [32]. So, by when distilling into smaller networks, the regularization will be constraining an already constrained network. We argue that the smaller the network, the worse the regularization effect of the regularization in KD. Results in the experiment section demonstrate this claim.

3.4 Dark Knowledge Pruning Algorithm

We termed the proposed approach Dark Knowledge Pruning (DKP). In the context of NNs, pruning is removing part of a network either by setting a parameter zero or removing a structure literally from the network to exclude it from contributing to the output without

regard to whether the output is improvement or loss. But here, in this context, pruning is applied to remove the contribution of certain parts of the teacher's output which are called the *logits*. This removal of contribution is done by setting them to zero. The reason why pruning is applied or why the setting zero is termed pruning is, pruning gives a principled way of setting a parameter zero and there is an extensive amount of work on pruning, as described in 2, that can potentially be considered for extending this work.

The type of pruning applied here is what is known as **Magnitude Pruning**. It is simply set to zero for the ones with lower magnitude, absolute value. Figure 12 shows the mechanism of pruning with example numbers.

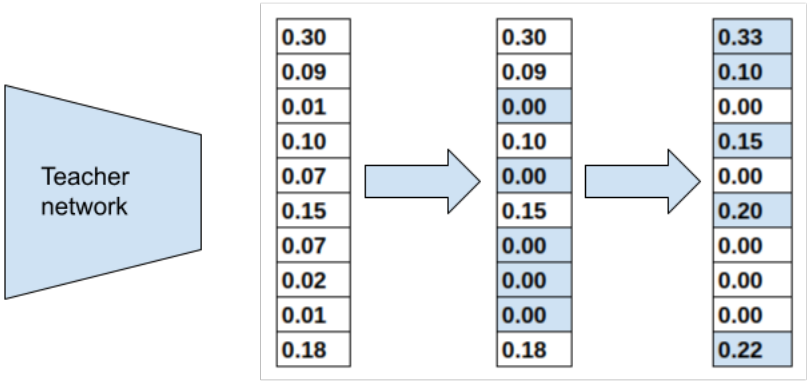

Figure 12: Dark Knowledge Pruning (DKP)

Figure 12 shows a visual demonstration of what the proposed algorithm does. Out of the ten logits from a teacher network, the smallest five of them are set to zero. The last, tampered softmax logits are then used in the same way as the conventional KD.

This research aims to apply this concept to the outputs of the teacher's knowledge. That means setting parts of the predictions(logits) to zero at different places in KD. The pruning is done after the softmax in KD. Pruning the dark knowledge simply refers to setting them to zero. The other one is Pruning Softmax, setting zero as a subset of outputs of teacher logits that are normalized already. This will disturb the normalized nature of softmax outputs. Therefore, the resulting softmax needs to be normalized again.

Certain characteristics can be checked as to the effect of this pruning. But the generic

Algorithm 1 Knowledge Distillation with Pruning Dark Knowledge. S : pruning rate

for each iteration **do**
  for each input batch x **do**
    Pass x through the network to get outputs
    Get output vectors before softmax (logits) of x
    Compute Cross Entropy Loss loss
    Compute distillation loss with S of the softmax outputs set to zero and re-normalize
    Compute loss as the weighted sum of Cross Entropy loss and Distillation loss
    Compute gradients and update parameters
  end for
end for
result = result

issues are the focus of this paper. These are the mathematical interpretation, what happens to the network while being trained, and how would the network performs after being trained against metrics such as accuracy.

Pruning rate. DKP introduces one critical hyperparameter, the pruning rate. It refers to how much of the logits to prune or set to zero. As described experiment section in 4, it plays a key role in performance. An educated way of finding the right pruning rate or learning it from data or making it dynamic is left as future work. Some high-level directions are suggested in the discussion section.

Mode. is the mode of pruning. It determines which part of the threshold to set to zero from the pruning rate on the logits. There can be many variants of these hyperparameters. Experiments on the reverse are done only on the initial run on LENET. The stochastic application of varying the pruning rate also gave a good result in CIFAR-100 with the pruning rate.

3.5 Theoretical Analysis

3.5.1 Mathematical Interpretation

The outputs of a classification network are converted into probability distributions via the softmax function. Then, the effect of post-softmax pruning can be re-formulated as:

$$p(z_i, T) = \frac{exp(z_i/T)}{\sum exp(z_j/T)} \tag{7}$$

If for a training sample from the s-th class, the classification probabilities can be denoted as $p = [p_1, p_2, ..., p_s, ..., p_C], q = [q_1, q_2, ..., q_s, ..., q_C] \in R^1C$, where p_i, q_i is the probability of the i-th class, C is the number of classes, T is the temperature of distillation, z_i represents the logit of the i-th class, then setting the s amount of the softmax outputs to zero is equivalent to adding a negative of s amount of the same terms. Then, pruning will cause the equation 1 to be:

$$Loss = \sum_{i=1}^{C} q_i * log(p_i) + \sum_{i}^{C} q_i * log(q_i/p_i) - \sum_{i=1}^{s} q_i * log(q_i/p_i) \tag{8}$$

Then the right-most term in equation 8 is the manifestation of the approach proposed in this paper. It is a KL divergence between the student and the teacher on what those classes that the teacher *thinks* are the least related to the actual class. The intuition is that, for a given data, the teacher should teach what it knows well. It forces the right-most term in 8 to become:

$$-\sum_{i}^{s} q_i * log(q_i) + \sum_{i}^{s} q_i * log(p_i) \tag{9}$$

The following properties can be observed from the equation 9:

- The rightmost term in 8 is similar to rewarding the student for diverging from what the teacher thinks is the least possible class for the given data.
- I general, Pruning the Dark Knowledge generalizes the conventional Knowledge Distillation described earlier. Setting the pruning rate to zero will yield the vanilla Knowledge Distillation.

- The first term in 9 rewards the network with the entropy of the teacher.
- The second term in 9 penalizes the network with the fidelity of the teacher on low non-target class values [32].
- By putting weight factors on the two terms, a desired control of the behavior can be achieved.

The two terms in 9 are equivalent to **rewarding the network with the entropy of the teacher** and **penalizing it with the divergence from the teacher** over less important (pruned or removed) class predictions of the teacher respectively.

3.5.2 Learning Theory Perspective

Computational or Statistical Learning Theory (CoLT) is a subfield of AI that studies the design and analysis of mathematical frameworks to understand the difficulty of learning tasks and the performance of learning algorithms [110]. We find it important to look at the proposed method from a purely theoretical perspective.

To do so we adapt the formulation used in VC theory [103] and [72]. With regard to [103], the error of a student classifier function f_s trying to learn a target classifier function f is given by:

$$R(f_s) - R(f) <= O\left(\frac{|F_s|C}{(n^\alpha)}\right) + \varepsilon_s \tag{10}$$

Where O(·) is the estimation error related to the statistical procedure for learning given the number of data points, ε_s is the approximation error, α is a constant between 1/2 and 1 that is related to the rate of learning of f_s, and n is the number of data points to learn from.

Then, by using the unification of the conventional KD with VC learning theory [72], KD can be formulated as:

$$R(f_s) - R(f_t) <= O\left(\frac{|F_s|C}{(n^\alpha)}\right) + \varepsilon_{st} \tag{11}$$

Where O(·) is the same estimation error described but ε_{st} is the approximation error of the student with respect to the teacher network while learning from a teacher [103].

Considering our proposed approach, we are introducing an artificial teacher network out of the existing teacher network. Let f_p be the teacher that we introduce by pruning the logits. Then, we can formulate it as:

$$R(f_p) - R(f) <= O\left(\frac{|F_p|C}{(n^\alpha)}\right) + \varepsilon_p \tag{12}$$

Then, distilling from the introduced teacher into the student network as opposed to the original teacher will give the equation:

$$R(f_s) - R(f_p) <= O\left(\frac{|F_s|C}{(n^\alpha)}\right) + \varepsilon_{sp} \tag{13}$$

Where O($\cdot$) is the same estimation error described but ε_{sp} is the approximation error of the student with respect to the new artificial teacher network we created.

Then, we can have the following:

$$R(f_s) - R(f_t) = R(f_s) - R(f_p) + R(f_p) - R(f_t) \tag{14}$$

This is because adding $-R(f_p) + R(f_p)$ is similar to adding 0 and thus will not make any difference. Then, we can have :

$$R(f_s) - R(f_p) + (R(f_p) - R(f_t)) = R(f_s) - R(f_p) + (0) \tag{15}$$

This is because both the estimation error and the approximation error made by the teacher and the artificial error are the same. Then from 11, we can have :

$$R(f_s) - R(f_p) <= O\left(\frac{|F_s|C}{(n^\alpha)}\right) + \varepsilon_{st} \tag{16}$$

Where ε_{st} is the approximation error of the student with respect to the teacher network while learning from an original teacher. The left-hand term in 16 is from 13 and represents the case where the student learns from the proposed teacher. This concludes our proof.

Thus, when a student learns via the proposed method, the error bound is the same as the error bound from the original teacher.

3.6 Datasets

The two representative datasets are used in literature to demonstrate Model Compression related tasks: MNIST [7] and Cifar-10[8]. Details of the datasets are discussed below.

3.6.1 MNIST

MNIST [7] is the earliest and most famous collection of images of handwritten digits. It contains 50000 training and 10000 test samples. A single sample is a handwritten 28x28 pixel image with its real label. The labels were these images were provided by actual humans who didn't write the data. The images are labeled into 10 classes matching the ten digits from 0 to 9. Due to its simplicity and accessibility, MNIST has become a standard dataset in the ML community. Many popular machine learning frameworks and libraries provide built-in functionalities to load and work with the MNIST dataset. Researchers and practitioners often use MNIST as a baseline to measure the performance of new algorithms and techniques. And also benchmarked various image classification models, including traditional ML algorithms as well as DL models like CNNs.

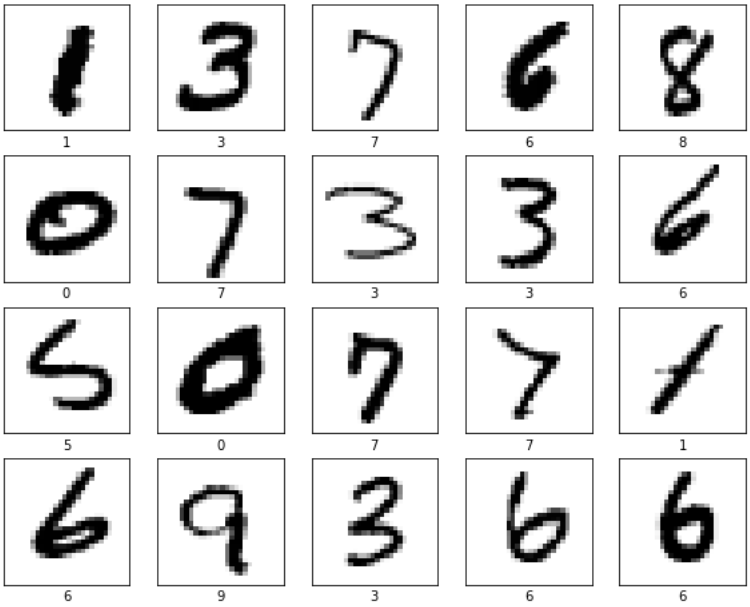

Figure 13: The famous MNIST dataset [7]

3.6.2 CIFAR 10 and CIFAR 100

CIFAR-10 [8] is another popular dataset used in the field of computer vision and machine learning. It stands for the Canadian Institute for Advanced Research, which originally released the dataset. CIFAR-10 consists of 60,000 color images, each of size 32x32 pixels, divided into 10 different classes. The dataset contains a balanced distribution of 6,000 images per class. The CIFAR-10 dataset covers a diverse range of object categories, including animals, vehicles, and everyday objects. The goal of using CIFAR-10 is to develop models that can accurately classify these images into their respective classes. Each training batch contains exactly 5000 images. The 10 different classes are airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

Similar to MNIST, CIFAR-10 is widely used as a benchmark dataset for evaluating and comparing image classification algorithms. It presents a more challenging task compared to MNIST due to the smaller image size and the presence of color, requiring models to learn more complex visual features. CIFAR-10 has become a standard dataset in the ML community and serves as a stepping stone for exploring more complex and real-world image classification challenges. It's popularity and widespread usage make it an essential resource for researchers, educators, and practitioners working in the field of computer vision.

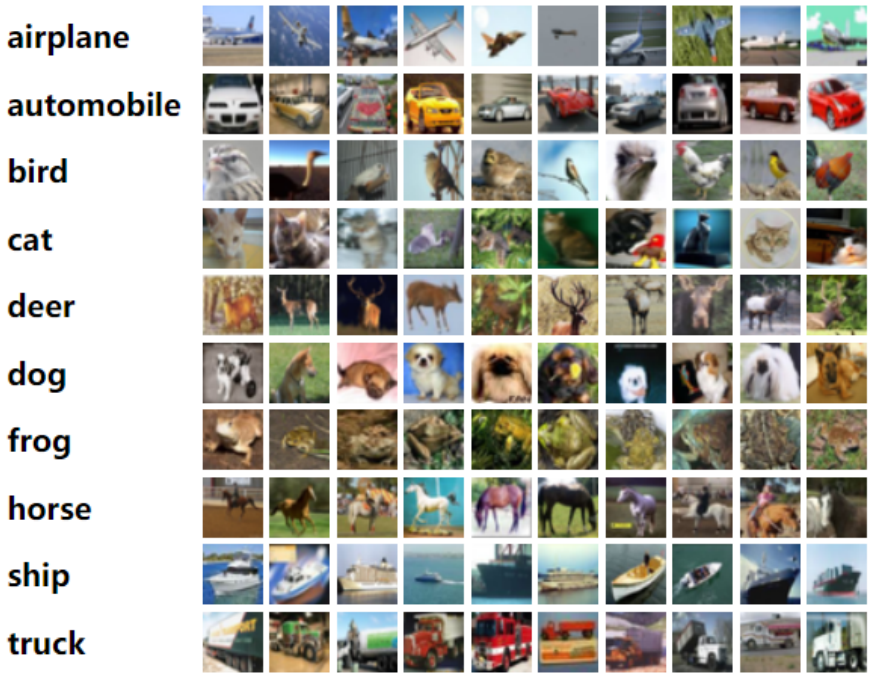

Figure 14: Cifar-10 dataset [8]

CIFAR 100 [8] is the same as CIFAR 10 but it has 100 classes. CIFAR-100 is a dataset of 60,000 32x32 color images, divided into 50,000 training images and 10,000 test images. The dataset contains 100 classes, each with 600 images. The classes are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs). As the task is more difficult than CIFAR 10, it is used as a benchmark for recent methods.

In summary, MNIST and CIFAR-10 have served as pivotal benchmarks for image classification tasks in computer vision. They have played a crucial role in advancing the field by enabling comparisons, promoting reproducibility, and encouraging the development of new algorithms and techniques. These datasets continue to be valuable resources for researchers and practitioners working in the domain of computer vision and machine learning.

Name	Size	Type	Classes	Remark
MNIST	60000	Image	10	28x28 grayscale handwritten digits [7]
CIFAR 10	60000	Image	10	32x32 color images [8]
CIFAR 100	60000	Image	100	32x32 color images [8]

Table 4: Summary of datasets used

3.7 Models

As discussed in the background section, there are prominent models that serve as a benchmark. Since these models are also milestones of the progress of the field of AI, they usually have specific names like LeNet, Alexnet, and Resnet50. So, this research will use these benchmark models: LeNet [25], RESNET [42], and their variants are selected as they are commonly used in model compression and optimization literature.

3.8 Evaluation metric

Unlike in the case of a compressed file, a compressed model is not expected to reproduce the exact contents before compression, but rather to perform just as well with a smaller file size despite differences in structure. The overall aim of the thesis focuses on the conventional KD which requires the network size to be fixed before the compression begins. This is not the case in other compression methods such as Pruning or Quantization. This makes taking compression rate as a metric infeasible. Therefore, one metric that can work is the generalization capability of the student network. To measure it, accuracy is the metric used which is the ratio of the data points the network got right divided by the points it got wrong converted into a percentage. Therefore, the primary performance measure that is being monitored is the accuracy of the test set. This choice of metric aligns with existing literature such as [6] and [34].

4 Experiments

This section discusses details of the experiments done on the selected models and datasets to demonstrate the effectiveness of the proposed approach. It contains four key parts: the general outline, which describes a high-level view of all the experiments done and their objectives, the tools used in the experiment, a setup section which describes the hyperparameters considered, and lastly, the last part presents the results of the experiments.

4.1 General Outline

To demonstrate the effectiveness of the proposed approach, three sets of experiments are carried out on the selected model-dataset pairs. The first is on LENET-MNIST. It is done to make a first run across a sweep of hyperparameters that were described above. The second and third experiments are on RESNET-CIFAR10 and RESNET-CIFAR100. They are used repeatedly used as a benchmark for similar works in literature [34], [71]. The objectives of these experiments are described in the following table.

Experiment	**Objective**
LENET-MNIST	A range of initial runs on MLP and custom-designed LENET variants to get first insights about the proposed method DKP.
RESNET 50 into 18 on CIFAR-10	Both models from Pytorch Standard Library [111]. Trained on CIFAR-10 with selected hyperparameters from the initial run on LeNet.
RESNET 56 into 8 on CIFAR-100	Smallest ResNet family (ResNet 8) with reported results to demonstrate the efficacy of DKP. [71]

Table 5: Outline of experiments done on LeNEt and ResNET

4.2 Experimental setup and tools

The following table organizes the tools used for the experiment throughout the study.

Type	Selected	Description
Cloud Platform	Google Coalb Pro	Monthly subscription
Programming	Python	A language preferred for DL
Framework	Pytorch	Python DL framework
Library Pruning	Pytorch prune	Pytorch Pruning library
Numpy	Numerical computation	Python-based library
Visualization/Tracking	Wandb	Web-based ML experiment tracking tool
Computation hardware	NVIDEA SMI	Tesla T4
Local Platform	Jupyter-notebook	For small-scale experiments
Monitoring tool	Tensorboard	A Monitoring tool

Table 6: Hardware and Software tools to be used

Table 6 shows the tools used in this research. The table includes both hardware and software tools that are free as well as premium ones.

Setup of hyperparameters. In training Neural Networks, a hyperparameter is the decision of the designer or the programmer. Since the proposed method, Dark Knowledge Pruning, tampers with the loss function directly as described in 3, a few hyperparameters are swept across to find out experimentally how they would perform in the initial run. The initial experiments are done on LENET and MNIST as they have been used for proof of concept in literature repeatedly [40], [5]. Also, doing hyperparameters sweep across RENSET and

CIFAR is computationally expensive and beyond the budget scope of this thesis. Then, from these results, a couple of them will be selected for further experiments on the bigger datasets and models, on the RESNET family on CIFAR datasets. To do so learning rate, pruning rate, pruning type, optimizer type, activation function, and temperature of distillation [1] are selected as a set of hyper-parameters.

- **Teacher networks**. The first thing needed for Knowledge Distillation (KD) is a teaching model that is trained. Here, the teacher models are LeNet-5 and ResnNet50. Both are types of CNNs with the following details presented in the table.

Type	**LeNET-5**	**ResNet56**
Number of parameters in the model	60,000	23 million
Accuracy	96%	95%
Optimizer	SGD	Aadam
Loss	CrossEntropy	CrossEntropy

Table 7: Details of teacher networks

- **Pruning rate** is how much of the logits from the teacher to prune(to set zero). This is an obvious property to keep track of in the hyperparameters sweep set. This is also where the conventional Knowledge Distillation (KD) [1] is embedded as a case when this value is zero.

- **Optimizer** Optimizers such as Stochastic Gradient Descent actually do the learning in Deep Learning (DL) by updating the parameters in the network [10].

- **Learning rate** controls how much to update a single parameter at a time. Since the behavior of the optimization which depends on the loss function that has been

tampered with the proposed method is highly dependent on the learning rate, it is selected for the sweep set [10].

- **Pruning type** is the question of whether or not to do the pruning in reverse or forward direction for a pruning rate. Again, pruning here is not the same as pruning the weights [40] as presented in literature review, its on the logits. For example, if the pruning rate is 0.3, then the pruning type can be whether to keep the top or bottom 0.3 logits from the teacher outputs. The reversed mode is not run beyond the LENET MNIST case, but its results on that can spark questions worth looking into.

- **Temperature** is the hyperparamter introduced by the authors of [1]. it controls the softness of the logits. As the tampering is applied directly to it, sweeping across different values can also unravel interesting findings.

- **Activation functions** can play a role in both ways of learning(forward pass and backpropagation). The weights are updated through them. Thus, keeping track of them can also show worthy results [46].

- **Epochs** The teacher was trained for 100 epochs but for the students 200 were chosen to observe their convergence speed as it can be expected to have a slow one.

Type	**Description**	**Values**
Learning rate	controls the rate of update	[0.00001,0.0001,.001,.01,.1]
pruning rate	how much to prune from the logits	[0, .1,.2, ... , .9]
pruning type	does higher/lower magnitude pruning	['forward','reverse']
Temperature	control softness of logits	[5,10,15,20,25]
Activation	activation functions	['RELU', 'TANH']

Table 8: Hyper-parameter details for initial experiment

4.3 Results

This section presents the results from the runs of the three sets of experiments. Only the important results with regard to the evaluation metrics we set out in 3 are presented in detail. The results are mostly presented in table form on par with similar works in literature such as [71], [34], [76], etc. But in places it makes sense to, other types of visualizations are presented such as time series graphs for accuracy and loss.

Initially the results from the runs of LENET and Multi Layer Perceptron on MNIST are presented. These are used for the initial inspection of what can the proposed method do. Then, the result from the RESNET family is presented along with reported results in the literature.

4.3.1 Experiments on LENET and Multi Layer Perceptron on MNIST

The two images below are the results of accuracy and loss across all the runs. The two images help see the general performance of the network comparison.

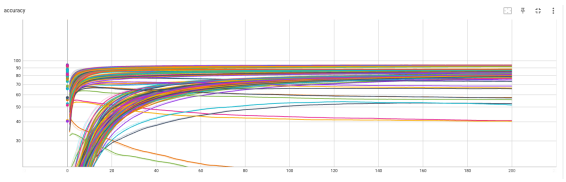

Figure 15: Accuracy

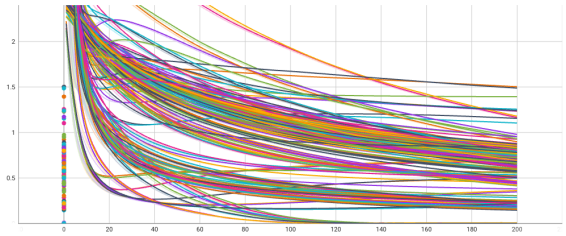

Figure 16: Loss

The above two images show the accuracy and loss values across a sweep of runs on hyperparameters described earlier in 8. There are about 1000 runs as a result of permutations from the hyperparameters. This result is meant to show a gist of what the proposed approach is doing experimentally. A broader view of the experiments can also be visualized in a scatter.

Unlike the above time series plot for accuracy and loss, these plots allow us to see all the metrics we care about at the same time.

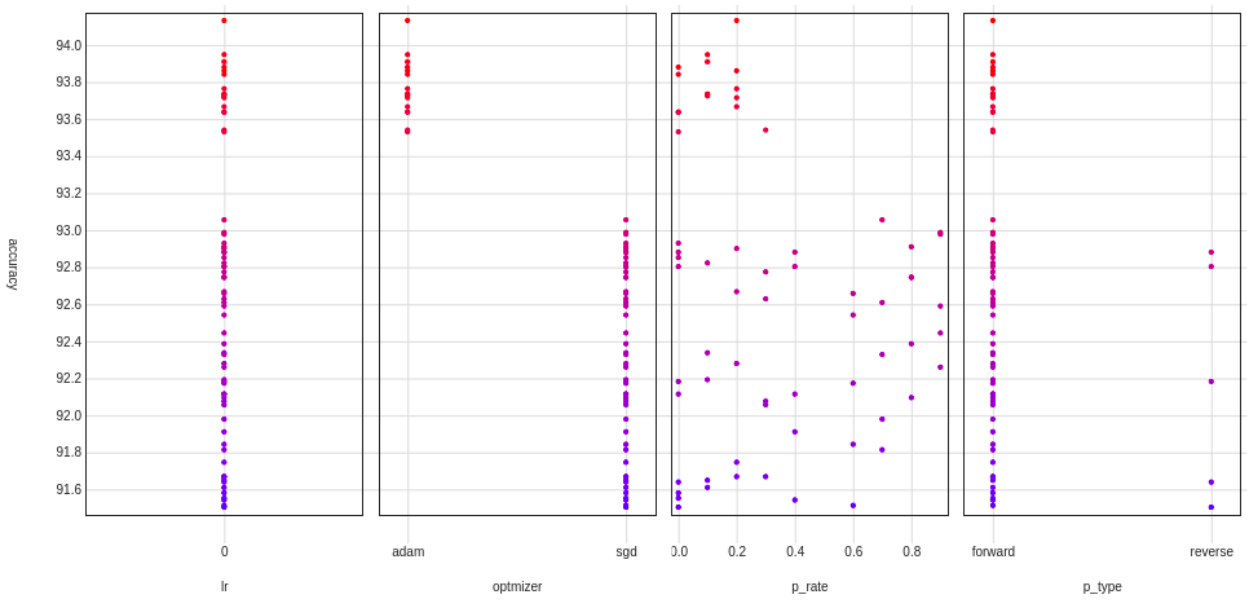

Figure 17: Top performer results more than 90% accuracy

Figure 17 shows a scatter plot of hyper-parameters that are top-performing (more than 90% accuracy). The second plot shows that the best results are obtained from the Adam optimizer runs as opposed to the SGD runs. The third scatter plot shows results that are most similar in accuracy across the different pruning rates from 0.1 up to 0.9 with a 0.1 increase. But it can also be seen that slight pruning rates, such as 0.2 and 0.3, can achieve better performances Knowledge Distillation (KD). This is expected as the proposed method generalized the conventional KD. The last scatter plot shows whether or not the DKP was done forward or reverse. The reverse indicates thresholding the smallest logits rather than the big ones. It can be seen that reversing almost never gives good results. This is expected as it damages the labels.

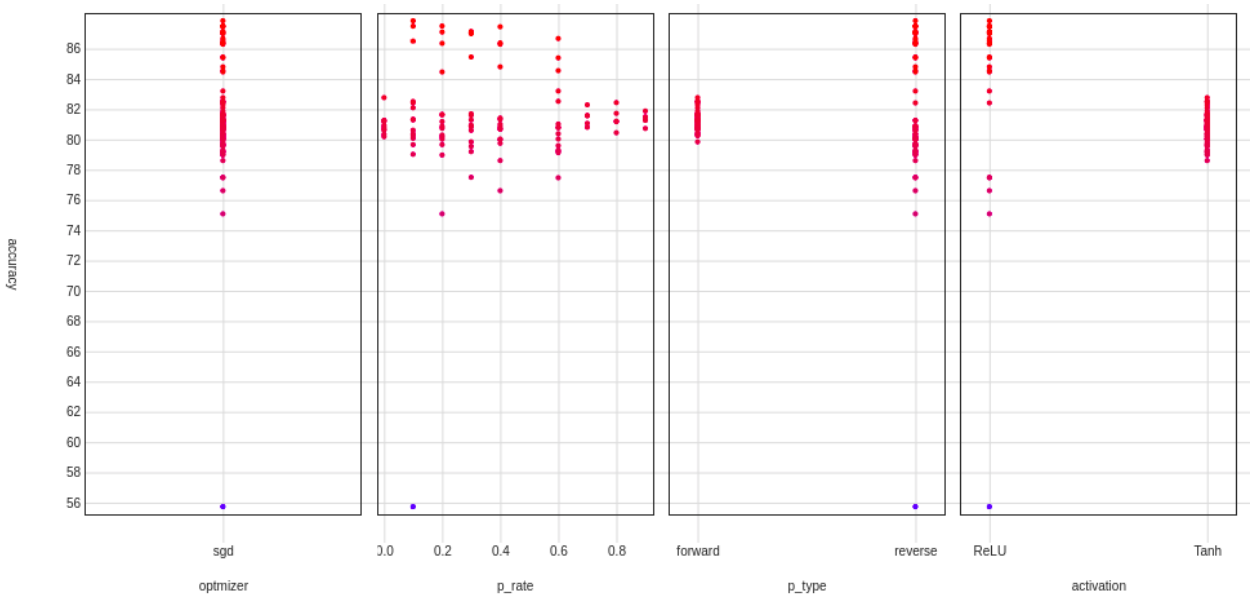

Figure 18: Worst performer results from 50-90% accuracy

Figure 18 shows the scatter plot of the results below 90% of over the runs. Worst-case performances in the second plot are significantly visible in the reverse pruning runs. The worst results are the reverse pruning run with a pruning rate of more than 0.7. This means that making zero at least the top 70% predictions of the teacher model can significantly drop a model's performance. And all of them are using the SGD optimizer. This shows that the Adam optimizer is resisting alterations caused by pruning the dark knowledge better than the SGD. These results in general show the value of Dark Knowledge as tampering with them is shown to potentially damage the teacher's performance. This property of student degradation can also be a desirable one.

Approach	**Best**	**Worst**
KD	60000 100%	96.71 T 15 p_rate 0.8, T 20, 96.77%
DKP	p_rate: 0.2, 0.3, optimizer:Adam, T:15%	96.35 T 20 p_rate 0.8, T 20, 96.4%

Table 9: Best and Worse performance results of the initial run.

Table 11 shows the hyperparameters that gave the best and worst results from the runs across hyperparameters when distilling LENET into a two-layer MLP. T is temperature and p_rate is the pruning rate applied.

For extreme compression, two simple experiments were done on MLP students with 1% size of the original LENET size and custom-designed variants of LENET. The former were trained with a learning rate of 0.0001, a Cross-Entropy loss, and an Adam optimizer for 100 epochs. The following table summarizes the results in which the teacher is the original LeNET and the student models are different-sized MLP networks.

Pruning rate	0.0 KD	0.5	0.6	0.7	0.8	0.9	1 (Hard Label)
Accuracy	9.43%	9.9%	10.%	10.%	11.176 %	11.76 %	11.76 %

Table 10: Average accuracy table for custom-designed MLP with 1% size on selected hyper-parameters. p_rate is the pruning rate applied.

Obviously, these are examples of bad-performing models, but this is due to their extremely small size, 1% of LENET, compared to the task trained on. But as it can be seen the higher the pruning rate the better, the distillation compared to the baseline KD.

The following table summarizes the results in which the teacher is the original LeNet and the students are small but customized versions of the model.

Model Details	**Model Size**	**Size %**	**Baseline KD**	**Proposed**
LENET self	60000	100%	96.71 T 15	p_rate 0.8, T 20, 96.77%
5 layer Conv	9170	15%	96.35 T 20	p_rate 0.8, T 20, 96.4%
3 Layer Conv	4950	8%	96.45 T 20	p_rate 0.8, T 20, 96.50%

Table 11: Average accuracy table from LENET to smaller versions

The above table shows the accuracy table for custom-designed compressed LeNet versions learning from the original LeNet on MNIST on selected hyperparameters:T is temperature and p_rate is the pruning rate applied. This is the best performing for both KD and the proposed method. This experiment shows that, as the network size gets smaller, the size performance gap between the proposed and the conventional KD increases. But this experiment is inconclusive for= the success of the proposed approach, it only shows the potential so far.

In summary, the above experiments on LENET and MNIST were done as an initial run of the proposed method for the first time ever to get important observations. These observations can be made from the experiments :

- There are cases where the student trained with a certain pruning rate and mechanism will perform better, worse, or equal to the conventional KD method.
- Optimizers Adam was better across the experiments observed here.
- Worst-case performances in the second plot are significantly visible in the reverse pruning runs.
- Smaller learning rates (such as 0.00001) do better in terms of accuracy
- ReLU is the best-performing activation function in the experiment done here for both pruned and KD runs. Suggesting we keep RELU fixed for the next experimentation.
- Relatively high temperatures (such as 10 and 15 on the 10 class MNIST) give good performance.

These served as suggestions for our next experiments. But they are themselves questions that have to be answered potentially by another research.

4.3.2 Experiments on ResNET architectures and CIFAR datasets

Another experiment was done on two ResNET networks. The knowledge was being transferred from a ResNET50 into ResNET18. Since doing an extensive amount of experiments is

expensive and infeasible within the budget scope of this thesis, the experiment is done once but by making use of the best-performing hyperparameters from the LeNet experiments above. The figure below shows the accuracy of two compressed models one trained with conventional KD and the other with Pruned Dark Knowledge.

Model	**Accuracy**	**Details of the models**
KD	74%	pruning rate= 0.0, temperature : 15 , optimizer: adam
Prunned	**77%**	p_rate= 0.2, T : 10 , optimizer: adam

Table 12: Accuracy table for compressed ResNET18 models via the conventional KD and the proposed method under their best-performing hyperparameters. T : Temperature of distillation, p$_r ate$: $prunningrate$

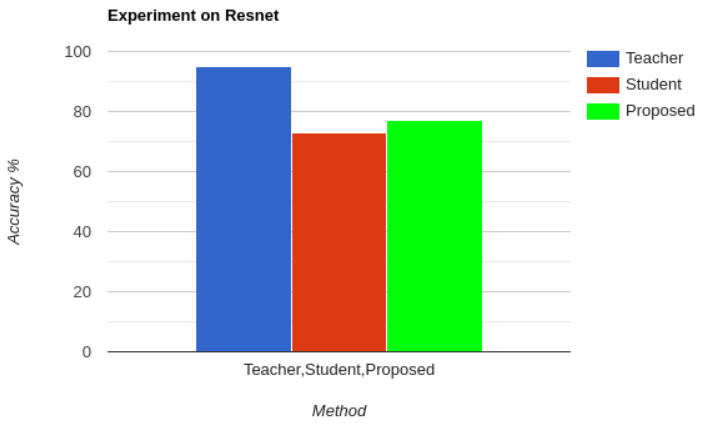

Figure 19: Experiment on Resnet

The above graph shows the result of a simple run of two ReNET models on CIFAR 10 being distilled from a teacher. The graph shows the performance improvement by the proposed method.

Type	Dataset	Naive	Teacher	Student	Improved
Decoupled KD [34]	CIFAR 100	ResNet-20 69.06%	ResNet-56-72.34%	70.66%	71.97%
	CIFAR 100	ResNet-110-74.31%	ResNet-32 71.14%	73.08%	74.11%
	CIFAR 100	ResNet-32-79.42%	ResNet-8 72.50%	73.33%	76.32%
Professor Network [24]	Cifar 100	WResnet-16 78.8%	PreResnet-50 71.18%	71.76%	71.93%
Teacher-Free KD Reg [106]	CIFAR-100	ResNet-50	ResNet-18 75.87%	77.19%	77.36%
	CIFAR-10	ResNet-18 95.12%	MobileNetV2: 90.98%	91.69%	95.71%
	CIFAR-10	ResNeXt29: 95.76%	ResNet18: 95.12%	95.80%	96.49%
Teacher Assistant [6]	CIFAR-10	ResNet8 88.52%	ResNet	88.65%	88.98%
	CIFAR-100	ResNet8	ResNet 110	61.41%	61.8%
Proposed method	CIFAR-10	ResNet	ResNet-50 96%	ResNet-18 74.0%	**77.1**
	CIFAR-100	ResNet-8	ResNet-56 71%	ResNet-8 61.41%	**63.62**%

Table 13: Performance comparison across related work

The table above presents the results of experiments done on RESNET architecture on the two CIFAR datasets along with results in literature in similar works.

The result on CIFAR-100 described in the last row of the table shows the fruitfulness of the approach presented. As described in related works earlier, the work in [71] is the closest related work in terms of studying the size gap between a student and the teacher network. The presented result in the table is from a [6] work that studied the inefficacy of KD when there is a high gap between the teacher and student. To the best of our knowledge, this is the only work that studied this gap. Their concern was to enable teachers to get bigger and bigger for a fixed student to choose a better one of them. The researchers reported results on different datasets and model pairs. Here in the table (row one) is their result on CIFAR 100. To achieve this they required an additional assistant network that tries to fill the size gap between the teacher and student. The result from DKP does not need a teacher assistant network to achieve a 62% accuracy to the same model they distilled to. Therefore, is justified

to say the result proposed in this thesis shows competed against the current state-of-the-art report on ReNET-8 without the need for an additional network in the middle.

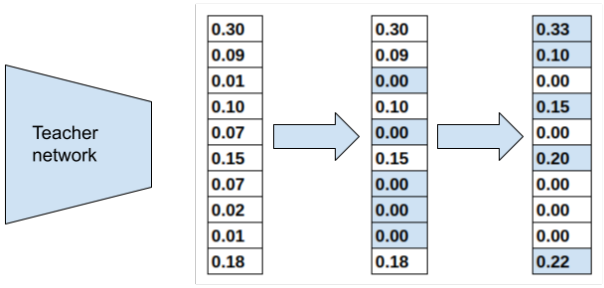

Figure 20: Proposed DKP

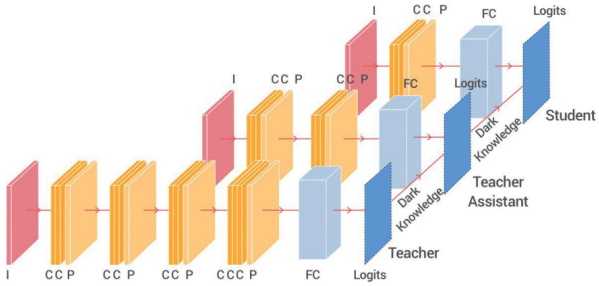

Figure 21: Distillation with teacher assistant [6]

Images 20 and 21 are presented side by side as a way to visualize how the proposed approach (left) compares to existing work on the large size gap between a teacher and student (right).

What is the cost of the proposed method? The proposed method's space complexity as compared to the naive KD is O(1) as it only needs to store the pruning rate. As compared to the baseline KD, the proposed method requires more computation and thus more computation time and resources due to the extra pruning step required. But as compared to similar works such as [6] as can be seen in 21, it takes significantly less time and computation resources as there is no network that sits between the teacher and the student. In terms of Dark Knowledge, the proposed teacher introduces a new artificial teacher that has the same performance as the original teacher because only the small logits are pruned. So, the student is still learning from a teacher of the same performance. This is on par with the privileged information formulation of KD [103], [72].

4.4 Explanation

In this section, we discuss the question *What is it about reducing the regularization that helped the students perform well?*.

- The added the term $\sum_{i=1}^{C} q_i * log(q_i/p_i)$ in 8, which is always positive and reduces the regularization in KD, ensures the divergence of the student from the lower predictions of the teacher, and not from the highest predictions of the teacher.

- Fidelity is inversely correlated with the regularization in KD. Less fidelity makes optimization hard [32]. Although fidelity does not necessarily imply generalization [32], in the case of the extreme compression experiment seen below, it is correlated with accuracy. By applying the proposed method, we are increasing the fidelity of the student artificially.

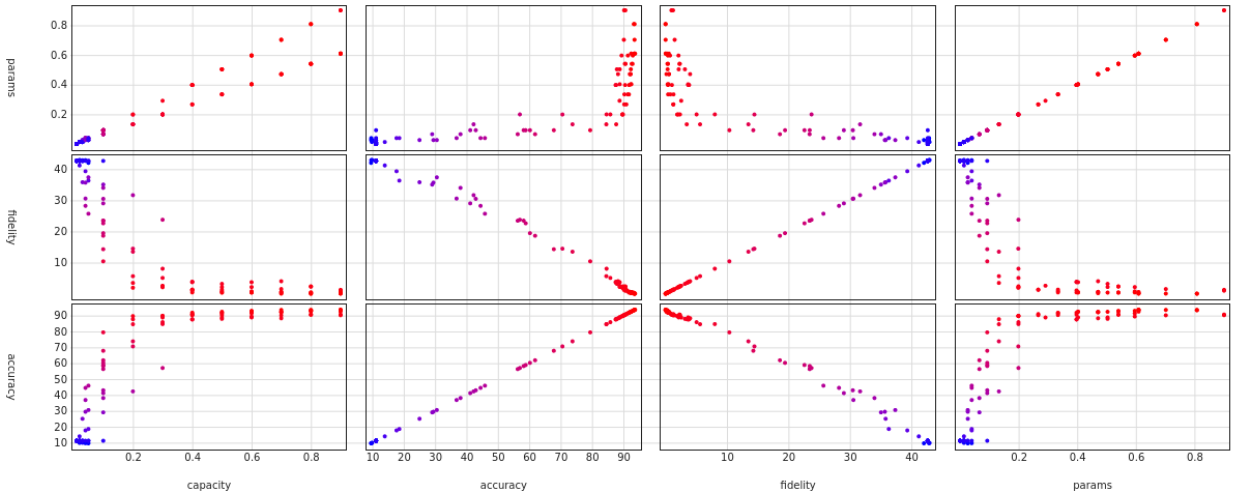

Figure 22: Fidelity in small networks.

The accuracy versus fidelity graph in figure 22 is from an experiment on KD as students' capacity gets smaller. As it can be seen the fidelity of the student is directly correlated with the accuracy of the teacher especially as the capacity drops below 10%.

- By maximizing $\sum_{i=1}^{C} q_i * log(q_i/p_i)$, the student is encouraged to diverge from whatever the teacher predicts as less important.

- Intuitively, regularization is a solution proposed for networks that are likely to overfit [19] by constraining the network from memorizing or over-training. But in model compression, the focus is on smaller networks which are by design constrained and don't have enough parameters to memorize with. But the regularization effect of the Distillation Loss in KD is very strong [1], [32]. So, distilling into smaller networks with the full regularization effect will be constraining an already constrained small network.

- The last result on CIFAR-100 and RESNET-8 is obtained by stochastically updating it changing the pruning rate from 0.6 and 0.7. This is equivalent to mimicking multiple teachers that have the same performance as the original teacher. This is because, at every epoch, the student gets different predictions as if it's them from different teachers. This is similar to [37]

- Theoretical guarantee formulated in 3.5.2

4.5 Findings

The following are key findings from the experiments done above:

- When a DKP is applied on networks when compressing networks via KD, the student can perform as good, worse than, and better than the conventional KD. This was expected as mentioned in the methodology section. the proposed study generalizes the conventional KD.

- Performance can be enhanced by finding the right pruning rate for a given problem. For example, in the MLP experiment above, some students outperformed the KD baseline method [1] with pruning rates 0.2 and 0.3 in the initial experiment, and in the last experiments on RESNET-CIFAR.

- The cases where the proposed method outperformed the conventional KD demonstrate that via experiment where a less regularized but better knowledge transfer than a relatively higher magnitude regularizer.

- According to the small experiments done on LENET -MNIST on distilling into very small students, including 1%, the non-distillation result was better followed by the proposed approach, then KD. The experiments were the results of multiple runs which gave different results and averages were presented.

- With regard to a similar experiment done from LENET into Multi Layer Perceptron, fidelity, the teacher-student agreement becomes highly correlated as capacity drops below 10% percent in distilling 22.

- As seen in the results section, there are setting that degrade over time to a point they are worse than a random classifier. These are all cases when the DKP is applied in the reversed setting which was tried in LENET-MNIST initial experiment.

- Experiment on CIFAR-100 outperformed the current reported smallest RESNET family on the dataset [6] where the authors employed an extra network to assist the student network.

4.6 Discussion

The experiments conducted in this study showcase the effectiveness of the proposed approach, which introduces a mechanism to reduce the effect of regularization. in KD.

The cases where a performance enhancement is observed, show that by finding the right pruning rate, the baseline KD can be improved. This implies also that the proposed approach can unlock the potential of the teacher untapped by the conventional approach. It is also fair to conclude that the new works that outperform the conventional KD are less regularized while performing well. This shows that they are either benefiting from purified information from the teacher, or they are leveraging the reduced stress from less regularization. Either way, this mechanism can potentially enable the disentanglement of regularization and Dark Knowledge components within the distillation loss. It can even potentially assist in quantifying Dark Knowledge.

As seen demonstrated in the results section in 18, reversing the pruning pattern can deteriorate a student's performance worse than the student network. This is an interesting property that needs further research to be leveraged for something useful. For example, due to the emergence of Data Free Knowledge Distillation, the security of models is becoming an issue in its own right [112]. And therefore, the ability to deteriorate a student network at will can be desirable. If a network is somehow made to output a distribution that is similar to a reverse-pruned network, it can translate to the student being trained on that network will be degraded. The question of *How to train a network in that way ?* is then a research question to be explored. Assuming it is possible to do so, then by utilizing this approach, it becomes possible to deteriorate the performance of a student network as desired. This level of control empowers researchers and practitioners to manipulate the learning process and optimize the student model based on specific requirements or objectives.

Moreover, the proposed method allows a student network to benefit from purified knowledge that is derived from a network with less intense regularization. This implies that the student can acquire valuable insights and information that might not have been as strongly emphasized in the original network's regularization process.

One noteworthy aspect of the proposed approach is that it operates externally to the network itself. This model-agnostic nature implies that the method can be applied to different types of networks, irrespective of their architecture or underlying design principles. This flexibility and adaptability make the proposed approach a versatile and widely applicable solution in the field of KD.

The following research questions were posed:

RQ1 How can the impact of the regularization be reduced from the distillation loss in Knowledge Distillation (KD)? A novel approach to reduce the impact of regularization has been proposed as described in 3. The proposed approach which we termed Dark Knowledge Pruning is applied to the loss function to reduce regularization. The proposed approach generalizes the baseline KD as it equals it when the pruning rate is set to zero.

RQ2 What happens to the performance of different-sized student networks if Pruning is applied to the teacher's predictions in the Knowledge Distillation framework? DKP is applied, initially, on LENET as a teacher and on a custom designed MLP across a sweep of hyperparameters such as pruning rate, distillation temperature [1], optimizer, activation functions, etc. We find that the usual hyperparameters such as learning rate, activation functions, and. The introduced hyperparameters' pruning rate and mode of pruning are key to the proposed algorithm.

RQ3 How do different hyper-parameters perform when Dark Knowledge Pruning is applied to a student? There are three types of hyperparameters. First, there are the usual ones such as learning rate, activation functions, and optimizers preserved their performance and behavior [10]. Second, there are hyperparameters introduced by the distillation mechanism [1]. These are mainly the temperature and the distillation weight. The latter was kept fixed at 0.5. But the results on temperature kept varying across experiments. This can be explored in future research. The third set of hyperparameters is the pruning rate and the pruning mode. They are introduced by the

proposed approach DKP. The former is the magnitude threshold to set to zero, and the latter determines which part of the threshold to set to zero. The latter only has two values 'forward' and 'reverse'. 'reverse' mode was experimented on the initial run and highly deteriorates student performance. It misleads the network. The rest of the experiments focused on the 'forward' mode. These imply smaller magnitude weights will be set to zero. As seen in the results and described in the discussion section, there can be an improvement, worsen, or equal the performance of the distillation across these hyperparameters mainly across the pruning rate.

How to choose pruning rate ? In general, the pruning rate is to be chosen according to the problem at hand. Therefore, it can be considered as a hyperparameter introduced by the proposed mechanism. The following are suggestions for an educated way of finding the right pruning rate. Some of them are from intuitions learned from the above experiments others are possible future directions.

- The smaller the student network, the higher the pruning rate.
- Dynamic pruning rate that depends on the data, behavior, epoch, or some other characteristic. For example, the best results from the above experiment were found by applying increasing it with epoch. Representation similarities such as Centred Kernel Alignment can be considered [113].
- Reinforcement Learning based on choice of rate.
- Stochastic choice of the pruning rate as well as different modes of pruning.

5 Conclusion

Since the exact number of parameters and their arrangements is still not known for a given problem, the current approach in NNs is to design big models that guarantee performance and then optimize [20]. Their size comes with challenges such as longer training, inference, and iteration time, difficulty to deploy on resource-constrained and edge machines, and economic and environmental impact [28]. Model Compression is a way of reducing the size of models with the smallest performance loss possible [31]. The necessity of model compression algorithms is increasing in parallel with the advancement and complexity of models.

Knowledge Distillation is a model compression method that works by transferring the knowledge of a bigger model into a student model. It is an effective, flexible, and model agnostic [1] method, but it doesn't give high compression rates. It doesn't give a good performance when the gap between the student and the teacher is too big. This phenomenon is highly neglected in the literature.

This thesis proposed a novel approach termed as DKP to reduce the effect of the regularization that is entangled with the Dark Knowledge in KD framework. Experiments were done on benchmark datasets and models show fruitful results. A sweep of runs was conducted on LENET-MNIST to observe initial run experiments, then on a family of ResNET networks on the CIFAR-10 and CIFAR-100 datasets, which are considered an even more common dataset-model pair is used as a benchmark to this day for validation. The proposed approach outperformed the currently reported result in [6] on RESNET-8 and CIFAR-100 datasets by 0.58% from 61.82% to 62.4% without a need for an assistant model to sit between the student and the teacher to mediate the gap. Beyond generalizing the conventional KD and bridging Pruning and KD in an entirely new perspective, the proposed approach improves the understanding of the knowledge transfer in KD, allows the use of features of KD separately at will, helps train better-compressed models and pause a variety of further questions for further research for studies concerned in the efficiency and knowledge transfer domains.

5.1 Future Work

The idea proposed, the experiments done, and the results observed, pose a lot of open research-worthy questions and ideas that can serve to guide future research directions. Most of the future works are left as so due to computing power limitations.

The following are recommended directions for exploration.

- **Variants**. How will different variants of KD behave in light of the proposed idea?
- Does applying the concept for the pre-softmax results change observations made? This question makes sense because the proposed method is applied after the softmax.
- **Post the training properties** of a network in the proposed way, further questions can be asked. For example, how important characteristics of a student network change as a result of the way it was trained is one such question. Here we can study essential characteristics such as explainability, security, etc.
- **On different network architectures** The experiments done here are only on CNN and MLP variants. But there are other state-of-the-art algorithms out there such as transformer networks.
- **Data** The experiments done here are on small-scale datasets namely MNIST and CIFAR. While they are in fact standard and have been used to demonstrate concepts, they are considered small-scale datasets by today's standards. Even, IMAGENET [11], the large-scale image dataset served as a benchmark for the object classification tasks in Artificial Intelligence (AI) research literature over 10 years since 2012 [20] is not considered large scale anymore. Therefore, the idea can be explored on large-scale datasets.

Besides scale, different kinds of questions can be explored with related data. For example, by studying the proposed idea in relation to a dynamic whether or not, and how the learning will behave under certain settings, as shown 4, the method can yield better results via dynamic approaches: by assigning the settings such as pruning rate dynamically.

References

[1] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv*, vol. abs/1503.02531, 2015.

[2] L. Bernstein, A. Sludds, R. Hamerly, V. Sze, J. S. Emer, and D. Englund, "Freely scalable and reconfigurable optical hardware for deep learning," *Scientific Reports*, vol. 11, 2020.

[3] R. Pramoditha, "The concept of artificial neurons (perceptrons) in neural networks," *Towards Data Science*, 2022. [Online]. Available: https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc

[4] X. Ding, Y. Wang, Z. Xu, Z. J. Wang, and W. J. Welch, "Distilling and transferring knowledge via cgan-generated samples for image classification and regression," *Expert Systems with Applications*, vol. 213, p. 119060, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422020784

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *arXiv: Computer Vision and Pattern Recognition*, 2016.

[6] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *AAAI Conference on Artificial Intelligence*, 2019.

[7] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[8] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[9] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson Education, 2020.

[10] I. J. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2016.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84 – 90, 2012.

[12] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[13] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[14] K. Hornik, M. B. Stinchcombe, and H. L. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[15] Y. Bengio, 2009.

[16] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[18] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1706.00384*, 2018.

[19] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *NIPS*, 1989.

[20] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[21] S. L. Kukreja, J. Löfberg, and M. J. Brenner, "A least absolute shrinkage and selection operator (lasso) for nonlinear system identification," *IFAC proceedings volumes*, vol. 39, no. 1, pp. 814–819, 2006.

[22] S. Matet, L. Rosasco, S. Villa, and B. L. Vu, "Don't relax: early stopping for convex regularization," *arXiv preprint arXiv:1707.05422*, 2017.

[23] L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," in *2018 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2018, pp. 1542–1547.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[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] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[28] D. Patterson, J. Gonzalez, Q. V. Le, C. Liang, L.-M. Munguía, D. Rothchild, D. R. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," *ArXiv*, vol. abs/2104.10350, 2021.

[29] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[30] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 8, pp. 1553–1565, 2014.

[31] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *KDD '06*, 2006.

[32] S. Stanton, P. Izmailov, P. Kirichenko, A. A. Alemi, and A. G. Wilson, "Does knowledge distillation really work?" in *Neural Information Processing Systems*, 2021.

[33] T. Choudhary, V. K. Mishra, A. Goswami, and S. Jagannathan, "A comprehensive survey on model compression and acceleration," *Artificial Intelligence Review*, pp. 1–43, 2020.

[34] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 943–11 952, 2022.

[35] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, "Revisiting knowledge distillation via label smoothing regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3903–3911.

[36] D. Bang, J. Lee, and H. Shim, "Distilling from professors: Enhancing the knowledge distillation of teachers," *Information Sciences*, vol. 576, pp. 743–755, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521008203

[37] B. B. Sau and V. N. Balasubramanian, "Deep model compression: Distilling knowledge from noisy teachers," *arXiv preprint arXiv:1610.09650*, 2016.

[38] M. Phuong and C. H. Lampert, "Towards understanding knowledge distillation," *ArXiv*, vol. abs/2105.13093, 2019.

[39] T. Huang, S. You, F. Wang, C. Qian, and C. Xu, "Knowledge distillation from a stronger teacher," *ArXiv*, vol. abs/2205.10536, 2022.

[40] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[41] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[43] D. E. Rumelhart and J. L. McClelland, *Learning Internal Representations by Error Propagation*, 1987, pp. 318–362.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[46] J. L. McClelland, D. E. Rumelhart, P. R. Group *et al.*, *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*. MIT press, 1987, vol. 2.

[47] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[49] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv: Learning*, 2019.

[50] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

[51] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *NIPS*, 1988.

[52] R. Reed, "Pruning algorithms-a survey," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[53] S.-K. Yeom, P. Seegerer, S. Lapuschkin, S. Wiedemann, K.-R. Müller, and W. Samek, "Pruning by explaining: A novel criterion for deep neural network pruning," *ArXiv*, vol. abs/1912.08881, 2021.

[54]

[55]

[56] Y. Wang, X. Zhang, X. Hu, B. Zhang, and H. Su, "Dynamic network pruning with interpretable layerwise channel selection," in *AAAI*, 2020.

[57] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1398–1406, 2017.

[58]

[59] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," *ArXiv*, vol. abs/1808.06866, 2018.

[60] E. Malach, G. Yehudai, S. Shalev-Shwartz, and O. Shamir, "Proving the lottery ticket hypothesis: Pruning is all you need," in *ICML*, 2020.

[61] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari, "What's hidden in a randomly weighted neural network?" *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 890–11 899, 2020.

[62] Y. Wang, X. Zhang, L. Xie, J. Zhou, H. Su, B. Zhang, and X. Hu, "Pruning from scratch," in *AAAI*, 2020.

[63] Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. Gonzalez, "Train large, then compress: Rethinking model size for efficient training and inference of transformers," *ArXiv*, vol. abs/2002.11794, 2020.

[64] X. Dong and Y. Yang, "Network pruning via transformable architecture search," in *NeurIPS*, 2019.

[65] M. Gupta, S. Aravindan, A. Kalisz, V. R. Chandrasekhar, and L. Jie, "Learning to prune deep neural networks via reinforcement learning," *ArXiv*, vol. abs/2007.04756, 2020.

[66] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *ECCV*, 2018.

[67] D. W. Blalock, J. J. G. Ortiz, J. Frankle, and J. V. Guttag, "What is the state of neural network pruning?" *ArXiv*, vol. abs/2003.03033, 2020.

[68] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *CoRR*, vol. abs/1412.6550, 2015.

[69] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *ArXiv*, vol. abs/2006.05525, 2021.

[70] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3962–3971, 2019.

[71] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *AAAI*, 2020.

[72]

[73] J. Yim, D. Joo, J.-H. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7130–7138, 2017.

[74] N. Papernot, P. Mcdaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, 2015.

[75] V. Joseph, S. A. Siddiqui, A. Bhaskara, G. Gopalakrishnan, S. Muralidharan, M. Garland, S. Ahmed, and A. R. Dengel, "Going beyond classification accuracy metrics in model compression," 2020.

[76] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *ICML*, 2018.

[77] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *ArXiv*, vol. abs/2103.13630, 2022.

[78] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Aciq: Analytical clipping for integer quantization of neural networks," *ArXiv*, vol. abs/1810.05723, 2018.

[79] Y. Cai, Z. Yao, Z. Dong, A. Gholami, M. W. Mahoney, and K. Keutzer, "Zeroq: A novel zero shot quantization framework," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13 166–13 175, 2020.

[80] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *ArXiv*, vol. abs/1602.02830, 2016.

[81] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *NIPS*, 2015.

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

[83] Z. Zhang, W. Shao, J. Gu, X. Wang, and L. Ping, "Differentiable dynamic quantization with mixed precision and adaptive resolution," *ArXiv*, vol. abs/2106.02295, 2021.

[84] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2754–2761.

[85] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *NIPS*, 2015.

[86] J. Kossaifi, Z. C. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar, "Tensor regression networks," *ArXiv*, vol. abs/1707.08308, 2020.

[87] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2754–2761.

[88] G. G. Calvi, A. Moniri, M. Mahfouz, Q. Zhao, and D. P. Mandic, "Compression and interpretability of deep neural networks via tucker tensor layer: From first principles to tensor valued back-propagation," *arXiv: Learning*, 2019.

[89] N. Aghli and E. Ribeiro, "Combining weight pruning and knowledge distillation for cnn compression," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3185–3192, 2021.

[90] Y. Wang, Y. Lu, and T. Blankevoort, "Differentiable joint pruning and quantization for hardware efficiency," *ArXiv*, vol. abs/2007.10463, 2020.

[91] M. Van Baalen, C. Louizos, M. Nagel, R. A. Amjad, Y. Wang, T. Blankevoort, and M. Welling, "Bayesian bits: Unifying quantization and pruning," *Advances in neural information processing systems*, vol. 33, pp. 5741–5752, 2020.

[92] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," in *Conference on Empirical Methods in Natural Language Processing*, 2016.

[93] J. Park and A. No, "Prune your model before distill it," in *European Conference on Computer Vision*, 2021.

[94] L. Chen, Y. Chen, J. Xi, and X. Le, "Knowledge from the original network: restore a better pruned network with knowledge distillation," *Complex & Intelligent Systems*, 2021.

[95] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.

[96] Y. Boo, S. Shin, J. Choi, and W. Sung, "Stochastic precision ensemble: self-knowledge distillation for quantized deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 6794–6802.

[97] J. Kim, Y. Bhalgat, J. Lee, C. Patel, and N. Kwak, "Qkd: Quantization-aware knowledge distillation," *arXiv preprint arXiv:1911.12491*, 2019.

[98] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marbán, T. Marinc, D. Neumann, T. Nguyen, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek, "Deepcabac: A universal compression algorithm for deep neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, pp. 700–714, 2020.

[99] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[100] S. Zhai, Y. Cheng, Z. Zhang, and W. Lu, "Doubly convolutional neural networks," in *NIPS*, 2016.

[101] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017.

[102] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size," *ArXiv*, vol. abs/1602.07360, 2016.

[103] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural networks*, vol. 22, no. 5-6, pp. 544–557, 2009.

[104] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4793–4801, 2019.

[105] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4320–4328.

[106] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, "Revisiting knowledge distillation via label smoothing regularization," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3902–3910.

[107] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.

[108] R. Wieringa, "Design science methodology for information systems and software engineering," in *Springer Berlin Heidelberg*, 2014.

[109] K. Peffers, T. Tuunanen, C. E. Gengler, M. Rossi, W. Hui, V. Virtanen, and J. Bragge, "Design science research process: A model for producing and presenting information systems research," *ArXiv*, vol. abs/2006.02763, 2020.

[110] V. N. Vapnik, "The nature of statistical learning theory," in *Statistics for Engineering and Information Science*, 2000.

[111] A. Paszke, S. Gross, S. Chintala, G. Desjardins, L. Antiga, M. Bachlechner, and et al., "Pytorch: An imperative style, high-performance deep learning library," 2019. [Online]. Available: http://pytorch.org/

[112] H. Ma, T. Chen, T.-K. Hu, C. You, X. Xie, and Z. Wang, "Undistillable: Making a nasty teacher that cannot teach students," *arXiv preprint arXiv:2105.07381*, 2021.

[113] S. Kornblith, M. Norouzi, H. Lee, and G. E. Hinton, "Similarity of neural network representations revisited," *ArXiv*, vol. abs/1905.00414, 2019.

Appendix