Master Thesis

Training of Ensembles of Quantized Neural Networks with Shared Weights Using Knowledge Distillation

Trainieren von Ensembles quantisierter neuronaler Netze mit geteilten Gewichten mittels Wissensdestillation

Alexander Kromer

Supervisors

Joseph Bethge, Haojin Yang and Prof. Dr. Christoph Meinel
Internet Technologies and Systems

Hasso Plattner Institute at University of Potsdam

27th January 2022

Master Thesis

Training of Ensembles of Quantized Neural Networks with Shared Weights Using Knowledge Distillation

Trainieren von Ensembles quantisierter neuronaler Netze mit geteilten Gewichten mittels Wissensdestillation

by
Alexander Kromer

Supervisors
Joseph Bethge, Haojin Yang and Prof. Dr. Christoph Meinel
Internet Technologies and Systems

Hasso Plattner Institute at University of Potsdam

27th January 2022

Abstract

Quantization of convolutional neural networks (CNNs) proved to reduce memory size and shorten inference time. Hence, quantization facilitates the deployment of CNNs on devices with varying resource constraints by quantizing the model to different bit-widths. However, each model needs to be trained individually, which is expensive in both time and space.

This problem can be circumvented by training models of different bit-widths in an ensemble. All models in the ensemble share the same set of latent weights. In this work, we trained two ensembles, which differ in the bit-widths of the models included in the ensembles. The accuracy of the models can be further improved using knowledge distillation. We investigate whether pretrained or untrained teachers are more useful for teaching the ensemble and propose "Progressive knowledge distillation within the ensemble" as a new method to distill knowledge from a single teacher into an ensemble. Additionally, we test a variety of training schemes in multiple stages to further improve the ensemble's performance.

Results on the CIFAR-100 dataset show that untrained teachers and "Progressive knowledge distillation within the ensemble" provide valuable regularization to the model. Applying knowledge distillation also improves the results on ImageNet. Training in two stages performs best across most experiments from the tested multi-stage training schemes. In the first stage, only activations are quantized before quantizing weights and activations in the second stage. The models in the ensemble reached a higher accuracy than models trained individually.

The results show that knowledge distillation can be used to efficiently train models quantized to different bit-widths in an ensemble with shared weights, thus improving memory efficiency and training speed.

Zusammenfassung

Die Quantisierung faltender neuronaler Netze hat sich als effizienter Weg erwiesen, um den Speicherverbrauch zu reduzieren und die Inferenzzeit zu verkürzen. Daher erleichtert Quantisierung den Einsatz faltender neuronaler Netze auf Geräten mit unterschiedlich beschränkten Ressourcen, indem das Modell auf verschiedene Bitbreiten quantisiert wird. In diesem Fall muss jedoch jedes Model einzeln trainiert werden, was sowohl Trainingszeit als auch Speicherverbrauch erhöht.

Dieses Problem kann umgangen werden, indem Modelle mit unterschiedlichen Bitbreiten in einem Ensemble trainiert werden. Alle Modelle in dem Ensemble teilen die selben latenten Gewichte. In unserer Arbeit haben wir zwei Ensembles trainiert, die sich in den Bitbreiten der in den Ensembles enthaltenen Modelle unterscheiden. Die Genauigkeit der Modelle kann durch Wissensdestillation weiter verbessert werden. Wir untersuchen, ob untrainierte Lehrer oder Lehrer, die zuvor trainiert wurden, besser geeignet sind, um das Ensemble zu unterrichten. Außerdem entwickeln wir "Progressive Wissensdestillation innerhalb des Ensembles". Dabei handelt es sich um eine neue Methode um Wissen von einem einzelnen Lehrer in ein Ensemble zu destillieren. Darüber hinaus testen wir eine Reihe mehrstufiger Trainingsschemata, um die Leistung des Ensembles weiter zu verbessern.

Die Ergebnisse auf dem CIFAR-100 Datensatz zeigen, dass untrainierte Lehrer und die "Progressive Wissensdestillation innerhalb des Ensembles" geeignet sind, um das Modell zu regularisieren. Die Anwendung von Wissensdestillation verbessert außerdem die Ergebnisse auf ImageNet. In der Mehrzahl der durchgeführten Experimente zum Vergleich der mehrstufigen Trainingsschemata, liefert ein zweistufiges Trainingsschema die besten Ergebnisse. In dem Schema werden in der ersten Stufe nur die Aktivierungen quantisiert, bevor in der zweiten Stufe die Gewichte und Aktivierungen quantisiert werden. Die Modelle im Ensemble erreichten eine höhere Genauigkeit als einzeln trainierte Modelle.

Die Ergebnisse zeigen, dass die Wissensdestillation genutzt werden kann, um Modelle, die auf unterschiedliche Bitbreiten quantisiert sind, effizient in einem Ensemble zu trainieren. Dies verbessert die Speichereffizienz und Trainingsgeschwindigkeit.

Contents

1 Introduction

In recent years, computer vision has been an important and fast-growing field of research. It contains methods for solving a variety of different problems, such as image classification or localization[41]. In image classification, the task is to detect the object depicted in an image and categorize it into a distinct class.

With the development of convolutional neural networks (CNNs), machine learning has brought significant advancements to the field of computer vision. CNNs are artificial neural networks that can be used, for example, to extract essential features from images. They can be considered state of the art to solve a variety of computer vision problems, such as image classification[41].

In 1989 LeCun et al.[23] developed *LeNet*, one of the first practical applications of convolutional neural networks. The network successfully recognized handwritten postal zip codes. It consisted of only five layers.

Over time, researchers developed a variety of different architectures for CNNs. This includes for example *AlexNet* and *VGGNets*. *AlexNet* introduced important techniques for CNNs, such as ReLU activation functions[24]. *VGGNets* use small convolution filters but significantly increase the depth of the network to up to 19 layers. An important step towards deep and accurate CNNs was the development of the *ResNet* architecture. Through the use of shortcut connections, the depth of a *ResNet* can be increased significantly. The increase in depth resulted in more accurate networks[14]. In order to achieve even higher accuracies, other network architectures, such as *ShuffleNets* or *ResNeXt*, have been developed.

Improvements in model accuracy often correlate with an increase in the number of parameters. The higher parameter number can increase the model size and inference time. While a *ResNet-34* has a total of 21.8 million parameters[3], modern architectures, such as *ResNeXt-101*, can reach up to 829 million parameters[28]. The resulting models are often too big to be deployed on low-power devices such as mobile phones[19].

Different techniques have been proposed to mitigate this problem. For example, researchers developed various network architectures that aim to reduce the model size and inference speed while maintaining decent accuracy. One of these architectures is *MobileNet*[16]. Another way to accelerate deep neural networks is model compression. Model compression includes techniques such as parameter pruning or quantization. Pruning reduces the number of parameters of the model by removing notes or weights that are redundant or do not contain significant information[6]. However, in this work, we focus on quantization to reduce a model's size.

Usually, weights and activations in a CNN are represented as 32-bit floating-point numbers. When using quantization, the precision of these parameters is reduced. For example, weights are represented as eight-bit integers instead of 32-bit floating-point numbers. This reduction significantly reduces the model's size and increases inference speed[17]. An extreme case is binarization, where binary values represent the parameters[37]. When one

wants to obtain accurate quantized models, it is insufficient to reduce the precision of a trained full precision model, as this results in a significant decrease in accuracy. Higher accuracies can be achieved when training a quantized model from scratch[17]. However, even when training the quantized model from scratch, the accuracy of the quantized model is often lower than the accuracy of its full precision counterpart.

The need to train the quantized model from scratch leads to a problem when quantizing the same model to different bit-widths. In that case, one has to retrain the model on each bit-width and optimize the training for each model individually.

Quantizing the same model to different bit-widths can be helpful when the model shall be deployed to different platforms with varying resource constraints. This use case is depicted in Figure 1.1.

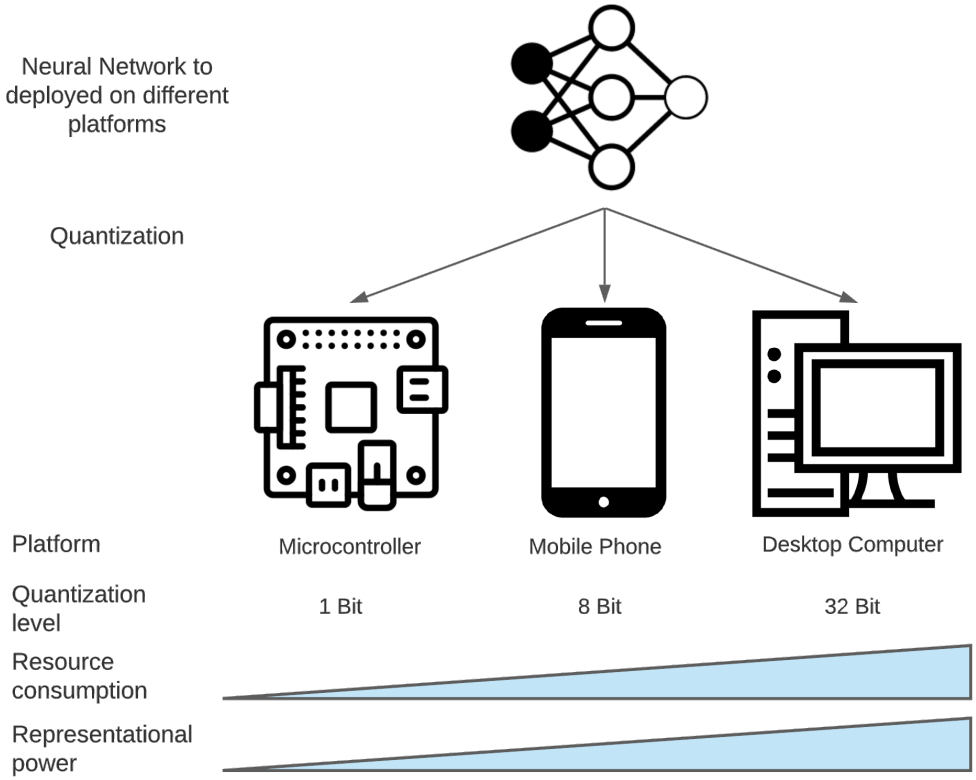

Figure 1.1: Visualization of the problem we want to mitigate. A neural net shall be deployed on three different platforms. Each has varying computational resources. Quantization allows the same model architecture to run on all devices since it lowers the resource consumption of the model. However, when quantized to a low bit-width, the representational power of the model shrinks.[33][34][35][36]

To give a more practical view on the topic, we want to construct an example scenario. In this scenario, we have a model that detects and classifies different types of vehicles in an image. We want to use this model in different applications. Three example applications are:

1. Deploy the model on microcontrollers of traffic surveillance cameras, for example, at a parking lot to classify incoming vehicles.
2. Deploy the model in mobile phone apps that automatically classifies photos taken by the user.
3. Deploy the model on high-power desktop computers to analyze traffic situations based on various pictures.

The three different platforms all have different limitations in terms of computational power. For example, the desktop computer may have more available resources, which allows it to use the full precision model for inference. On the other hand, the microcontroller may have the least power, which requires the smallest model. Therefore, the model needs to be quantized to a low bit-width, like one bit. In that case, it has to account for accuracy losses when using the model for inference. Finally, the computational resources of the mobile phone are in between the microcontrollers resources and the desktop computers resources, which may require a quantization to eight-bit to use the model efficiently. Typically, this use case would require training the model three times, which is cumbersome and may take significant effort to optimize the training for each bit-width.

In this work, we want to mitigate this problem by applying a method to train models of different bit-width simultaneously while reducing memory consumption during inference and the duration of training. We build upon the work of Jin et al[19]. They trained a model that can efficiently be quantized to different bit-widths. To do this, they quantized a set of latent weights to multiple bit-widths. The resulting models were trained simultaneously while sharing the same weights. The final quantized models reached an accuracy comparable to training each model individually. In this work, we adopt the approach of Jin et al.[19] and use an ensemble of models that are quantized to different bit-widths. They all use the same set of shared weights and are trained simultaneously.

The performance of this model could be increased using knowledge distillation. Knowledge distillation is a technique to improve the performance of a small student model by transferring knowledge from a powerful teacher to a student[15]. It has found many applications, for example, to improve the accuracy of quantized models[31][21]. Since knowledge distillation is beneficial to the training of quantized networks, it was also used to train models that can be quantized to different bit-width, for example, by Du et al.[10] and Guerra et al.[46]. However, while these works use some form of knowledge distillation, they lack a comprehensive study on how different knowledge distillation techniques influence the training process and which techniques provide the most significant benefit when training an ensemble of models with shared weights. In this work, we want to examine the effects different knowledge distillation techniques and training schemes have on the ensemble.

In general, this work revolves around studying the following research question:

> How can knowledge distillation be used to improve the accuracy of the individual models in an ensemble of models with shared weights, where each model is quantized to a different bit-width?

Our work provides the following contributions to answer this question:

1. We study how the different models in the ensemble influence each other and how this influence is affected by knowledge distillation.
2. We show that using a teacher with a low capacity is superior to using a high-capacity teacher if the student's capacity is low or not enough regularization is applied to the training.
3. We show that using an untrained teacher is superior to using a pretrained teacher if the capacity gap between student and teacher is big or if the model overfits.
4. We propose "Progressive knowledge distillation within the ensemble", a novel technique in which the quantized models in the ensemble teach each other and show that it can provide useful regularization.
5. We propose and evaluate a set of training schemes in multiple stages that use knowledge distillation to train the ensemble. The best schemes trained an ensemble with quantized activations and full precision weights first. Then, the model is quantized fully and retrained in the second stage.

This thesis is structured in the following way. In Chapter 2 we provide the reader with the necessary background knowledge on CNNs, quantization, and knowledge distillation needed to understand the methods presented in this work. Chapter 3 gives an overview of related work for the topics of quantization, weight sharing between CNNs with different bit-widths, and knowledge distillation. The approaches used in this work are described in Chapter 4, where we provide details on the ensemble with shared weights, the knowledge distillation techniques we test, as well as the proposed multi-stage training schemes. Chapter 5 describes the experiments we conducted on the CIFAR-100 and the ImageNet dataset and presents their results. The results are discussed in Chapter 6. In Chapter 7 we evaluate the performance of the best-trained models before concluding the work in Chapter 8.

2 Background

This chapter gives an overview of the fundamental knowledge needed to understand the methods developed in this work. First, convolutional neural networks (CNNs) and residual networks (*ResNets*), as one example of a convolutional neural network architecture, are introduced in Section 2.1. *ResNets* are used during the experiments in Chapter 5. This is followed by an introduction to quantization in Section 2.2. Quantization is a model compression technique used throughout this work . An explanation of knowledge distillation concludes the chapter in Section 2.3. Knowledge distillation is applied during the training process to boost the performance of the quantized CNNs.

2.1 Convolutional Neural Networks

Convolutional neural networks are artificial neural networks that process data with a grid-like topology, like image data[12, page 330]. A CNN consists of multiple convolutional layers. An activation function and batch normalization are applied between the layers[12, page 339].

The convolution operation is a central element of CNNs. During the convolution, the element-wise product between the input tensor, for example, the image data and a kernel, is calculated. This process is visualized in Figure 2.1.

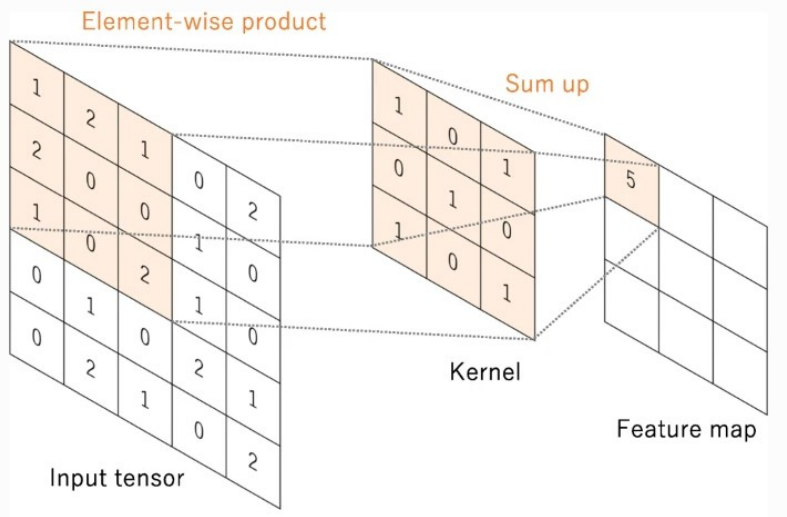

Figure 2.1: Example of one step in a convolution operation applied to a 2D image. Shown are the 2D input tensor, the applied 2D filter, and the resulting value on the feature map[43].

The kernel is a tensor and is moved across the input tensor while the element-wise product is calculated. The result of all the multiplications is the *feature map*[43]. A convolutional layer can have multiple kernels[12, page 331]. The parameters of the convolutional layer, known as weights, get updated during the training process.

After the convolution, a nonlinear activation function is applied to the feature map. An example of an activation function is the rectified linear activation function (ReLU)[12, page 339].

Additionally, batch normalization layers are often used to stabilize the training process and make the training faster. They were proposed by Ioffe et al.[18]. During training, the distribution of inputs for each layer in the network changes with respect to the parameter updates done on other layers. This is referred to as *internal covariance shift*. It makes optimization difficult because updating a layer's parameter using the gradient assumes that the parameters of other layers do not change[12, page 317]. Adding batch normalization layers mitigates this problem by standardizing each layer's inputs. The standardizing is done using the mean and standard deviation of the input. Additionally, the mean and standard deviation of the dataset are learned during the training to be used for inference[18].

The pooling layer is another important layer used for constructing CNNs. They provide a method to downsample the feature maps of the network. The downsampling is done by summarizing a location in the feature map through statistical means[12, page 339]. One example is the maximum pooling, in which an area in the input feature map is represented by its max value in the output[12, page 340].

Over time a variety of different CNN architectures have been developed. This includes residual networks (*ResNets*). Throughout this work, *ResNets* are used for the experiments since they are widely utilized in related literature[19][29][4]. Therefore, using a *ResNet* provides good comparability. The *ResNet* architecture allows for the training of neural networks with hundreds of convolutional layers[14]. Adding additional layers can improve the network's performance. The reason is that deeper layers can learn more high-level features[50]. However, when adding too many layers, the accuracy starts to degrade[14]. Using *ResNets* allows for training deeper CNNs through the introduction of shortcut connections. The shortcut connection adds the input of a *ResNet* block to the output of the block. The concept is visualized in Figure 2.2.
A block typically contains two or three convolutional layers. *ResNetE* is a variation of the *ResNet* architecture, where a shortcut connection is added between each convolution. The change was proposed by Liu et al.[27] and named *ResNetE* by Bethge et al.[1]. The E is short for "Extra Shortcut". Adding the additional shortcut connections can help to improve the accuracy of networks with low bit-widths[27].

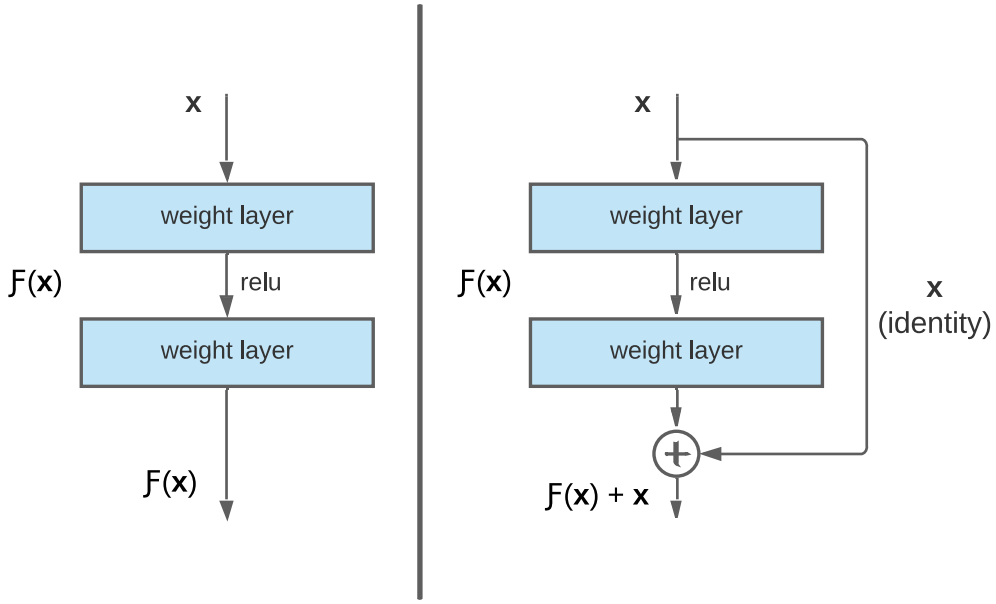

Figure 2.2: CNN layers with and without shortcut connections. The left side shows a CNN layer without a shortcut connection. A CNN layer with shortcut connection, as introduced in [14], is shown on the right side. The shortcut connection performs the identity mapping of the input x. x is then added to the block's output $\mathcal{F}(x)$ [14].

2.2 Quantization

Quantization is a popular technique used to speed up the computation and reduce memory usage of neural networks during inference[17].

It reduces the precision of the network by using integers with a limited number of bits to represent the network's parameters. For example, eight-bit integers could be used instead of the full precision integers for weights and activations[17]. Using integers with a lower bit-width replaces expensive floating-point arithmetic with faster integer arithmetic. Additionally, the size of the quantized model is reduced since the quantized weights require less memory than full precision weights. The inputs to the first convolutional layer and the output layer are often not quantized because quantizing them causes severe accuracy degradation[51].

Weights and activations are quantized during the forward pass. The parameter updates during the training are done on the full precision weights[17]. Weights and activations are quantized using quantization functions. A quantization function q_k transforms a real-valued input, e.g., a weight, to an integer with b bits. While there are different quantization functions, this section focuses on the weight and activation quantization functions proposed by Zhou et al.[51]. The reason is that they will be used throughout this work. We chose these quantization functions since they are well established in related literature. For example, they are used in the works of Sun et al.[42] and Zhuang et al.[52].

Moreover, they do not require additional parameters, which would increase the model complexity. Zhou et al.[51] defines q_k as:

$$z_q = q_k(z) = \frac{1}{2^k - 1} * \text{round}((2^k - 1) * z_r) \tag{2.1}$$

where k is the number of bits and z_r a real-valued input with $z_r \in [0, 1]$.

Weight quantization: Zhou et al.[51] proposes two different weight quantizations, one for quantization to binary bit-width and one for quantizing to bit-widths $b > 1$. The performance difference between these functions is stated as insignificant, so the quantization for bit-widths $b > 1$ is used for all weight quantizations.
The weights are quantized as follows:

$$w_o = 2 * q_k \left(\frac{\tanh(w)}{2 * \max(|\tanh(w)|)} + 0.5 \right) - 1 \tag{2.2}$$

where w is the full precision weight. This function normalizes the weight before applying the quantization function[52].

Activation quantization: Activations could be quantized similarly. However, Zhou et al.[51] reported bad results when using the weight quantization function for activations. Therefore, activations are quantized as follows: First a clip function $clip(x)$ is used to constrain the activation to values in [0,1]. Afterward, q_k is applied[51].

$$a_o = q_k(\text{clip}(x)) \tag{2.3}$$

Thirty-two-bit floating-point values are mapped to integers with a limited range during quantization. Therefore, approximating the floating-point values with integers introduces a quantization error. The error is higher for lower bit-width and causes the accuracy of the model to degrade[20].

2.3 Knowledge Distillation

Knowledge distillation is a popular technique that can be used to train a small student network using a powerful teacher network[40]. The teacher network usually has a higher capacity, which allows it to make more accurate predictions. On the other hand, the student is a smaller model that requires less memory and has shorter inference time but is often less accurate.

The technique was used by Hinton et al.[15] to distill the knowledge of an ensemble of models into a single model. We use knowledge distillation to distill the knowledge of a full precision teacher into an ensemble of quantized networks with shared weights. Section 4.1 describes the ensemble in detail.

One way to distill the teacher's knowledge into the student is logit matching. When using logit matching, the student aims to mimic the teacher's output instead of the ground truth labels. The teachers' outputs are soft labels, which means they are a distribution over the different classes. In contrast, the ground truth labels can be referred to as hard labels. In the case of image classification, the hard labels represent the class membership

of a sample using one-hot encoding. The concept of knowledge distillation through logit matching is shown in Figure 2.3.

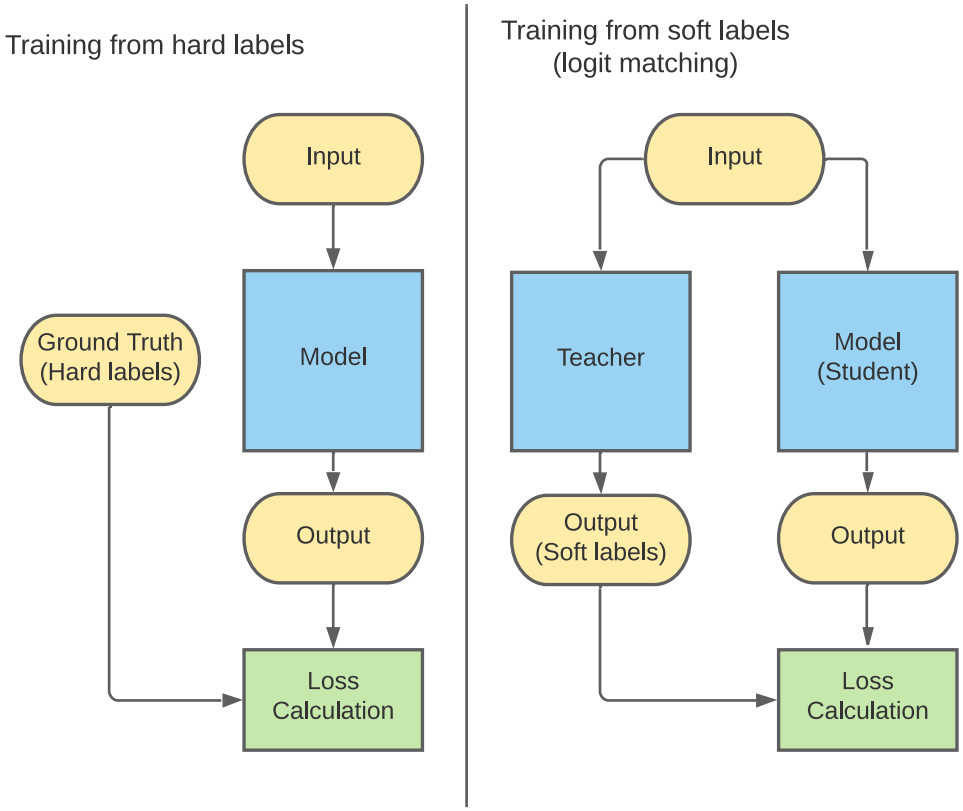

Figure 2.3: Comparison between training from hard labels only and soft labels only. When training from hard labels (left), the loss of the model is calculated from the ground truth. The dataset usually provides the ground truth information. When training from soft labels (right), the loss is calculated from the teacher's output.

A probability distribution can be generated from the output of a neural network by applying the *softmax* function to the output logits z_i. The *softmax* function transforms each of the logits into a class probability q_i. The *softmax* function has the following form[15]:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{2.4}$$

where z_i and z_j are logits and T is the temperature. The temperature T is usually set to one[15]. The output distribution of the *softmax* function are the soft labels[7].

The underlying assumption of logit matching is that the student can learn more information about a sample from the soft labels than the ground truth label[7]. This is because the ground truth labels only contain information about which class a sample belongs to. The soft label, on the other hand, can include additional information. For example, if

high probabilities are assigned for two classes, the sample may contain important features of both classes[7]. This additional knowledge can be transferred from the teacher to the student by training the student on the teacher's soft labels.
Hinton et al.[15] achieves this by introducing a new loss term to the training of the student. The new loss term is the distillation loss L_{KD}. It is calculated as the *Kullback-Leibler divergence* between soft labels of the teachers and the students. The *Kullback-Leibler divergence* can be seen as a distance measure between two probability distributions. By minimizing L_{KD}, the soft labels of the student become more similar to the soft labels of the teacher[7].

Additionally, Hinton et al.[15] uses a second loss term L_{CE} which is the *cross entropy loss* that is calculated from the class labels. Hinton et al.[15] uses the weighted average between L_{KD} and L_{CE} as the training loss. This results in the following loss function:

$$L = \alpha L_{CE} + (1 - \alpha) L_{KD} \tag{2.5}$$

The parameter α balances the influence of L_{KD} and L_{CE}. Using a value of $\alpha = 1$ is equivalent to using only the *cross entropy loss* and a value of $\alpha = 0$ is equivalent to using only the distillation loss for training.

This work uses only L_{KD} for training as proposed by Stanton et al.[40]. We do this to study the effects of knowledge distillation on the training without accounting for possible influences by the hard labels. Unless otherwise stated, we use the logit matching as explained above when applying knowledge distillation for the rest of this work.

3 Related Work

Various approaches to train accurate, quantized neural networks have been proposed in the past. The following chapter gives an overview of related literature regarding quantized neural networks in general in Section 3.1. Afterward, approaches that use weight sharing between networks quantized to different bit-width are shown in are shown in Section 3.2. Finally, the chapter is concluded by an overview of knowledge distillation techniques that can improve a neural network's performance in Section 3.3.

3.1 Quantized Neural Networks

Hubara et al.[17] did groundwork for quantized neural networks. They reduced the bit-widths of weights and activations of the neural network during the forward pass. This resulted in faster prediction times and lower power usage.

Zhou et al.[51] developed *DoReFa*-Net, which is a method to train networks quantized to low bit-widths. They presented quantization functions for weights, activations, and gradients that achieve good results when quantizing to low bit-widths.

Zhuang et al.[53] proposed a training scheme for quantized neural networks that includes two stages. In the first stage, only weights were quantized before also quantizing activations in the second stage.

Bethge et al.[2] provided insights on the training process of binary neural networks by comparing trainings using different architectures and hyperparameters. Additionally, they showed that increasing the number of connections in a quantized neural network increases its accuracy.

Further performance increases were obtained by Zhuang et al.[52], by decreasing the difference between the outputs of quantized and real-valued convolutions. First, they introduced a new loss term computed from the attention maps. Second, they applied re-scaling to the activations.

Martínez et al.[29] developed binary neural networks that achieved an accuracy close to their full precision counterparts. They managed to boost the performance of their baseline model using a variety of techniques, like "Real-to-Binary Attention Matching", and presented a novel training scheme for binary neural networks.

There has also been a variety of research into finding quantization functions that improve the performance of the quantized CNNs. One example is the work of Choi et al.[8]. They proposed Parameterized Clipping Activation (PACT). PACT quantizes activation functions by learning a clipping parameter α that is used to find the right scale for quantization.

3.2 Weight Sharing Between Convolutional Neural Networks with Different Bit-Widths

This section gives an overview of research that uses weight sharing to train models of different bit-widths simultaneously. The overview includes approaches that use a fixed bit-width within each model and approaches that train mixed-precision neural networks.

Jin et al.[19] build upon the idea of adaptive neural networks that can modify their own structure, for example, by adjusting the width of the network. Previous approaches in this field, such as the work of Cai et al.[5], focused on adapting only widths, depth, or kernel size. Jin et al.[19] proposed adaptive bit-widths for weights and activations that can be changed depending on the task and available resources. They tested several approaches to training networks with adaptive bit-widths and achieved performance comparable to individually training models for each bit-width.

Du et al.[10] proposed *Quantizable DNNs*, a neural network that can be dynamically quantized to different bit-widths. They introduced a "consistency-based loss" to train the model. It was defined as the weighted sum of the individual losses of the different bit-width. They also introduced batch normalization layers specific for each bit-width and added knowledge distillation to the training.

Guerra et al.[13] also trained a network that can be used at different quantization levels. They implemented "switchable batch normalization". It is a technique that was proposed by Yu et al.[46] and uses independent batch normalization layers for each compressed model. Guerra et al.[13] also employed a self-distillation technique, in which only the full precision network has access to the ground truth. The other networks in the ensemble aim to match the full precision model's internal representation and output distribution.

Cai et al.[39] combined the concept of jointly training multiple bit-width with neural architecture search. The authors trained a full precision supernet, from which subnets with different bit-width can be sampled using quantization. After training, specific models can be found using the *coarse-to-fine architecture selection* proposed by Yu et al.[45].

Sun et al.[42] improved the performance of networks with adaptive bit-width using knowledge distillation. They dynamically selected the teacher for the student model based on the accuracy of the teacher and the similarity of the student and teacher model. Additionally, they tried to improve the knowledge distillation by swapping the teacher and the student model blocks during training.

Bulat et al.[4] focused on mixed-precision networks, where each layer in the network can have a different bit-width. Their approach used an ensemble with shared weights and separate batch normalization layers to transition between different bit-widths. The network was trained in three stages.

3.3 Knowledge Distillation

Hinton et al.[15] laid important groundwork for the development of knowledge distillation. They distilled the knowledge of an ensemble of models into a single student. The distillation was done by introducing a new loss term. This term was defined as the cross-entropy between the soft labels of the student and the teacher.

Mishra et al.[31] used knowledge distillation to improve the performance of quantized neural networks. They tested three different schemes. The first scheme used a pretrained teacher to teach an untrained student. The second scheme jointly trained the teacher and the student. Third, they used a trained teacher to fine-tune a quantized student model that was first trained in full precision. They reported accuracy gains for each of the schemes. However, the first two schemes outperformed the last one.

Kim et al.[21] proposed a different training scheme to combine knowledge distillation with quantization. In a first *Self-studying* phase, the student is trained alone. In the second *Co-studying* phase, the student is trained jointly with a full precision teacher. Finally, in the third *Tutoring* phase, the teacher's weights are fixed, and only the student is trained on the soft labels of the teacher.

Deng et al.[9] studied the relation between student and teacher performance and showed that students can outperform their teachers given a high enough student capacity and enough distillation data. Another work studying the relationship between student and teacher is the work of Cho et al.[7]. They showed that a larger teacher does not necessarily produce more accurate results if the teacher and student capacity difference is too big.

Mirzadeh et al.[30] worked towards bridging the gap between small student models and large teacher models. Therefore, they developed a multi-stage training scheme that uses a teacher assistant in an intermediate step. The teacher assistant is an intermediate-sized model to bridge the gap between a small student and a big teacher model.

Stanton et al.[40] studied how knowledge distillation improves a student's performance. Their work showed that a student's ability to match the teacher does not necessarily increase generalization. Additionally, they studied the influence of the dataset and optimization process used during training.

4 Methodology

The following chapter presents methods and ideas used in this work to train accurate ensembles of quantized neural networks with shared weights. First, we outline details of the ensemble and the training process in Section 4.1. Second, the knowledge distillation approaches we use to improve the ensemble's performance are introduced in Section 4.2. The approaches include showcasing the advantages of trained and untrained teachers and presenting different techniques to perform knowledge distillation in the ensemble. Last, we present training schemes in multiple stages in Section 4.3.

4.1 Ensemble with Shared Weights

This section introduces the ensemble with shared weights as used during the experiments in this work. First, we show the composition of the ensemble. Second, the training process is described.

Ensemble Composition: The composition of the model is based on the approach of Jin et al.[19]. Figure 4.1 shows an overview.

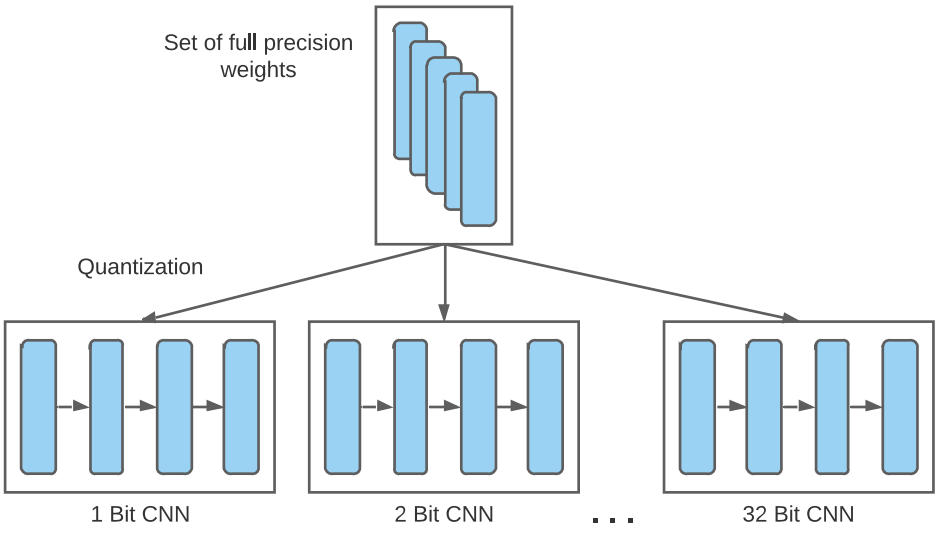

Figure 4.1: Ensemble of CNNs with shared weights. All models in the ensemble use the same set of latent weights. The weights for a model can be retrieved by applying a quantization function to the full precision weights.

The ensemble consists of an arbitrary number of models with different bit-widths. All the models share a set of latent weights and have the same architecture. The shared weights include the weights for the convolutional layers and the output layer. During training, all models update the latent weights.

A quantization function is used to derive a model with a particular bit-width from the set of shared weights. The quantization functions used were proposed by Zhou et al.[51] and are explained in Section 2.2. The weights can be quantized to different bit-widths by adjusting the parameter k, which represents the number of bits. Quantization to different bit-widths allows for sampling models of any desired bit-width from one set of latent weights. Activations of these derived models are also quantized to the desired bit-width. Each model uses an independent batch normalization layer as proposed by Yu et al.[46]. Independent batch normalization is necessary because quantization to different bit-widths results in different variances of the weights and activations, thus requiring different batch normalization layers[19].

Training Process: The ensemble is then jointly trained as proposed by Jin et al.[19]. Pseudocode describing the training is shown in Listing 4.1

Listing 4.1: Pseudocode of the training process of the ensemble

```
Get batch x and labels y
For model in ensemble:
       output = forward pass x through model
       loss = compute loss from output and label y
       Compute gradients for loss

Update weights
```

The training process works in the following way: During the forward pass, a batch of samples is forwarded through all models in the ensemble. The individual loss for each model is calculated. This can be either the *cross entropy loss* or the *distillation loss*, depending on whether knowledge distillation is used during training. Each loss is then backpropagated through the model. The resulting gradients are accumulated on the shared weights. After the gradients of all the individual models are accumulated, the parameters get updated.

4.2 Knowledge Distillation Techniques

This section describes the knowledge distillation techniques used in this work. It focuses on methods used when training the model in only one stage. Techniques used when training the model in multiple stages are described in the next section. The first subsection gives details on the importance of the teacher size. The second subsection describes the concepts of pretrained and untrained teachers. The section concludes by presenting methods to apply knowledge distillation in the ensemble with shared weights successfully.

4.2.1 Choice of the Teacher Size

When using knowledge distillation, the size of the teacher can have a significant impact on the student's accuracy. High-capacity teachers may be beneficial for the training process since they are more accurate in their predictions and the soft labels represent the underlying class distribution better, thus providing better guidance for the student[7]. However, when the capacity difference between student and teacher gets too big, the student's accuracy starts to degrade[9]. A possible reason could be problems of the low-capacity student to mimic the high-capacity teacher[49].

4.2.2 Training Level of the Teacher

For knowledge distillation, a teacher that has already been trained on the dataset can be used. An alternative is using a neural network that has not yet been trained as a teacher. This subsection introduces both approaches.

Pretrained teacher: This approach uses a pretrained teacher to transfer knowledge to the student model. The teacher was trained on the same dataset the student shall train on. Using a pretrained teacher may benefit the training since the teacher already has learned knowledge to provide informed guidance at each training step. This approach is used by Mishra et al.[31] or during the training done by Hinton et al.[15].

Untrained teacher: An alternative to the previous approach is using a teacher that was not previously trained on the dataset. In this case, the student and the teacher can be trained jointly. While the teacher is trained on the hard labels, the student is trained on the soft labels of the teacher. Using an untrained teacher may be beneficial for the student. The soft labels presented by a fully trained teacher may be difficult to mimic for a small student since they already include a lot of additional knowledge. The soft labels of the untrained teacher do not contain much information at the start of the training. As the training continues, the information in the soft labels steadily increases. The student is guided towards the final, complex labels provided by the teacher at the end of the training. This could make it easier for the student to mimic the teacher[31]. On the downside, the classes predicted by the teacher may not be correct in the early stages of the training, which may delay the convergence of the student model. An untrained teacher was used, for example, by Mishra et al.[31] and Zhuang et al.[53].

4.2.3 Knowledge Distillation within the Ensemble

When using knowledge distillation to train a student neural network with a teacher, the distillation loss can be calculated as the *Kullback-Leibler divergence* between the student and the teacher model[15]. When a teacher shall be used to train an ensemble, this is not possible. The ensemble's output can be considered a concatenation of the outputs of the individual models in the ensemble. Calculating the *Kullback-Leibler divergence* between student and teacher would not yield useful results in this case. In order to calculate the distillation loss between student and teacher, this work proposes two techniques, a "Simple knowledge distillation between ensemble and teacher" and "Progressive knowledge distillation within the ensemble". Visualization of both approaches can be seen in Figure 4.2

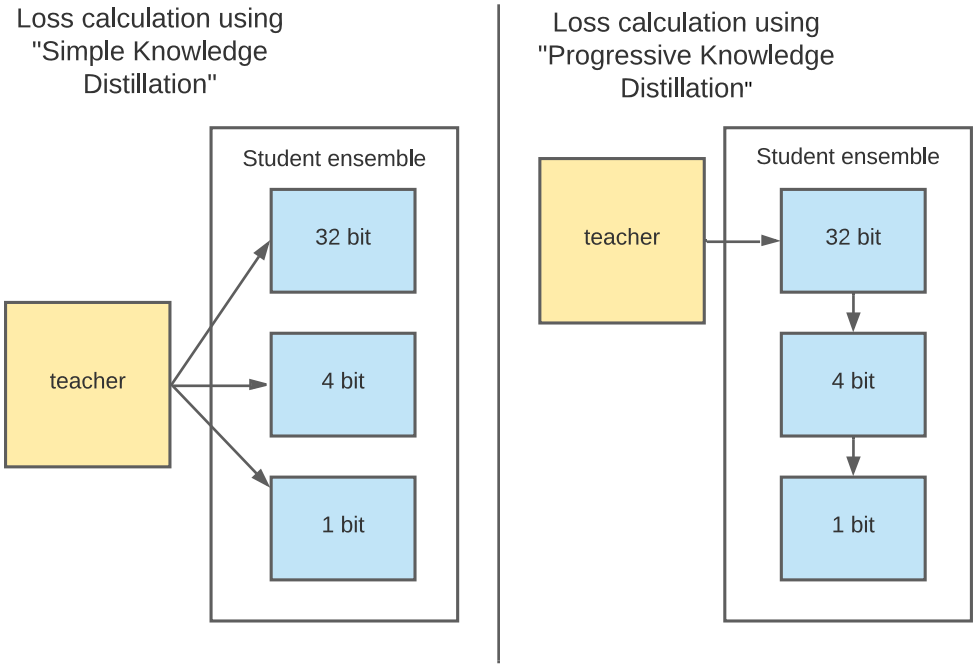

Figure 4.2: Comparison between the simple and progressive knowledge distillation technique using an ensemble with three different bit-widths as an example. Arrows indicate a knowledge distillation between the teacher at the origin of the arrow and the student as the arrow's target. For the simple knowledge distillation (left), the distillation loss is calculated directly between all models in the ensemble and the teacher. When using progressive knowledge distillation (right), the distillation loss is calculated between the teacher and the model in the ensemble with the highest bit-width. The other models in the ensemble are taught by the model with the higher bit-width.

Simple knowledge distillation between ensemble and teacher: When applying this method, we consider the teacher teaching all the individual models in the ensemble. For each model in the ensemble, the distillation loss can be calculated as the *Kullback-Leibler divergence* between its output and the teacher's output. This results in an individual loss for each model in the ensemble. This loss can be used for backpropagation and parameter updates. Parameter updates can then be calculated as described in Section 4.1. This approach provides each model in the ensemble with the same guidance by the teacher.

Progressive knowledge distillation within the ensemble: Using this method, knowledge distillation is applied between the model with the highest bit-width in the ensemble and the teacher. The other models are taught by models in the ensemble. It modifies the idea of the self distillation proposed by Guerra et al.[13]. In the work of Guerra et al.[13] the full precision switch in the *Switchable Precision Neural Network* has access to the ground truth, and other models learn from the soft labels of the full precision model. In our work, the model with the highest bit-width in the ensemble is trained on the soft labels of the teacher. Additionally, each model in the ensemble is trained on the soft labels of the model with the next higher bit-width and trains the model with the next lower bit-width. This approach may be beneficial since the difference in the feature maps between the teacher and a low-precision model in the ensemble is bigger than the difference between two models with low precision. For example, using "Progressive knowledge distillation within the ensemble" a model quantized to two-bit precision could teach a binary model. In that case, the binary model may not be able to mimic the full precision teacher but could mimic the model with a bit-width of two. Therefore, progressively distilling the knowledge within the ensemble may provide better guidance for low-bit-widths models. However, it is also possible that the accurate full precision teacher model provides better guidance to the individual models in the ensemble.

4.3 Training in Multiple Stages

In addition to the knowledge distillation approaches presented in the previous section, a model's performance can be further increased by training in multiple stages. Those training schemes also allow for new knowledge distillation techniques, for example, using an ensemble with real-valued weights and quantized activations as the teacher. This section presents the multi-stage training schemes tested in this work. Those are training in two stages, a training adopted from Martínez et al.[29] and two of its variations, as well as jointly training three models.

Two-stage Training: This training strategy was proposed, for example, by Zhuang et al.[53]. It involves training the model in two stages. Zhuang et al.[53] quantize only the weights in the first stage. In the second stage, the model's activations are also quantized, and the model is retrained. This training scheme is useful because the quantization of weights and activations introduces noise when updating the parameters of the network. The problem is easier to optimize when quantizing only weights or activations. The easier optimization allows for better solutions to be found. Optimizing the second stage is more effective because the model already has a decent starting point instead of training from scratch.

The "Two-stage Training" is used in this work. However, only activations instead of weights are quantized in the first stage. We chose this order of quantization because Liu et al.[26] compared it to quantizing weights first and achieved better results when quantizing activations first. This order of quantization was also used by Bulat et al.[4]. Both stages are guided by a full precision trainer using knowledge distillation[53].

Adopted Real-to-Binary Training: Martínez et al[29] propose a training in three stages, called "Progressive teacher-student". First, a real-valued network trains a real-valued network with a binary architecture, but parameters are kept in full precision. In the second step, the student from the first step is used as a teacher for a network with binary activations and real-valued weights. In the last step, the previous student model is used as a teacher for a fully quantized model.

However, the proposed training scheme is not directly applicable to the ensemble used in this work. The first stage of the training involves training a binary architecture with real-valued weights and activations. In the case of the ensemble, this is not useful since there would be no difference in the models contained in the ensemble if all of them were full precision models. Therefore, the first stage used by Martínez et al[29] is not used in the adopted training scheme. As a result, the training contains the following two stages:

1. Train an ensemble with full precision weights and quantized activations using a full precision teacher model.
2. Train an ensemble with quantized weights and activations using the ensemble from stage one as the teacher.

This training scheme differs from the "Two-stage Training" since an ensemble is used as a teacher in the second stage. When using the ensemble as a teacher, calculating the distillation loss is not trivial. The output of each ensemble contains multiple outputs of the contained models. This work proposes to calculate the distillation loss for each model in the student ensemble as:

$$L_m = KL(O_m, O_{m'})$$

where m is a model in the student ensemble with bit-width b, and m' is a model in the teacher ensemble with bit-width b. This way, each model in the student ensemble is taught by the model in the teacher ensemble with the same bit-width. It is visualized in Figure 4.3

Adopted Real-to-Binary Training with attention matching: In addition to the knowledge distillation in multiple stages that uses the logit matching, Martínez et al.[29] also use a technique called "Real-To-Binary Attention Matching". The technique builds upon the assumption that the accuracy of the quantized model improves if the output of the quantized convolutions matches the outputs of the corresponding real-valued convolutions. To achieve this matching, an additional loss term L_{att} is introduced at the end of each convolutional block. L_{att} is calculated from the attention maps of the quantized and real-valued activations.

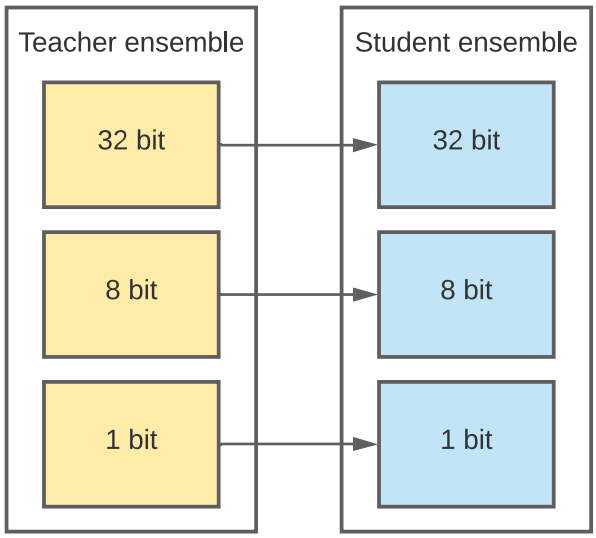

Figure 4.3: Knowledge distillation using an ensemble teacher. Arrows indicate a knowledge distillation between the teacher at the origin of the arrow and the student as the arrow's target. Each model in the student ensemble is taught by the teacher ensemble with the same bit-width.

For a set of convolutional blocks J L_{att} can be calculated as

$$L_{att} = \sum_{j=1}^{J} \| \frac{Q_S^j}{\|Q_S^j\|_2} - \frac{Q_T^j}{\|Q_T^j\|_2} \|$$

where $Q^j = \sum_{i=1}^{c} |A_i|^2$ and A_i is the i–th channel of an activation map A[29]. The index S indicates a student model and index T a teacher model.

Our approach uses an ensemble of models as teachers and an ensemble as students. We implement the attention matching in the same way we did for the knowledge distillation in the "Adopted Real-to-Binary Training". Therefore, we calculate the attention loss between the student and teacher models quantized to the same bit-width.

Adopted Real-to-Binary Training with continued training: This training scheme combines the "Two-stage Training" from [53] and the "Adopted Real-to-Binary Training". It follows the training stages of the "Adopted Real-to-Binary Training". This means, first, an ensemble with quantized activations and full precision weights is trained using a full precision teacher. This ensemble is then used as a teacher for a fully quantized ensemble. However, the student in the second stage is not an untrained ensemble. Instead, the ensemble from the previous stage is used as the student, and its weights and activations are fully quantized. Like the reasoning for the "Two-stage Training", retraining the previous model may provide a better starting point since the underlying problem is easier to optimize.

Joint training of all stages: As pointed out in Section 4.2.2, it may be useful to jointly train a teacher and the student instead of using a pretrained teacher. The reason is that jointly training both ensembles could provide a better guidance signal during the early stages of the training. It is possible to use this idea for the multi-stage training process of "Adopted Real-to-Binary". Instead of training one stage after another, all stages are trained jointly. This results in a full precision teacher A teaching ensemble B with quantized activations and full precision weights. Ensemble B is used as a teacher for a fully quantized ensemble C. In each iteration of the training process, the parameters of A, B, and C are updated.

5 Experiments

This chapter describes the experiments conducted to study which knowledge distillation (KD) methods are beneficial and how they compare to each other. The tested methods included the "Training Level of the Teacher", the "Knowledge distillation within the ensemble", and the multi-stage training schemes.

Datasets used during the experiments: We conducted experiments on two datasets. Those datasets are CIFAR-100[22] and ImageNet[38]. Imagenet was chosen because it is a popular dataset that allows easy comparisons between our experiments and the current state of the art. CIFAR-100 contains fewer and smaller images. We chose it because the small size allows for faster testing of various ideas. Also, comparing the results of the experiments across datasets of different sizes may yield additional insights. Section 5.1 shows the experiments conducted on CIFAR-100, and their results. This is followed by the presentation of the ImageNet experiments in Section 5.2.

Model architecture used in the experiments: During the experiments, the architecture for the models trained in the ensemble was a *ResNetE* as introduced in Section 2.1. We chose this architecture since it allows for efficient training of networks quantized to low bit-widths. We used a *ResNetE* with 18 convolutional layers. The low number of layers was chosen to study the effects of knowledge distillation on a small student model. Additionally, real-valued downsample layers were used for the quantized model in the ensemble. The usage of real-valued downsample layers allows for significant performance gains with only a small increase in computational complexity[29]. The first and last layer was also real-valued to prevent a significant performance degradation[51]. When quantizing weights and activations, we used the quantization functions introduced in section 2.2.

There is a difference in the initial layers between the model used for CIFAR-100 and the model used for ImageNet. This difference is due to the different sizes of the input images. The architecture used for CIFAR-100 uses a batch normalization layer followed by a convolutional layer with a kernel size of 3, a stride of 1, and a padding of 1. The ImageNet architecture uses a batch normalization layer and a convolutional layer, with a kernel size of 7, a stride of 2, and a padding of 3. The layers are followed by another batch normalization, a ReLU activation function, and a max-pooling layer. The pooling layer has a pool size of 3, a stride of 2, and a padding of 1.

Ensembles used in the experiments: In addition to testing on two different datasets, we also tested two different ensembles. One ensemble was constructed to maximize the accuracy of the binary model in the ensemble, and the other to maximize the accuracy of the four-bit model. High binary accuracy was chosen as a target since it represents

the highest quantization level and, therefore, the maximum model size reduction. This ensemble will be called $Ensemble_1$. We also aimed to achieve high four-bit accuracy with our approaches. To achieve this, we used a different ensemble we will reference as $Ensemble_4$. We chose this target bit-width since the industry desires good four-bit accuracy. The two ensembles differed in the number and quantization level of the models contained in each ensemble. The construction of the ensembles is described in Section 5.1.2.

Experiment Implementation: All experiments were implemented using *BMXNet* [44]. *BMXNet* is an open-source library based on *Apache MXNet*.[1] It supports the development of quantized neural networks and was developed at the Hasso Plattner Institute [44]. The code used to conduct the experiments is available for download.[2]

5.1 Experiments on CIFAR-100

This section describes the experiments conducted on the CIFAR-100 dataset. The CIFAR-100 dataset contains 100 different classes, e.g. "shark" or "forest". Each class contains 600 images. Those are split into 500 images for training and 100 images for testing. Each of the images is colored and has a size of 32x32 pixels [22].

All experiments on CIFAR-100 used the same hyperparameters. They are listed in Section 5.1.1. The table also includes details on the trained model, training hyperparameters, and applied image augmentation techniques. Afterward, we describe the conducted experiments in detail and present their results. Since the CIFAR-100 dataset is small, the results can vary, even if all hyperparameters are kept the same. For example, this can be caused by differences in the model initialization. To mitigate these issues, the results presented in this section were the average of three independent runs. The standard deviations for all of the experiments on CIFAR-100 are listed in Appendix A.1. Table 5.1 provides an overview of the conducted experiments.

The experiments in which the models were trained in a single stage were conducted using three different teachers with varying capacities. Using multiple teachers allowed us to gain additional knowledge about how the teacher's capacity affects the success of the training. The reasons as to why the teacher size can influence the performance of the ensemble are given in Section 4.2.1. The teachers used were a *ResNet-20*, a *ResNet-56* and a *ResNet-110*. The *ResNet-20* had the lowest capacity, while the capacity of the *ResNet-110* was the highest. They were retrieved from the "gluon model zoo"[3] and were implemented as proposed by He et al. [14]. For the experiment comparing trainings in multiple stages, only the teacher that worked best in the previous step was used. Before describing the different experiments in detail, the next section gives an overview of the hyperparameters used during the training.

[1] https://mxnet.apache.org
[2] https://www.dropbox.com/sh/9rtepd5rt35qbyv/AABoFjlZpFipWzr3UdSwAEq3a?dl=0
[3] https://cv.gluon.ai/model_zoo/classification.html

Experiment	Description	Methods	Section
Ensemble Setup	Test which models to include in the ensemble to reach high binary or high four-bit accuracy	-	Section 5.1.2
Training Level of the Teacher	Test whether pretrained or untrained teachers perform better	Section 4.2.2	Section 5.1.3
KD within the Ensemble	Test whether the simple or progressive KD performs better	Section 4.2.3	Section 5.1.4
Multiple Stages	Test which training schemes in multiple stages are beneficial	Section 4.3	Section 5.1.5

Table 5.1: Overview of the experiments conducted on the CIFAR-100 dataset. In addition to the experiment descriptions, the table lists the section explaining the used method and the section in which the experiment is conducted. Knowledge distillation is as abbreviated with KD.

5.1.1 Hyperparameters and Image Augmentation

This section gives an overview of the hyperparameters used when training the ensemble on the CIFAR-100 dataset before listing the image augmentation techniques used for train and test images.

Training Hyperparameters: The ensemble was trained for 150 epochs with a batch size of 128. The initial learning rate was 0.01 and was steadily reduced during the training using cosine learning rate scheduling. We used *Rectified Adam* (RAdam), as proposed by [25], as the optimizer. A threshold of 1.3 was used for gradient canceling. Additionally, we chose a weight decay of $1 * 10^{-4}$ when training $Ensemble_4$. We did not use weight decay for $Ensemble_1$, since weight decay has a negative influence on the performance of the binary model[29].

The teachers were trained for 150 epochs with a batch size of 128. The learning rate was initialized to 0.01. It was reduced during the training using cosine learning rate scheduling. The optimizer used was SGD with a momentum of 0.9. Additionally, a weight decay of $1 * 10^{-4}$ was chosen. After training, the teachers trained on CIFAR-100 reached the following accuracies:

- *ResNet-20*: 66.76%
- *ResNet-56*: 70.73%
- *ResNet-110*: 72.94%

Image Augmentation: Different image augmentation was applied to the training and test data. The train images were augmented stronger to prevent the model from overfitting the training data. For training, the following image augmentation techniques were applied:

- **RandomCrop:** Applies padding of 4 pixels to the input image and randomly crops it to a size of 32x32
- **RandomFlipLeftRight:** Randomly flips the input image left or right
- **Normalize:** Normalizes the input image using the mean and standard deviation of the dataset

For testing, only normalization was applied. The different techniques used implementations from GluonCV. More information on the applied augmentations can be found in the official documentation[4]

5.1.2 Ensemble Setup

Before training the ensembles using knowledge distillation, we wanted to explore how a model included in the ensemble influences the performance of the other models. We suspect that the models influence each other's performance because of the shared weights. Therefore, the experiment aimed to construct two ensembles, one suitable for reaching a high binary accuracy during training and one to reach high four-bit accuracy.

Ensemble with High Binary Accuracy To construct an ensemble with high binary accuracy, we conducted an experiment in which ensembles containing different models were trained. The total number of possible bit-width reaches from 1 to 32. Testing all possible ensembles that can be constructed using these bit-widths requires a lot of computational resources. To lower the resource usage, we restricted the set of possible bit-width to powers of two, which are 1, 2, 4, 8, 16, and 32. This set puts less attention on the higher bit-widths than the lower ones. This is useful since the performance difference between low bit-width, for example, one to two bits, is bigger than between higher bit-widths like 16 to 32. Therefore, we believe that models quantized to different but high bit-widths have a similar influence on the ensemble.

With the given bit-widths, the number of ensembles that can be composed is still too high to test completely. Therefore, we designed test cases that allow us to judge the influence of individual models. For example, comparing an ensemble with bit-widths $= \{32, 4, 2, 1\}$ and one with bit-widths $= \{32, 8, 4, 2, 1\}$ allows to judge the influence of the eight-bit model on the ensemble. While this does not necessarily result in finding the optimal performance, it can give an idea of the influence of single models on the ensemble. The results of the experiments are shown in Table 5.2.
The experiments show that the ensemble with bit-widths $= \{1, 2, 4, 8, 32\}$ results in the highest binary accuracy. It also has the highest accuracy for all other bit-widths, except 32 bit.

The difference in binary accuracy between the ensembles is quite small. The same is true for the accuracy of the other bit-widths. The biggest difference in accuracy across

[4] https://mxnet.apache.org/versions/1.8.0/api/python/docs/api/gluon/data/vision/transforms/index.html

bit-widths	1	2	4	8	16	32
1, 2, 4, 8, 16, 32	64.97	66.19	66.38	66.5	**66.36**	66.27
1, 2, 4, 8, 32	**65.30**	**66.35**	**66.46**	**66.54**	-	66.31
1, 2, 8, 32	64.90	66.14	-	66.05	-	66.18
1, 2, 4, 32	64.94	66.23	66.44	-	-	66.12
1, 2, 4, 8	65.03	66.11	66.45	66.45	-	-
1, 4, 8, 32	65.29	-	66.5	66.35	-	**66.57**

Table 5.2: Results for comparing ensembles with models of different bit-widths. Shown is the accuracy reached by each model in %. Standard deviations are listed in Appendix A.1.1.

all tested ensembles and bit-widths is the difference in the eight-bit accuracy between the ensembles with bit-widths $= \{1, 2, 8, 32\}$ and bit-widths $= \{1, 2, 4, 8, 32\}$. The difference is 0.49%. The smallest difference observed in the experiment is the difference in four-bit accuracy between the best performing ensemble (bit-widths $= \{1, 2, 4, 8, 32\}$) and the ensemble with the worst four-bit accuracy (bit-widths $= \{1, 2, 4, 8, 16, 32\}$). The difference is only 0.08%.

The results indicate that the influence different models in the ensemble have on each other is small. Still, we use the ensemble that reached the highest binary accuracy for the rest of this work. It is the ensemble with bit-widths $= \{1, 2, 4, 8, 32\}$ and it will be referred to as $Ensemble_1$.

Ensemble with High Four-Bit Accuracy In this paragraph, we describe the ensemble used to maximize the accuracy of the four-bit model. The experiment conducted to construct a model with a high binary accuracy indicates that the influence of different models in the ensemble on each other is only marginal. Therefore, we used the combination of bit-widths that reached the highest four-bit accuracy in the previous experiment as a starting point. It was the ensemble with bit-widths $= \{1, 2, 4, 8, 32\}$.

Related work on ensembles with shared weights indicates that the inclusion of binary models in the ensemble degrades the accuracy across all bit-widths when training the ensemble alone[13] or when using knowledge distillation[10]. When training ensembles aimed to reach a high four-bit accuracy, it is not necessary to include a binary model. Therefore, we wanted to test if the other bit-widths' accuracies improve when removing the binary model from the ensemble. The results are shown in Table 5.3.

bit-widths	weight decay	1	2	4	8	32
1, 2, 4, 8, 32	No	65.30	66.35	66.46	66.54	66.31
2, 4, 8, 32	No	-	65.67	66.13	65.89	65.86
2, 4, 8, 32	Yes	-	**67.21**	**67.55**	**67.4**	**68.41**

Table 5.3: Comparison of ensemble setups for the $Ensemble_4$. The comparison includes one ensemble with the bit-widths that provided the highest accuracy for $Ensemble_1$. The second ensemble in the comparison excludes the binary model in the ensemble. For the third ensemble, a weight decay of $1 * 10^{-4}$ is added. All values are stated in %. Standard deviations are listed in Appendix A.1.2.

The test shows that removing the binary model from the ensemble reduces the accuracy of the ensemble across all bit-widths. This contrasts our assumption that the inclusion of the binary model degrades the performance of the other bit-widths. Nevertheless, we still used the ensemble without the binary model as the $Ensemble_4$. The reason is that using two ensembles allows us to compare the effects of the following experiments between them, which may allow gaining additional insights.

Additionally, removing the binary model from the ensemble allows making effective use of weight decay, which would degrade the performance of the binary model. As mentioned in Section 5.1.1, we used a weight decay of $1 * 10^{-4}$ for the $Ensemble_4$. We wanted to verify if it is beneficial to use the weight decay in the ensemble by comparing it to the ensemble trained without weight decay. The results are shown in Table 5.3.

It can be seen that adding a weight decay provides significant improvements across all bit-widths in the ensemble, with an average improvement of 1.52%. The biggest accuracy increase is obtained for the 32-bit model. Its accuracy increases by 2.23%. Therefore, weight decay is used when training the $Ensemble_4$ in the upcoming experiments.

After conducting experiments to find the ensembles most suitable for achieving an accurate binary model and an accurate four-bit model, experiments to enhance the performance of these models with knowledge distillation are conducted in the following subsections.

5.1.3 Training Level of the Teacher

This experiment aimed to determine whether using a pretrained teacher or training a teacher with the ensemble is more useful. More details on the reasoning and method can be found in Section 4.2.2. We trained ensembles using pretrained and untrained teachers to compare which approach performs better. In this experiment, the "Simple knowledge distillation between ensemble and teacher" was used to train the ensemble. This means each model in the ensemble trains on the teacher's soft labels. It is described in detail in Section 4.2.3.

Both ensembles, the one optimized for one bit accuracy ($Ensemble_1$) and the one for high four bit accuracy ($Ensemble_4$) were tested. Each was tested on all three teachers (*ResNet-20*, *ResNet-56*, *ResNet-110*). During each test, the ensemble was trained once using a pretrained teacher and a second time using the same teacher architecture but training the teacher with the ensemble. The results for $Ensemble_1$ are shown in Table 5.4, and the results for $Ensemble_4$ are shown in Table 5.5. Additionally, a comparison of the results is shown in Figure 5.1 to provide better visualization.

The experiment shows that almost all models from ensembles that train using knowledge distillation outperform the ensemble trained alone. Therefore, applying knowledge distillation improves the ensemble's performance on the CIFAR-100 dataset. The only exception is the training of $Ensemble_4$ when using a pretrained *ResNet-110* teacher. In that case, the ensemble trained with a teacher has lower two-bit accuracy than the ensemble trained without a teacher.

When looking at the results using a pretrained teacher, the *ResNet-20* teacher outperforms the other two teachers. Additionally, the *ResNet-56* teacher outperforms the *ResNet-110* teacher. We observed this for both tested ensembles. Therefore, we can assume that low-capacity teachers are beneficial when using pretrained teachers.

Teacher	Training	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	65.2	66.35	66.46	66.54	66.31
ResNet-20	Pretrained	68.56	70.39	70.76	70.74	70.77
ResNet-20	Untrained	68.69	70.34	70.71	70.69	70.70
ResNet-56	Pretrained	66.24	68.05	68.37	68.4	68.32
ResNet-56	Untrained	69.3	71.0	71.51	71.5	71.6
ResNet-110	Pretrained	65.66	67.49	67.78	67.79	67.71
ResNet-110	Untrained	**69.51**	**71.48**	**71.86**	**71.89**	**71.91**

Table 5.4: Comparison of the accuracies of the models included in $Ensemble_1$ using pretrained and untrained teachers. Results obtained from training the ensemble without knowledge distillation are added for comparison. All values are stated in %. Standard deviations are listed in Appendix A.1.3.

Teacher	Training	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	67.21	67.55	67.4	68.41
ResNet-20	Pretrained	69.46	70.17	70.23	70.88
ResNet-20	Untrained	68.92	70.02	70.08	70.55
ResNet-56	Pretrained	67.5	68.19	68.21	69.13
ResNet-56	Untrained	**69.85**	**70.79**	**70.89**	**71.66**
ResNet-110	Pretrained	66.57	67.68	67.56	68.8
ResNet-110	Untrained	69.52	70.57	70.65	71.48

Table 5.5: Comparison of the accuracies of the models included in $Ensemble_4$ using pretrained and untrained teachers. Results obtained from training the ensemble without knowledge distillation are added for comparison. All values are stated in %. Standard deviations are listed in Appendix A.1.4.

When using an untrained *ResNet-20* teacher, the performance of $Ensemble_1$ barely improves, and the accuracy of $Ensemble_4$ decreases compared to using a pretrained teacher. However, when using one of the bigger teachers, training the teacher jointly with the ensemble lead to significant accuracy gains. The biggest of those gains is obtained by $Ensemble_1$. For example, when using an untrained *ResNet-110* teacher, the accuracy of the binary model improves by almost 4% compared to using a pretrained teacher. The improved accuracy leads to the ensembles trained jointly with a high-capacity teacher to outperform those trained with a *ResNet-20* teacher. For $Ensemble_1$, the best results were obtained using an untrained *ResNet-110* teacher, which reaches a binary accuracy of 69.51%. $Ensemble_4$ performed best using an untrained *ResNet-56* teacher, reaching a four-bit accuracy of 70.79%.

Additionally, in all test cases, we can see that, reaching the highest accuracy on the target bit-width corresponds to reaching the highest accuracy on the other bit-widths. Thus, we can argue that, the different knowledge distillation techniques improve the entire ensemble's performance, not just specific bit-widths. When comparing the performance of the ensembles, it can be seen that, $Ensemble_1$ provides better results across all bit-widths except for the two-bit model.

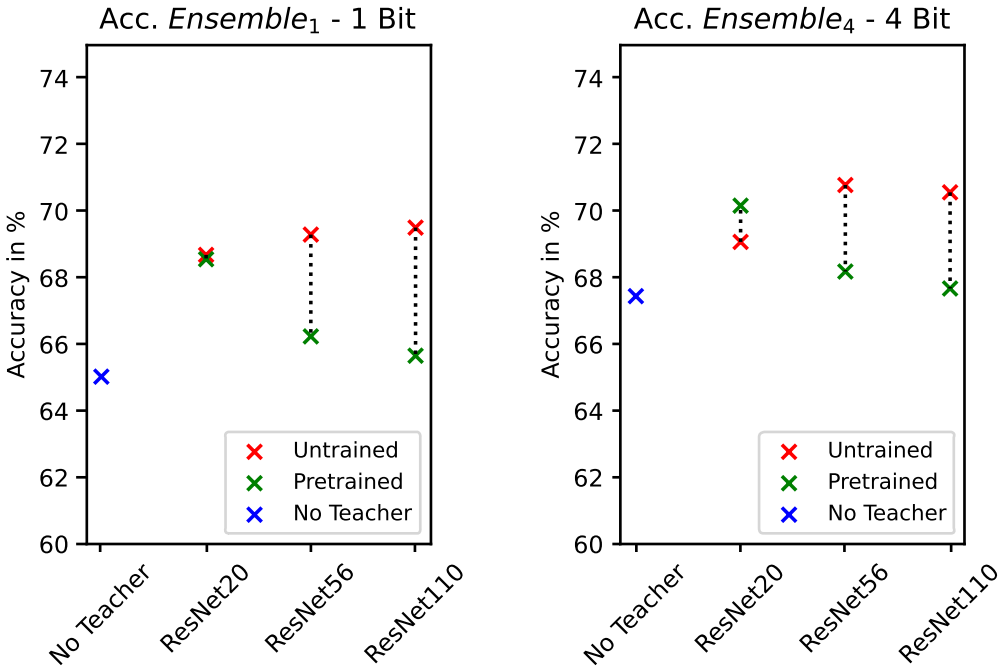

Figure 5.1: Comparison of the accuracy, when using knowledge distillation with a pretrained or untrained teacher for *Ensemble*$_1$ (left) and *Ensemble*$_4$ (right). The accuracy of the target bit-width is shown. Accuracy of training without knowledge distillation is added for comparison.

In summary, the experiment generated the following insights on the CIFAR-100 dataset:

- Using knowledge distillation to train the ensemble improves its accuracy
- Jointly training the ensemble with an untrained high-capacity teacher yields the best results
- Using pretrained teachers provides the best results with low-capacity teachers

These insights are discussed in Section 6.2.1.

5.1.4 Knowledge Distillation within the Ensemble

This experiment aimed to determine how to perform knowledge distillation in the ensemble. It compared the "Simple knowledge distillation between ensemble and teacher" with the "Progressive knowledge distillation within the ensemble". In the simple approach, the distillation loss for each model is calculated directly from the model and the teacher output. When using progressive distillation, models with a high bit-width in the ensemble train the models with lower bit-widths. Section 4.2.3 introduces these techniques in detail.

Again, *Ensemble*$_1$ and *Ensemble*$_4$ were tested on *ResNet-20*, *ResNet-56* and *ResNet-110* teachers. We jointly trained the teacher with the ensemble since it provided the best results during the experiments conducted in Section 5.1.3.

The results obtained from training $Ensemble_1$ are shown in Table 5.6, and the results for $Ensemble_4$ are shown in Table 5.7. The accuracies for the target bit-width of both ensembles are visualized in Figure 5.2.

Teacher	KD Mode	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	65.2	66.35	66.46	66.54	66.31
ResNet-20	Simple	68.69	70.34	70.71	70.69	70.70
ResNet-20	Progressive	67.61	69.1	69.75	70.14	70.24
ResNet-56	Simple	69.3	71.0	71.51	71.5	71.6
ResNet-56	Progressive	69.62	71.25	71.69	71.7	71.66
ResNet-110	Simple	69.51	71.48	71.86	71.89	71.91
ResNet-110	Progressive	**70.1**	**71.68**	**72.17**	**72.14**	**72.04**

Table 5.6: Comparison of the accuracies of the models included in $Ensemble_1$ using simple KD and progressive KD within the ensemble. Results obtained from training the ensemble without knowledge distillation are added for comparison. All values are stated in %. Standard deviations are listed in Appendix A.1.5.

Teacher	KD Mode	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	67.21	67.55	67.4	68.41
ResNet-20	Simple	68.92	70.02	70.08	70.55
ResNet-20	Progressive	69.29	69.95	70.12	70.45
ResNet-56	Simple	69.85	70.79	70.89	71.66
ResNet-56	Progressive	71.26	72.11	72.12	72.12
ResNet-110	Simple	69.52	70.57	70.65	71.48
ResNet-110	Progressive	**71.95**	**72.76**	**72.69**	**72.59**

Table 5.7: Comparison of the accuracies of the models included in $Ensemble_4$ using simple KD and progressive KD within the ensemble. Results obtained from training the ensemble without knowledge distillation are added for comparison. All values are stated in %. Standard deviations are listed in Appendix A.1.6.

The experiments show that, using the "Progressive knowledge distillation within the ensemble" performs better than using "Simple knowledge distillation between ensemble and teacher". It improved the accuracy of the ensembles in all test cases, except when using a *ResNet-20* teacher for $Ensemble_1$. In that case, the accuracy of the binary model is reduced by 1.08%.

When looking at the results obtained from using a *ResNet-56* or *ResNet-110* teacher for the $Ensemble_1$ the accuracy gains using the progressive approach are rather small. For example, when using the *ResNet-56* the accuracy of the target bit-width increases only by 0.32%. Looking at the $Ensemble_4$ the accuracy gains are larger. For example, when using a *ResNet-110* teacher, the progressive approach improves the performance of the four bit model by 2.19%. It should be noted that the accuracy gains obtained from using the progressive approach seem to correlate with the teacher's complexity, leading to bigger accuracy gains for high-capacity teachers.

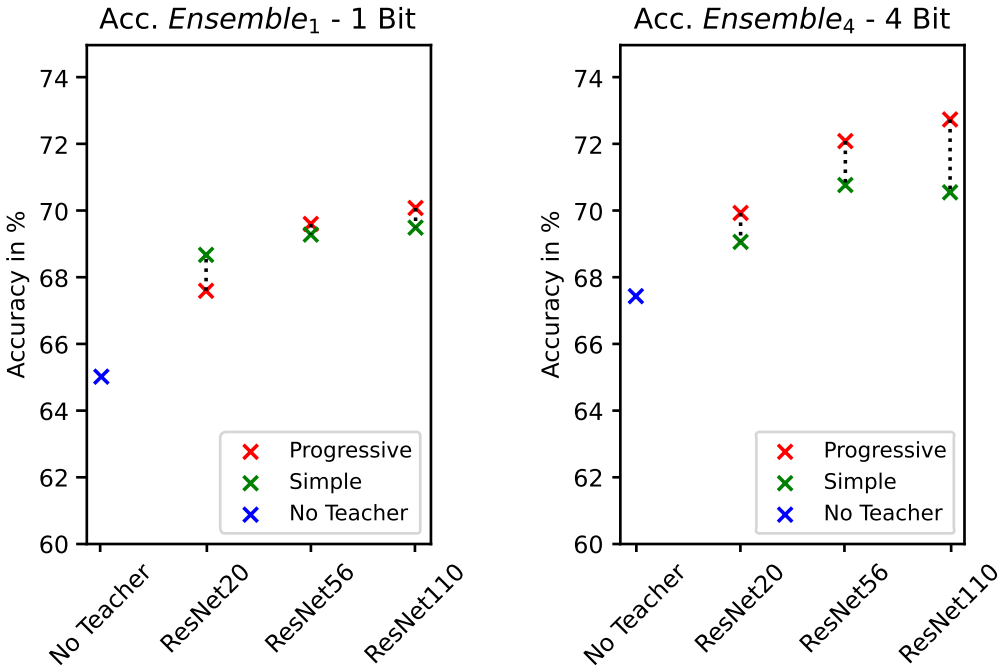

Figure 5.2: Comparison of the accuracy, when using simple or progressive knowledge distillation for *Ensemble*$_1$ (left) and *Ensemble*$_4$ (right). The accuracy of the target bit-width is shown. Accuracy of a training without knowledge distillation is added for comparison.

The highest accuracies for both ensembles are obtained using "Progressive knowledge distillation within the ensemble" and a *ResNet-110* teacher. For the *Ensemble*$_1$ the highest binary accuracy is 70.1 and for the *Ensemble*$_4$ the highest four-bit accuracy is 71.95%.

When comparing the performance of *Ensemble*$_1$ and *Ensemble*$_4$ it can be seen that the *Ensemble*$_4$ performs better on all bit-widths that are included in both ensembles. The performance gap is bigger on the lower bit-widths compared to the higher bit-widths. The performance of the two-bit models differs by 1.85%, while the difference on the full precision models is only 0.55%. This finding is especially interesting, since the the results of Section 5.1.3 showed a better performance for *Ensemble*$_1$.

Summarizing the experiments, we gained the following insights:

- Progressive knowledge distillation improves the performance of the ensemble when using high-capacity teachers
- When using low-capacity teachers, the performance gain are lower, or the ensembles accuracy even decreases

Section 6.2.1 discusses these results.

5.1.5 Multi-Stage Training Schemes

This section describes the experiments conducted to compare different multi-stage training schemes. We compare five different training schemes. Those are:

- Two-stage Training
- Adopted Real-to-Binary Training
- Adopted Real-to-Binary Training with attention matching
- Adopted Real-to-Binary Training with continued training
- Joint training of all stages

The individual training schemes are described in Section 4.3. To choose the best teacher for the ensemble, we build upon the knowledge of the previous experiments in Section 5.1.3 and Section 5.1.4. Therefore, we used a *ResNet-110* teacher that was jointly trained with the ensemble and used the "Progressive knowledge distillation within the ensemble" since these techniques reached the highest accuracies.

The second stage of the Real-to-Binary trainings and the joint training of all stages uses an ensemble with quantized activations and real-valued weights as a teacher for a fully quantized ensemble. In this case, we apply the knowledge distillation using an ensemble teacher as shown in Figure 4.3.

We use a lower weight decay of $2 * 10^{-6}$ when training the first stage of $Ensemble_4$. The reason is that the standard weight decay of $1 * 10^{-4}$ leads to a malformed learning curve and a bad performance of the ensemble. We conducted tests using different weight decays, which can be found in the Appendix A.2.

For the $Ensemble_1$ the results for all bit-widths are shown in Table 5.8, and the results for $Ensemble_4$ are shown in Table 5.9. The performance in the first stage is only stated once since the ensembles with quantized activations and full precision weights that are trained in the first stage are the same for all training schemes.

The results show that two training schemes improved the ensemble's performance compared to the best single-stage training. Those were the "Two-stage Training" and the "Adopted Real-to-Binary Training with continued training". The other training schemes did not improve the ensemble's performance.

Training Scheme	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
Best single-stage	70.1	71.68	72.17	72.14	72.04
First stage	71.31	72.06	72.06	72.17	72.05
Two-stage Training	70.19	71.88	72.28	72.31	72.2
Real-to-bin.	68.98	70.42	70.72	70.63	70.46
Real-to-bin. + Attention matching	69.24	71.44	71.63	71.58	71.28
Real-to-bin. + cont. training	**70.93**	**72.36**	**72.7**	**72.76**	**72.73**
Joint training of all stages	70.03	71.78	72.3	72.22	72.08

Table 5.8: Results obtained from testing the multi-stage training schemes on $Ensemble_1$. The performance of the ensemble after the first stage of the training and the best single-stage training is added for comparison. All values are stated in %. Standard deviations are listed in Appendix A.1.7.

Training Scheme	2 Bit	4 Bit	8 Bit	32 Bit
Best single-stage	71.95	72.76	72.69	72.59
First stage	74.93	75.15	75.18	75.37
Two-stage Training	**72.61**	**73.13**	**73.21**	**73.35**
Real-to-bin.	70.67	71.26	71.35	70.95
Real-to-bin. + Attention matching	70.79	71.66	71.65	71.10
Real-to-bin. + cont. training	72.21	72.92	72.93	71.92
Joint training of all stages	71.39	72.21	72.25	72.27

Table 5.9: Results obtained from testing the multi-stage training schemes on $Ensemble_4$. The performance of the ensemble after the first stage of the training and the best single-stage training is added for comparison. All values are stated in %. Standard deviations are listed in Appendix A.1.8.

For $Ensemble_1$ the "Adopted Real-to-Binary Training with continued training" results in the highest accuracy. It improves the accuracy of the binary model by 0.62%. The accuracy of the 32-bit model increases by 0.69%. This is the biggest performance increase on the $Ensemble_1$. The performance gains when using the "Two-stage Training" were substantially smaller. For example, the accuracy of the binary model only increased by 0.09%.

The "Two-stage Training" performs best for $Ensemble_4$. The training scheme increases the performance of the target bit-width by 0.37% compared to the best single-stage training. The 32-bit model achieves the highest accuracy increase of 0.76%. The "Adopted Real-to-Binary Training with continued training" also increases the accuracy across all bit-

widths, but the increases are lower than ones obtained from the "Two-stage Training". On the target bit-width, the accuracy increases by 0.16%. The results are discussed in Section 6.3.1.1.

During the experiment, we tested three variations of the "Adopted Real-to-Binary Training". Those were "Adopted Real-to-Binary Training with Attention Matching", "Adopted Real-to-Binary Training with continued training" and "Joint training of all stages". The experiments show that these variations result in better-performing models than applying only the "Adopted Real-to-Binary Training". However, only the "Adopted Real-to-Binary Training with continued training" leads to higher performance than the best single-stage training. Possible reasons are discussed in Section 6.3.1.2.

5.2 Experiments on ImageNet

This section describes the experiment conducted on the ImageNet dataset. We experimented on a subset of ImageNet used for the ILSVRC2012 (Imagenet Large Scale Visual Recognition Challenge 2012). It contains 1,281,167 training images, 50,000 validation images and 100,000 test images. They are sorted into 1000 categories[38].

First, Section 5.2.1 introduces the training hyperparameters and image augmentation techniques used for the experiments. Afterward, the different experiments are described and their results presented. Table 5.10 gives an overview of the conducted experiments.

Experiment	Description	Methods	Section
Training Level of the Teacher	Test whether pretrained or untrained teachers perform better	Section 4.2.2	Section 5.2.2
KD within the Ensemble	Test whether the simple of progressive KD performs better	Section 4.2.3	Section 5.2.3
Multiple Stages	Test whether the multi-stage training scheme that worked best for CIFAR-100 also improves the accuracy of the ensemble on ImageNet	Section 4.3	Section 5.2.4

Table 5.10: Overview of the experiments conducted on the ImageNet dataset. In addition to the experiment descriptions, the table lists the section explaining the used method and the section in which the experiment is conducted. Knowledge distillation is as abbreviated with KD.

The experiments were conducted using the ensembles $Ensemble_1$ and $Ensemble_4$ as developed in Section 5.1.2. We did not evaluate different ensembles, as we did on CIFAR-100, on ImageNet, due to the high computational demand of the training. Training one ensemble with five different bit-widths takes approximately 90 GPU hours. This training was conducted on an NVIDIA DGX A100 workstation with a Dual AMD Rome 7742 CPU and an NVIDIA Tesla A100 Tensor Core GPU.

The first two experiments, "Training Level of the Teacher" and "KD within the Ensemble", were conducted analogously to the previous section's experiments on the CIFAR-100 dataset. The approaches were tested using a *ResNet-18* teacher. Additionally, we used a *ResNet-101* teacher to test the different approaches for $Ensemble_1$. We did not use this teacher for the other ensemble due to the long training time. Training the large teacher and the ensemble results in a total training time of 200 GPU hours.

5.2.1 Hyperparameters and Image Augmentation

This section describes the training hyperparameters and image augmentation applied during testing and training for the ImageNet dataset.

Training Hyperparameters: The ensemble was trained for 60 epochs using a batch size of 128. The learning rate was initialized to 0.002. During training, we reduced it gradually using cosine learning rate scheduling. We used the RAdam optimizer[25] and applied gradient canceling at a threshold of 1.3.

For teachers, we used pretrained models from the "gluon model zoo"[5]. The models were trained using an initial learning rate of 0.4 and cosine learning rate scheduling. The batch size was 1024, and the models were trained for 120 epochs. Five warmup epochs at the start were used. The training used Nesterov Accelerated Gradient (NAG) descent[32] with a momentum of 0.9 for optimization. Additionally, a weight decay of $1 * 10^{-4}$ was applied.

When training the teachers jointly with the ensemble in Section 5.2.2, we had to adjust some of the parameters. First, we changed the number of epochs to 60, so the training time of the teacher fits the training time of the ensemble. Second, we reduced the batch size to 128 due to constraints of the computational resources. Last, we also did not use weight decay when jointly training the teacher with the ensemble. The reason is that adding a weight decay causes the teacher to converge slower. The slow convergence has a negative impact on the success of the knowledge distillation. We tested this in Appendix A.4. The accuracies of the teachers after training are given in table Table 5.11. Our changes to the training hyperparameters resulted in lower accuracy of the teacher trained jointly with the ensemble. However, the accuracy loss on the *ResNet-18* teacher is low enough that we do not expect it to have a high impact on the results of the experiments. Unfortunately, the difference for the *Resnet-101* teacher is higher, which may have a negative influence on the results when training with the untrained teacher.

Teacher	Gluon training accuracy	Our training accuracy
ResNet-18	70.93	66.86
ResNet-101	78.34	68.97

Table 5.11: Overview of the accuracies reached by the teachers on ImageNet. All values are stated in %.

[5] https://cv.gluon.ai/model_zoo/classification.html

Image Augmentation: We applied image augmentation to the test and the train images. The train images were augmented more to prevent the model from overfitting during training. We used the following techniques:

- **RandomResizedCrop:** Randomly crops the image and resizes it to 224x224 pixels.
- **RandomFlipLeftRight:** Randomly flips the input image left or right
- **Normalize:** Normalizes the input image using the mean and standard deviation of the dataset

The test images got cropped at the center to a size of 224x224. Afterward, the images were normalized. More information on the applied augmentations can be found in the official gluon documentation.[6]

5.2.2 Training Level of the Teachers

This experiment aimed to determine whether using a pretrained or an untrained teacher performs best for training the ensemble on ImageNet. Section 4.2.2 introduces the method. The experiment was conducted by training the ensemble once using a pretrained teacher for knowledge distillation and once training it jointly with the teacher. We used a *ResNet-18* as the teacher and tested the experiment on the two ensembles $Ensemble_1$ and $Ensemble_4$. The results of $Ensemble_1$ are shown in Table 5.12, and the results of $Ensemble_4$ are shown in Table 5.13.

Teacher	Training	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	55.29	62.11	63.03	63.10	63.09
ResNet-18	Pretrained	**55.4**	63.50	65.01	65.10	65.10
ResNet-18	Untrained	54.98	62.67	63.97	64.03	64.04
ResNet-101	Pretrained	54.71	62.38	64.98	65.04	65.06
ResNet-101	Untrained	54.51	**63.7**	**65.05**	**65.15**	**65.16**

Table 5.12: Comparison of the accuracies of the models included in $Ensemble_1$ using pretrained and untrained teachers. Results obtained from training the ensemble without knowledge distillation are added for comparison. All values are stated in %.

Teacher	Training	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	62.00	63.25	63.23	63.99
ResNet-18	Pretrained	**63.43**	**65.59**	**65.68**	**65.86**
ResNet-18	Untrained	62.98	64.82	64.97	65.02

Table 5.13: Comparison of the accuracies of the models included in $Ensemble_4$ using pretrained and untrained teachers. Results obtained from training the ensemble without knowledge distillation are added for comparison. All values are stated in %.

[6] https://mxnet.apache.org/versions/1.8.0/api/python/docs/api/gluon/data/vision/transforms/index.html

First we compare the results obtained from training $Ensemble_1$. The highest accuracy on the target bit-width is obtained when using a pretrained *ResNet-18* teacher. The accuracy of the binary model increases by 0.11% compared to the training without a teacher. Additionally, the pretrained *ResNet-18* performs better, than the untrained *ResNet-18* on all bit-widths. However, when comparing the results of the *ResNet-101* teacher, we can see that the untrained teacher performs better on most bit-widths. It reaches the highest accuracies for $Ensemble_1$ for all bit-widths, except the binary model. With both teachers, the accuracy gained from applying knowledge distillation increases with the bit-width of the model. The highest increase of 2.07% is obtained by the 32-bit model when using an untrained *ResNet-101* teacher.

The results for the $Ensemble_4$ are similar to the results of $Ensemble_1$. Using a pretrained *ResNet-18* teacher provides the best accuracy across all bit-widths. It improves the performance of the four-bit model by 2.34%. The higher bit-widths reached bigger accuracy improvements. The biggest improvement of 2.45% is obtained by the eight-bit model. Using an untrained teacher shows a similar behavior but increases the accuracy by a smaller amount. For the untrained teacher, the highest accuracy increase is obtained by the eight-bit model. The increase amounts to 1.74%.

Comparing the two ensembles when training both ensembles with a pretrained *ResNet-18* teacher, $Ensemble_1$ yields the more accurate two-bit model, while the performance of the other models is better in $Ensemble_4$. However, no model is close to reaching the pretrained teacher's performance of 70.93%.

Through the experiment, we gained the following insights:

- Using knowledge distillation to train the ensemble improves its accuracy.
- Using a pretrained teacher produces better results than training the teacher with the ensemble, when using a *ResNet-18* teacher.
- When using a *ResNet-101* teacher, training the teacher with the ensemble instead of using a pretrained teacher results in higher accuracies for all models in the ensemble, except the binary model.

Section 6.2.2 discusses these results.

5.2.3 Knowledge Distillation within the Ensemble

The experiment is used to determine which knowledge distillation strategy between teacher and student performs better. The two tested methods were "Simple knowledge distillation between student and teacher" and "Progressive knowledge distillation within the ensemble". Both strategies are introduced in Section 4.2.3.

In the experiment, we compared both strategies by training $Ensemble_1$ and $Ensemble_4$ using a pretrained *ResNet-18* teacher. We useed a pretrained teacher, since the pretrained teacher provided the best results in the experiment conducted in Section 5.2.2. $Ensemble_1$ was also trained using a *ResNet-101* teacher. We used a pretrained *ResNet-101* since it reached the highest accuracy on the target bit-width in the experiments in Section 5.2.2. The results of the comparison of $Ensemble_1$ can be seen in Table 5.14, and for $Ensemble_4$ in Table 5.15.

Teacher	KD Mode	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	55.29	62.11	63.03	63.10	63.09
ResNet-18	Simple	**55.4**	**63.50**	**65.01**	**65.10**	**65.10**
ResNet-18	Progressive	49.91	54.33	57.27	59.13	60.29
ResNet-101	Simple	54.71	62.38	64.98	65.04	65.06
ResNet-101	Progressive	46.01	51.63	55.83	58.15	59.48

Table 5.14: Comparison of the accuracies of the models included in Ensemble_1 using simple KD and progressive KD within the ensemble. Results obtained from training the ensembles without knowledge distillation are added for comparison. All values are stated in %.

Teacher	KD Mode	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	62.00	63.25	63.23	63.99
ResNet-18	Simple	**63.43**	**65.59**	**65.68**	**65.86**
ResNet-18	Progressive	58.90	60.27	60.94	61.79

Table 5.15: Comparison of the accuracies of the models included in Ensemble_4, using simple KD and progressive KD within the ensemble. Results obtained from training the ensembles without knowledge distillation are added for comparison. All values are stated in %.

The results show that the application of progressive knowledge distillation significantly harms the performance of all models in the ensemble. The results are worse than the results obtained when using simple knowledge distillation. The ensembles also perform worse than the ensembles trained using no knowledge distillation.

When comparing results using the *ResNet-18* teacher for $Ensemble_1$, the two-bit model suffers the biggest accuracy loss of 7.78% compared to training without knowledge distillation. The 32-bit model loses the least accuracy, with a decrease of 3.97%. Accuracy losses for $Ensemble_4$ are smaller. Again, the two-bit model loses the most accuracy, with a loss of 3.1%. With a loss of 2.2%, the 32-bit model loses the smallest amount of accuracy. The results indicate that the application of progressive knowledge distillation is less harmful to models with higher bit-widths.

When using a *ResNet-101* teacher, progressive knowledge distillation also does not improve the ensemble's performance. The loss in accuracy is even more severe than for the *ResNet-18* teacher. For example, the binary model's accuracy decreases by 9.28%.

The experiment shows that progressive knowledge distillation degrades the accuracy of the models in the ensemble. Section 6.2.2 discusses these insights.

5.2.4 Multi-Stage Training Schemes

We present the experiment conducted to evaluate multi-stage training schemes on the ImageNet dataset in this section. We assessed the two training schemes that produced the best results on the CIFAR-100 dataset. Those are the "Two-stage Training" and the "Adopted Real-to-Binary Training with continued training". We wanted to verify that the schemes that produced good results on CIFAR-100 also work on the more extensive ImageNet dataset.

The trainings used a *ResNet-18* teacher for knowledge distillation. Building upon the insights of the previous experiments, we used a pretrained teacher and simple knowledge distillation between teacher and student models since it produced the best results for the target bit-width on ImageNet. When using an ensemble as a teacher during the "Adopted Real-to-Binary Training with continued training", we used the knowledge distillation using an ensemble teacher introduced in Figure 4.3. The results of the experiment are given in Table 5.16, and Figure 5.3 shows a visualization of the accuracy of the target bit-width.

Ensemble	Training scheme	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
$Ensemble_1$	Best single-stage	55.4	63.5	65.01	65.10	65.10
$Ensemble_1$	First stage	59.61	64.24	64.9	64.99	64.92
$Ensemble_1$	Two-Stage Training	**55.71**	**64.93**	**66.85**	**66.92**	**66.91**
$Ensemble_1$	Real-to-bin. + cont. training	52.88	62.23	64.16	64.15	64.3
$Ensemble_4$	Best single-stage	-	63.43	65.59	65.68	65.86
$Ensemble_4$	First stage	-	64.94	66.22	66.33	66.29
$Ensemble_4$	Two-Stage Training	-	**65.21**	**67.54**	**67.67**	**67.61**
$Ensemble_4$	Real-to-bin. + cont. training	-	62.54	65.75	65.88	65.92

Table 5.16: Comparison of the multi-stage training schemes on ImageNet for $Ensemble_1$ and $Ensemble_4$. All values are stated in %.

The results show that the "Two-stage Training" performs best for both ensembles. It improves the accuracy of all models in the ensemble. The accuracy of the binary model in $Ensemble_1$ is 55.71%, which is an increase of 0.31% compared to the best single-stage training. The biggest accuracy increase in $Ensemble_1$ is obtained by the four-bit model, with an increase of 1.84%. For $Ensemble_4$ the accuracy of the target bit-width increases by 1.95% to 67.54% compared to the best single-stage training. This is the highest increase achieved in the ensemble. The accuracy of the model with bit-width eight increases by the same amount. The performances reached by ensembles trained with the "Two-stage Training" are the best we obtained during our experiments on ImageNet.

The "Adopted Real-to-Binary Training with continued training" performs worse than the "Two-stage Training". No model on both ensembles reached a performance comparable to the "Two-stage Training" when trained with the "Adopted Real-to-Binary Training with continued training". For $Ensemble_1$, the accuracies do not reach the performance of the best single-stage training. For example, the target bit-width of $Ensemble_1$ obtains an accuracy of 52.88%, which is 2.52% lower than the accuracy of the binary model trained

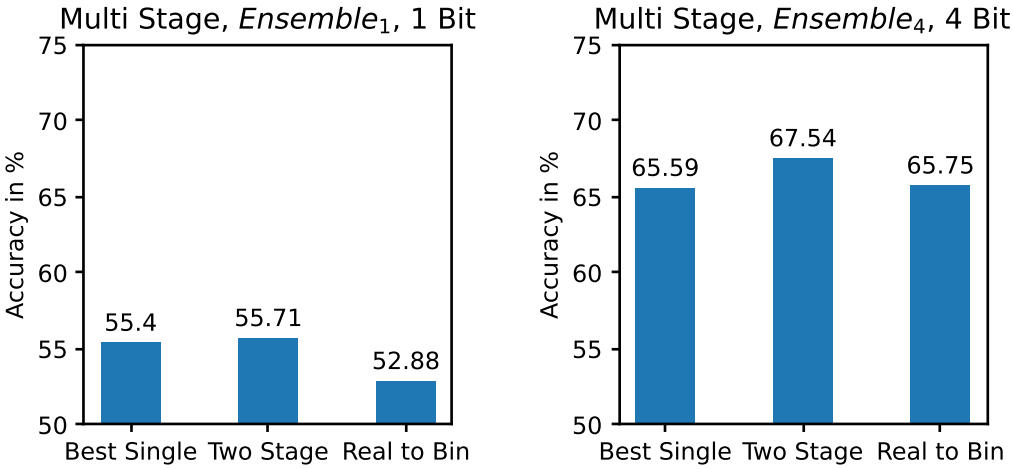

Figure 5.3: Comparison of the accuracy for the multi-stage trainings on ImageNet. Shown are $Ensemble_1$ (left) and $Ensemble_4$ (right). The accuracy of the target bit-width is shown. Accuracy of the best single-stage training is added for comparison.

in a single-stage. For $Ensemble_4$, the performance of the models is comparable to the best single-stage training.

We can summarize the findings of this section in the following points, which are discussed in Section 6.3.2.

- The "Two-Stage Training" results in the best performance on ImageNet for both ensembles.
- The "Adopted Real-to-Binary Training with continued training" does not improve the models' accuracies compared to the single-stage training using a *ResNet-18* teacher.

6 Discussion

This chapter discusses the important findings of the experiments conducted in this work and explains these findings. First, we discuss the setup of the ensemble and the influence of individual models in the ensemble on the other models in Section 6.1. Afterward, in Section 6.2 the single-stage trainings on CIFAR-100 and ImageNet are examined. This includes evaluations on which teachers perform best and analyzing the usefulness of the different knowledge distillation techniques, such as the "Progressive knowledge distillation within the ensemble". Last, we discuss the usefulness of the proposed multi-stage training schemes on both datasets in Section 6.3.

6.1 Discussion of the Ensemble Setup

In Section 5.1.2 we conducted experiments to access the influence a model in the ensemble has on the other models. We suspected the models of influencing each other due to the fact that they share the same set of latent weights.

When testing different combinations of bit-widths for the ensembles in the initial experiments without knowledge distillation in Section 5.1.2, the performance differences of the models were only marginal and never exceeded 1%. The small difference indicates a single model in the ensemble has only a small influence on the other models.

While related literature does not investigate the influence between the models, some report that the inclusion of the binary model degrades the performance of the other models[10] [13]. We can observe this effect when applying knowledge distillation to $Ensemble_1$ and $Ensemble_4$. $Ensemble_1$ includes a model with binary weights, while all other included bit-widths are the same. The plots in Figure 6.1 compare the best results obtained by the ensembles during training in one stage on CIFAR-100 and ImageNet. The results on CIFAR-100 were achieved by training the ensemble jointly with a *ResNet-110* teacher and using progressive knowledge distillation. The ImageNet training used a pretrained *ResNet-18* teacher and simple knowledge distillation.

The results show, that the models from $Ensemble_4$ produce slightly better results, than models from $Ensemble_1$. This is true for all bit-widths, across both datasets, except for the model with bit-width two on ImageNet. The observation that the inclusion of a binary model decreases the performance of other models was also made by Guerra et al. [13] and Du et al. [10]. We can also see that the binary model performs particularly badly compared to the other models in the ensemble. We suspect that the updates on the shared weights done by the poor-performing binary model degrade the performance of the other models. We suspect the accuracy of the binary model in the ensemble to be low, because it is particularly hard to optimize in the ensemble with shared weight. Du et al. [10] gives the following reason: When quantizing weights to a bit-width greater than one, the weights hold a Gaussian-like distribution. However, when quantizing to one bit, the

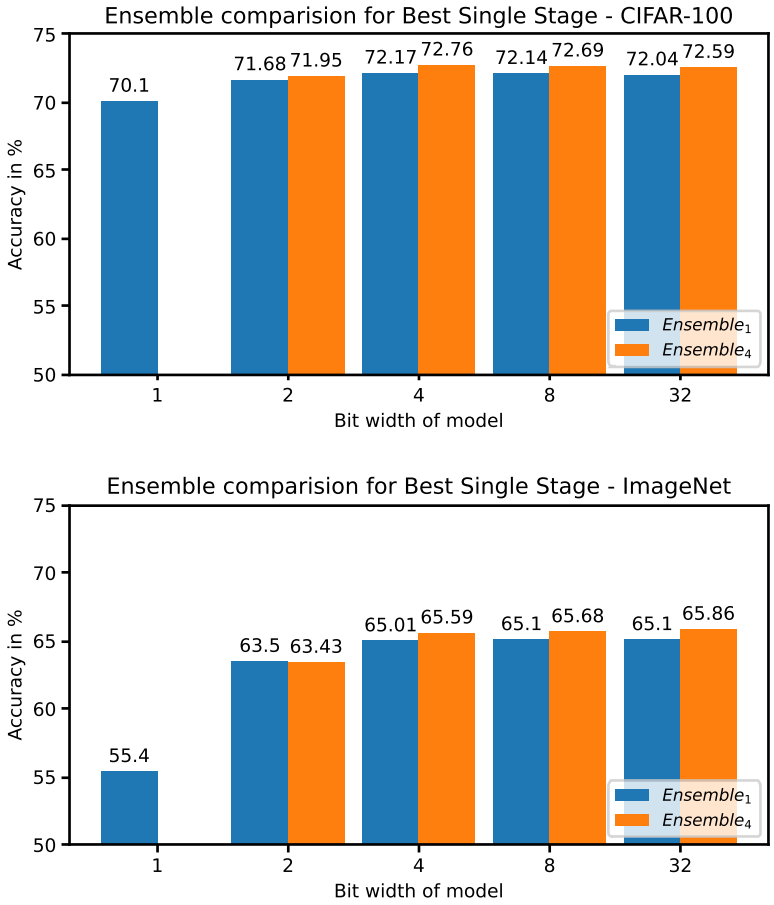

Figure 6.1: Comparison of the accuracy of $Ensemble_1$ and $Ensemble_4$ on CIFAR-100 (top) and ImageNet (bottom). The best trainings using only a single stage are compared. The comparison shows slightly higher accuracies for $Ensemble_4$.

weights hold a Bernoulli distribution. This change in distribution makes it less compatible with the other models in the ensemble and, therefore, harder to optimize.

A different insight we can gather from the experiment is that in some cases, lower precision models in the ensemble reach a higher accuracy than models with higher precision. For example, in the comparison of the CIFAR-100 results in Figure 6.1, we can see that the eight-bit models reach a higher accuracy than the 32-bit models. On ImageNet, we can observe the eight-bit model outperforming the 32-bit model when comparing the results of the "Two-stage Training" as shown in Table 5.16. Usually, we would expect the model with the higher precision to perform better since it has the higher representational power and does not suffer from the quantization error. However, in some cases, quantized models can match the performance of their full precision counterparts [53]. Since the differences are small, we suspect they are caused mainly by randomness in the training process, for example, during the initialization. Furthermore, on CIFAR-100, additional regularization that is introduced by the quantization may also play a role.

6.2 Knowledge Distillation Techniques

This section discusses the experiments conducted to improve the results of the ensembles using knowledge distillation techniques. The discussion includes the experiments regarding the "Training Level of the Teacher" and the "Knowledge Distillation within the Ensemble". Even though we tested both approaches in different experiments, we discuss them together. We do this since we suspect regularization to be an important factor for the effectiveness of both techniques. First, the results of the CIFAR-100 dataset are discussed. Second, we discuss the effects when training on the ImageNet dataset.

6.2.1 Knowledge Distillation Techniques on CIFAR-100

This section discusses the experiments conducted to evaluate knowledge distillation techniques on CIFAR-100. This evaluation includes the experiments for the "Training Level of the Teacher" performed in Section 5.1.3 and the "Knowledge Distillation within the Ensemble" in Section 5.1.4. The section provides insights on why the techniques lead to significant accuracy improvements on CIFAR-100. Afterward, we discuss the performance differences between the different teachers.

6.2.1.1 Discussion of the "Training Level of the Teacher" and "Progressive Knowledge Distillation within the Ensemble"

The results show that using knowledge distillation to train an ensemble improves its accuracy. This is true for all tested ensembles, teachers, and knowledge distillation techniques. Therefore, we could argue that knowledge was successfully transferred from the teacher to the student. Additionally, for the *ResNet-56* and the *ResNet-110* teacher, the accuracy of the models in the ensemble further improves when using an untrained teacher and "Progressive knowledge distillation within the ensemble". This finding indicates that the techniques are more suitable for transferring knowledge from a high-capacity teacher to the students in the ensemble.

However, as Yuan et al.[47] shows, the reason for accuracy improvements when applying knowledge distillation may not be a successful transfer of knowledge but a regularization effect introduced through the usage of soft labels in the loss calculation. The use of soft labels can be considered a smoothing of the target labels[47]. To identify whether regularization is responsible for the success of the knowledge distillation methods on CIFAR-100, we analyzed the effects of the knowledge distillation methods on the train and test accuracies of the ensembles.

In Figure 6.2 we visualized the train and test accuracies on CIFAR-100 for $Ensemble_1$ and $Ensemble_4$ using a *ResNet-110* teacher. Similar visualizations for the *ResNet-20* teacher are shown in Figure 6.3, and plots for the *Resnet-56* teacher are added in Appendix A.3.

The visualization shows that an increase in test accuracy corresponds to a decrease in train accuracy when applying different knowledge distillation techniques. For example, using joint training instead of a pretrained teacher for the $Ensemble_4$ increases the test accuracy by 1.87% while reducing the train accuracy by 15.78%. Since the train accuracy decreases and the test accuracy increases, when using joint training or progressive knowledge distillation, we can argue that they have a stronger regularization effect on the

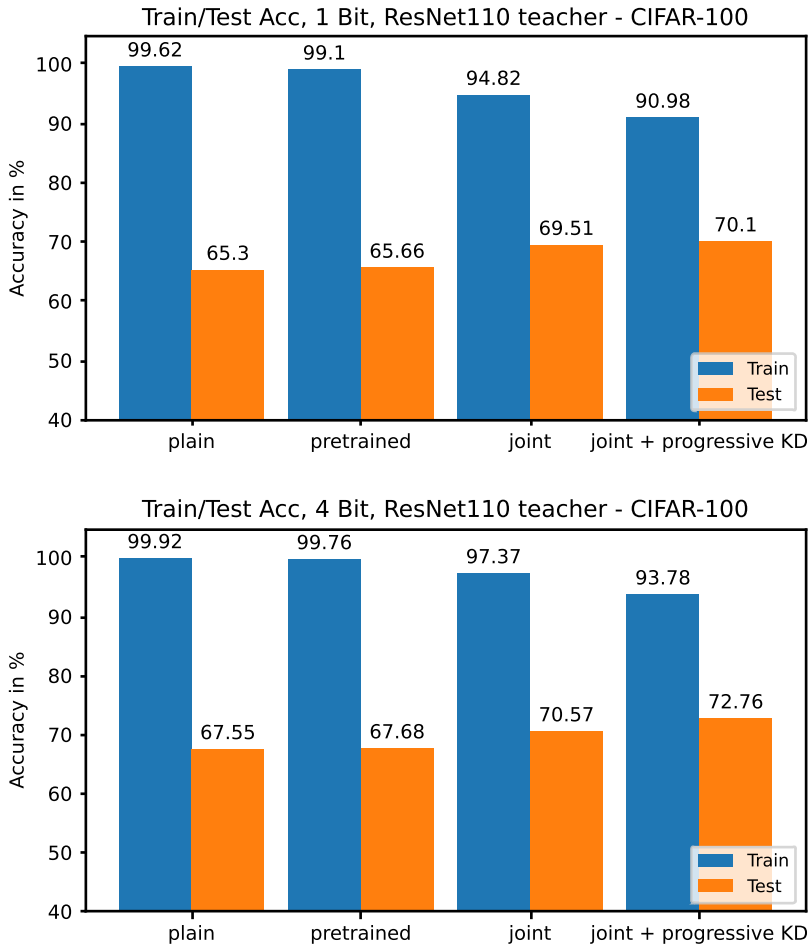

Figure 6.2: Comparison of train and test accuracies for the target bit-width of $Ensemble_1$ (top) and $Ensemble_4$ (bottom) when applying different knowledge distillation strategies on CIFAR-100. The results were obtained using a *ResNet-110* teacher. We see that the different techniques increase test accuracy while the training accuracy decreases.

ensemble and reduce overfitting. Therefore, we suspect that regularization may be the primary reason for the accuracy increase when using joint training of teacher and student or "Progressive knowledge distillation within the ensemble" on CIFAR-100.

There is one exception to this. Using progressive knowledge distillation with a *ResNet-20* teacher decreases train and test accuracy for both ensembles. The corresponding plots are shown in Figure 6.3. With a *ResNet-20* teacher, the train accuracy drops to 79.33%. This is the lowest train accuracy in all experiments on CIFAR-100. In that case, the model may be regularized too much, which leads to underfitting, thus hurting the test accuracy. We give reasons as to why models trained with a *ResNet-20* teacher are stronger regularized in the next section.

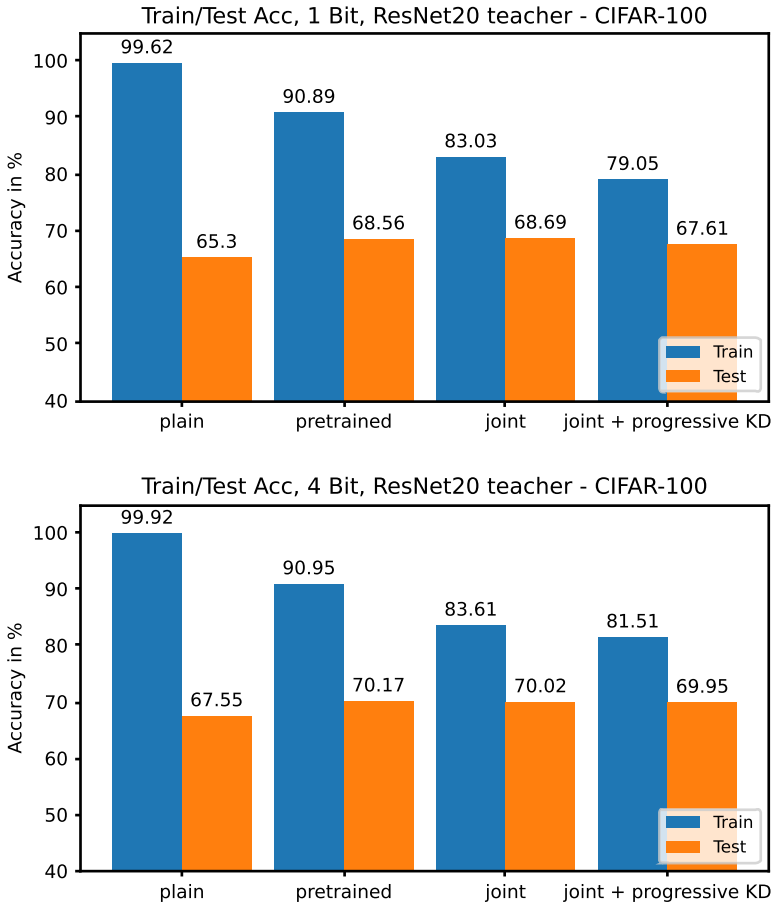

Figure 6.3: Comparison of train and test accuracies for the target bit-width of $Ensemble_1$ (top) and $Ensemble_4$ (bottom) when applying different knowledge distillation strategies on CIFAR-100. The results were obtained using a *ResNet-20* teacher. The test and train accuracy decrease when progressive knowledge distillation is added to the joint training.

6.2.1.2 Differences between Teachers

The experiments on CIFAR-100 were performed using three different teachers with different capacities. Those were a *ResNet-20*, *ResNet-56* and *ResNet-110*. The results of our experiments show that jointly training a teacher and ensemble and the "Progressive knowledge distillation within the ensemble" improve the accuracies of the ensembles trained with a *ResNet-56* or *ResNet-110* teacher. However, these techniques are less effective or degrade the accuracy of the ensembles when the teacher is a *ResNet-20*. Again, we suspect regularization to be the reason for this difference.

As discussed in the previous section, regularization through smoothing of the labels is an important factor for the success of our knowledge distillation techniques in the context of CIFAR-100. The teachers with a high capacity, such as the *ResNet-110*, reache higher test accuracies on the dataset. Therefore, we can assume they have more knowledge than the low-capacity *ResNet-20* teacher. Having a higher accuracy means that more of their predictions are correct, and they are more confident in their predictions. As the teachers become more accurate and confident, the output probabilities match a one-hot encoding of the ground truth label more closely[7]. This means the smoothing effect through soft labels is less pronounced. Thus, the teachers' soft labels provide less regularization. The reduced regularization effect may be the reason why the high-capacity teachers perform worse than the *ResNet-20* teacher when using a pretrained teacher. The *ResNet-20* provides the better regularization and therefore better results.

This changes when using joint training of the teacher and the ensemble or "Progressive knowledge distillation within the ensemble". The two approaches add additional regularization, as discussed in the previous section. The added regularization increases the performance of the ensembles trained with a high-capacity teacher. A model trained with a *ResNet-20* teacher is less prone to overfitting since the predictions of the *ResNet-20* teacher already provide more regularization. Adding additional regularization through the other knowledge distillation techniques may harm the model's accuracy.

6.2.2 Knowledge Distillation Techniques on ImageNet

This section discusses the results of the different knowledge distillation techniques on ImageNet. This includes the "Training Level of the Teacher", which was tested in Section 5.2.2, and "Knowledge Distillation within the Ensemble", which we tested in Section 5.2.3. We focus on two main aspects. First, we discuss why the joint training of student and teacher and the "Progressive knowledge distillation within the ensemble" perform poorly on ImageNet, even though they provided good results on CIFAR-100. Second, we examine the poor performance of the binary model on ImageNet.

6.2.2.1 Performance of Knowledge Distillation Methods

We trained the ensembles $Ensemble_1$ and $Ensemble_4$ on ImageNet using a *ResNet-18* teacher. The results obtained from this teacher are discussed first. Second, we compare these results to the results obtained from the *ResNet-101* teacher and discuss them.

ResNet-18 teacher: When using a pretrained teacher *ResNet-18*, we observe an increase in test accuracy for both ensembles. When jointly training teacher and student or when using "Progressive knowledge distillation within the ensemble", the accuracies of the models decrease. This decrease contrasts the results obtained when conducting the experiments on CIFAR-100. On CIFAR-100, we found regularization to be an important factor as to why the techniques increase the models' performances. To evaluate if this is also the case on ImageNet, we visualized the train and test accuracy of both ensembles in Figure 6.4.

When comparing the train accuracies of the models trained on ImageNet to the models trained on CIFAR-100, we can see that the train accuracy is substantially lower on ImageNet. Additionally, for most models, the test accuracy is higher than the train accuracy

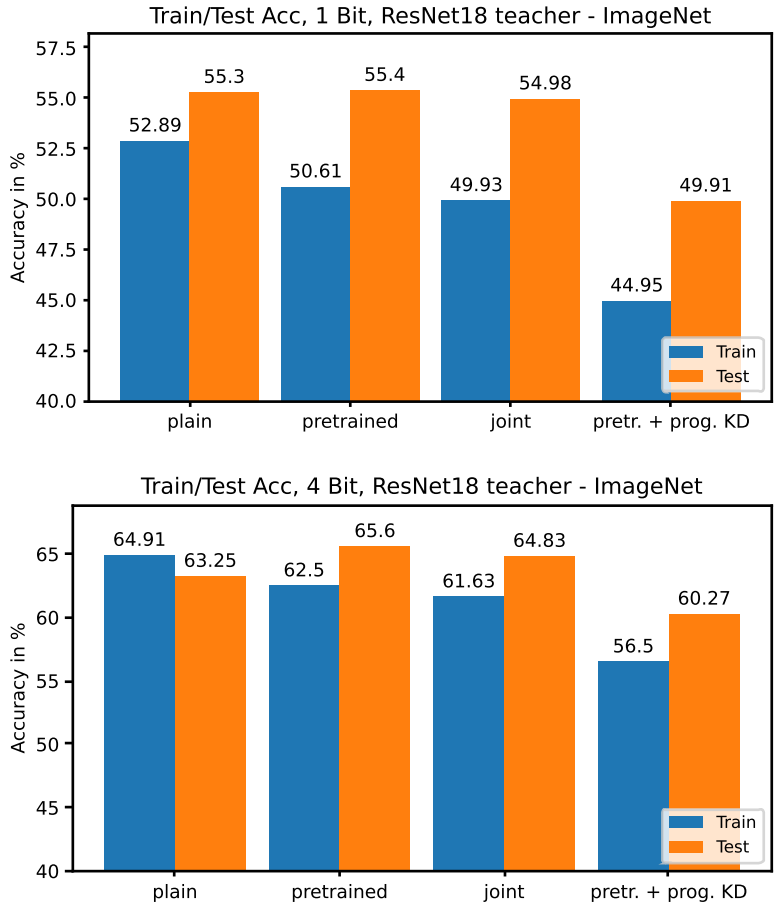

Figure 6.4: Comparison of train and test accuracies for the target bit-width of $Ensemble_1$ (top) and $Ensemble_4$ (bottom) when applying different knowledge distillation strategies on ImageNet. The results were obtained using a *ResNet-18* teacher. The accuracy decreases when using joint training or progressive knowledge distillation.

of the model. We attribute this difference to the fact that more image augmentation is applied to the train images than the test images. Due to the low train accuracy, our model is unlikely to overfit. However, if the model underfits, additional regularization can harm the model's performance. We suspect that the negative influence of the regularization is the reason for the drop in accuracy when jointly training students and teachers or when using "Progressive knowledge distillation within the ensemble". On CIFAR-100, we showed that both techniques have a regularizing effect. On ImageNet, where no additional regularization is needed, this effect degrades the model's performance, decreasing test and train accuracy.

ResNet-101 teacher: When evaluating the "Knowledge Distillation within the Ensemble", the results obtained from the *ResNet-101* teacher are similar to the *ResNet-18* teacher. This means the "Progressive knowledge distillation within the ensemble" performs worse than the "Simple knowledge distillation between ensemble and teacher". However, the results are different when comparing the "Training Level of the teacher". The untrained *ResNet-101* teacher performs better on all bit-widths, except for the binary model. The poor performance of the binary model may be a special case that is discussed in Section 6.2.2.2. Otherwise, the results align with the results obtained from the experiments on CIFAR-100. On CIFAR-100, we show that jointly training the teacher with the ensemble is beneficial when training with a high-capacity teacher. The good performance on ImageNet may indicate that regularization is not the only reason for the good performance of the joint training. The reason is that training on the complex ImageNet dataset does not require as much regularization as the training on CIFAR-100. We assume the joint training makes it easier for the student to mimic the high-capacity teachers, as suspected when presenting the method in Section 4.2.2.

6.2.2.2 Knowledge Distillation for the Binary Model

Our experiments show that the application of knowledge distillation can improve the accuracy of the binary model. The improvements obtained on CIFAR-100 are quite significant, with a difference of 4.81 % between best single stage training and the training without knowledge distillation. However, the improvements on ImageNet are only small. During the single-stage training, the application of knowledge distillation results in an accuracy increase of only 0.11%. As discussed in the previous section, regularization may play a huge role in why the knowledge distillation works well on CIFAR-100. However, the accuracy gains of the binary model on ImageNet are also significantly lower than the improvements of other bit-widths. For example the performance of the two-bit model in $Ensemble_1$ improves by 1.39% when using a pretrained *ResNet-18* as a teacher. These differences in the accuracy gains lead us to believe that there may be other problems with applying knowledge distillation to the binary model on ImageNet.

The problem was observed in other works as well. Zagoruyko et al.[49] did not achieve positive results when applying knowledge distillation on ImageNet. Du et al.[10] simultaneously trains models with different bit-width and shared weights. The binary model performed significantly worse than the other models. Both works stated that capacity differences between students and teachers caused the poor performance of the student models.

On ImageNet, we use the same architecture for the teacher and the student. Therefore, the chosen architecture is not the reason for capacity differences between teachers and students. However, the quantization to a lower bit-width can be seen as reducing the student's capacity. The quantization to binary weights and activations results in the student with the lowest capacity and, therefore, the student with the highest capacity difference between student and teacher. This difference in capacity may hinder the success of the knowledge distillation since a low-capacity student may not be able to mimic the output of high-capacity teachers.

6.3 Discussion of Multi-Stage Training Schemes

In this section, we discuss the results of the experiments conducted to improve the accuracy of $Ensemble_1$ and $Ensemble_4$ using trainings in multiple stages. In the first section, the results of the experiments on the CIFAR-100 dataset are discussed. Afterward, we discuss the experiments on the ImageNet dataset.

6.3.1 Multi-Stage Training Schemes on CIFAR-100

In this section, we discuss the results of the multi-stage training schemes on CIFAR-100 that were conducted in Section 5.1.5. First, we examine why the "Two-stage Training" and the "Adopted Real-to-Binary Training with continued training" are the only training schemes that improved the accuracy of the ensembles compared to the best single-stage training. Afterward, we discuss the results yielded by the different variations of the "Adopted Real-to-Binary Training".

6.3.1.1 Best Performing Training Schemes

The "Two-stage Training" and the "Adopted Real-to-Binary Training with continued training" improve the accuracy of the ensemble compared to the best single-stage training. Both training schemes share a characteristic that distinguishes them from the other approaches. This characteristic is that both approaches use the parameters of the ensemble with full precision weights and quantized activations trained in the first stage to initialize the fully quantized ensemble. We assume that this unique characteristic is why the two approaches perform better than the other multi-stage training schemes. The underlying assumption is that the optimization of the model in two stages reduces the difficulty of training. This assumption has been covered in more detail in Section 4.3. The usefulness of using a model with either the activations or weights quantized as a teacher has already been shown in other related work, such as the work of Zhuang et al.[53].

The "Two-stage Training" results in the highest accuracy for $Ensemble_4$, while $Ensemble_1$ reaches the highest accuracy using "Adopted Real-to-Binary Training with continued training". The two approaches differ in the teacher used during the training of the second stage. The "Two-stage Training" uses a *ResNet-110* teacher, while the "Adopted Real-to-Binary Training with continued training" uses an ensemble with real-valued weights and quantized activations as the teacher. We suspect the "Adopted Real-to-Binary Training with continued training" is better suited to train $Ensemble_1$ because the ensemble used as a teacher provides better guidance for the model with bit-width one. In comparison, the *ResNet-110* provides better guidance for the other bit-widths. Because the *ResNet-110* is the better teacher for the other bit-widths, the final accuracy is higher when the "Two-stage Training" is used to train $Ensemble_4$, where the model with bit-width one is not included. Using the *ResNet-110* for the model with bit-width one results in a lower accuracy compared to using the ensemble as a teacher. Since all the models share the same weights, the worse performing binary model also degrades the performance of the other bit-widths. That poor performing models in the ensemble can degrade the performance of the other models has already been shown by Guerra et al.[13].

6.3.1.2 Other Training Schemes

Next, we want to analyze why the ensembles' performance trained with the other multi-stage training schemes did not surpass the performance of the best single-stage training. Those training schemes are "Adopted Real-to-Binary Training", "Adopted Real-to-Binary Training with attention matching", and "Joint training of all stages". First, it should be noted that the "Adopted Real-to-Binary Training with attention matching" and "Joint training of all stages" both outperform the "Adopted Real-to-Binary Training". These results show that the added attention loss of "Adopted Real-to-Binary Training with attention matching" and the joint training of teacher and student of "Joint training of all stages" improve the ensemble's final accuracy.

The reason neither of these approaches outperforms the single-stage may be that the ensemble with quantized activations and full precision weights is less suitable as a teacher than the *ResNet-110* when training an ensemble from scratch. The reasoning behind using an ensemble as a teacher is that a different teacher teaches each model in the student ensemble. In a model without shared weights, this would be equivalent to training each model in the ensemble independently using the last two stages of the training scheme proposed by Martínez et al.[29]. Since this approach worked well, we suspected the teacher ensemble to give better guidance than a full precision teacher. However, the weight sharing between the different models and the usage of multiple teachers may make this approach less effective. The different teachers in the teacher ensemble provide different soft labels for each model. If there is a big difference between these soft labels, we optimize the same set of shared weights for different targets. For example, one teacher in the ensemble could predict a sample to have a high probability of belonging to class A, while another teacher predicts it to have a high chance of belonging to class B. This disagreement between the teachers results in different gradients for the models taught by the teacher, which makes optimization difficult.

6.3.2 Discussion of the Multi-Stage Training Schemes on ImageNet

In Section 5.2.4 we tested the multi-stage training schemes, that performed best on CIFAR-100, on ImageNet. We discuss the results in this section. Two training schemes were tested. Those were the "Two-stage Training" and the "Adopted Real-to-Binary Training with continued training". We first discuss the "Two-stage Training" before covering the "Adopted Real-to-Binary Training with continued training".

6.3.2.1 Discussion of the "Two-stage Training"

Compared to the best single-stage training, the "Two-stage Training" improves the performance of both ensembles across all bit-widths. The best single-stage training and the "Two-stage Training" both use a pretrained *ResNet-18* and "Simple knowledge distillation between ensemble and teacher". Therefore, the only difference when training the single-stage model and the second stage of the "Two-stage Training" is that the "Two-stage Training" uses the parameters of the ensemble trained in the first phase as a starting point. The results for all models improve when using this ensemble as a starting point. Therefore, we can assume that the "Two-stage Training" effectively enhances the ensemble's

performance on ImageNet by utilizing the parameters of the intermediate model. This finding is consistent with the results from CIFAR-100.
Interestingly, the fully quantized ensemble outperforms the ensemble with full precision weights on all models, except for the binary model. The ensemble with full precision weights has higher representational power than the fully quantized counterpart and should therefore be able to mimic the teacher better. We argue that the ensemble's training with full precision weights was not long enough. When training the second stage, the training is continued, even though the ensemble is now fully quantized. It reaches a better accuracy since it is trained longer.

To verify this, we trained the first stage of $Ensemble_4$ for 120 epochs, which is equivalent to the total training time of the "Two-stage Training". The results are shown in Table 6.1. They show that the longer training time results in a higher ensemble accuracy than the fully quantized ensemble. The longer training time may also be another benefit the "Two-stage Training" has compared to the single-stage training.

Ensemble	2 Bit	4 Bit	8 Bit	32 Bit
First Stage (60 Epochs)	64.94	66.22	66.33	66.29
First Stage (120 Epochs)	**66.98**	**68.41**	**68.51**	**68.49**
Two-Stage	65.04	67.54	67.63	67.59

Table 6.1: Comparison of the longer training of the first stage for $Ensemble_4$ with the normal training time of the first stage and the Two-stage Training.

6.3.2.2 Discussion of the "Adopted Real-to-Binary Training with continued training"

The results of the "Adopted Real-to-Binary Training with continued training" are different from the CIFAR-100 experiment. While the training scheme improves the results on CIFAR-100, applying it on ImageNet does not improve the results compared to the single-stage training. For $Ensemble_1$ the application of the "Adopted Real-to-Binary Training with continued training" even leads to a degradation of the accuracies.

We want to discuss a possible reason for this: The "Adopted Real-to-Binary Training with continued training" and the "Two-stage Training" both use an ensemble with real-valued weights and quantized activations as the starting point for the second stage. Since this works well for the "Two-stage Training", the starting point of the second stage is unlikely to be the reason why the "Adopted Real-to-Binary Training with continued training" performs poorly on ImageNet. The difference between the two approaches is the teacher used to train the second stage. The "Two-stage Training" uses a pretrained *ResNet-18*, while the "Adopted Real-to-Binary Training with continued training" uses the ensemble trained in the first stage as the teacher. The ensemble teacher is probably the reason for the poor performance of the "Adopted Real-to-Binary Training with continued training". In Section 6.3.1.2, we already outlined difficulties when optimizing conflicting predictions of the teachers in the teacher ensemble to be the reason.

Additionally, the performance of the ensemble teacher is low compared to the pretrained *ResNet-18* teacher. The best accuracy reached by a model in the teacher ensemble is 66.33% by the eight-bit model in $Ensemble_4$, while the pretrained *ResNet-18* teacher has an accuracy of 70.93%. We argue that the problems that arise when using an ensemble as a teacher and the poor performance of the ensemble teacher outweigh the benefits of the continued training in the "Adopted Real-to-Binary Training with continued training". Therefore, this approach performs worse than the "Two-stage Training".

7 Evaluation of the Performance of the Ensemble

This section evaluates the performance of the best ensembles trained in this work. To achieve this, we first compare the models trained in the ensemble with individually trained models in Section 7.1. Afterward, we compare the ensemble with the current stage of the art in Section 7.2.

7.1 Comparison with Individually Trained Models

This section compares the ensemble's training with training each model individually. First, we compare the accuracies of the most accurate ensembles with models trained alone. Afterward, we assess the influence of using an ensemble on the training and inference speed as well as memory usage during training.

7.1.1 Comparison of Accuracies

We compare the accuracies of the models in the ensemble with the accuracies of individually trained models in this subsection. The comparison allows us to assess if the training of the models in an ensemble with shared weights influences the accuracies of the final models. We compare our results to models trained on CIFAR-100 and ImageNet.

On CIFAR-100 our experiments show, that using a "Two-stage Training" with an untrained *ResNet-110* teacher and "Progressive knowledge distillation within the ensemble" produces the best results for the $Ensemble_4$ and good results for $Ensemble_1$. The experiments that resulted in these models are described in Section 5.1.5. For the comparison, we trained the individual models with the same technique. The results are shown in Table 7.1.

Training	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
Indiv. Model	69.5	70.38	70.69	70.85	70.46
$Ensemble_1$	**70.19**	71.88	72.28	72.31	72.2
$Ensemble_4$	-	**72.61**	**73.31**	**73.21**	**73.35**

Table 7.1: Comparison of the accuracies of models trained in an ensemble with models trained alone on the CIFAR-100 dataset. All values are stated in %.

Our results show that the models trained in the ensemble outperform the individually trained models for all tested bit-widths. This fact is true for both ensembles. The results indicate that on CIFAR-100, training the models in an ensemble with shared weights is beneficial for the individual models.

We obtained similar results on ImageNet. The individual models and the ensemble were both trained using the "Two-stage Training" with a pretrained *ResNet-18* teacher. We chose this approach because it provided the best results for the ensemble on ImageNet. The corresponding experiment is described in **??**. The related work shows the effectiveness of the "Two-stage Training"[53] and pretrained teacher[31] for training models individually, thus allowing for a good comparison. Table 7.2 shows the results.

Training	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
Indiv. Model	**58.95**	64.79	67.23	67.54	67.55
$Ensemble_1$	55.71	64.93	66.85	66.92	66.91
$Ensemble_4$	-	**65.21**	**67.54**	**67.67**	**67.61**

Table 7.2: Comparison of the accuracies of models trained in an ensemble with models trained alone on the ImageNet dataset. All values are stated in %.

We can see, that $Ensemble_1$ performs worse, than the individual models. Additionally, there is a big difference between the performance of the binary models. The accuracy of the individually trained binary model is 3.24% higher than the accuracy of the binary model trained in the ensemble. We suspect the big performance difference is caused by the difference in the shape of output distributions between the binary model and the other models in the ensemble, which makes it difficult to optimize the binary model in the ensemble. This was discussed in detail in Section 6.1.

The more accurate $Ensemble_4$ outperforms the individually trained models on ImageNet. This shows that when discarding the binary model, the training in an ensemble with shared weights is superior to individual training on ImageNet.

We identified two possible reasons for the superior performance of the ensemble on ImageNet and CIFAR-100. First, for a given model in the ensemble, the updates on the shared weights by the other models may serve as a form of regularization. Second, the lower bit-width models may benefit from the weight updates done by the models with high bit-widths that have a lower quantization error.

7.1.2 Comparison of Resource Consumption

This subsection analyzes the resource demand when training an ensemble with shared weights compared to training individual models. First, we compare the models' size before examining the differences in training speed. Last, we will analyze how quantization affects the speed during inference.

Memory Usage To compare the models' size during training, we examine the number of parameters and memory usage of the ensemble with shared weights. We compare these values to the size of a single model. Additionally, we compare the size of the ensemble to the combined size of all individual models in the ensemble. The single model is a *ResNetE-18* since it is the model used in the ensemble. The comparison is shown in Table 7.3. We use the models contained in $Ensemble_4$.
We can see that the memory used by $Ensemble_4$ is close to the memory usage of a single model. The reason is that all models in the ensemble share the same set of weights. Due

Model	No. of Parameters	Memory Usage
Single Model	11,689,646	44.59MB
All models	46,758,584	44.7MB
$Ensemble_4$	11,718,848	178.48MB

Table 7.3: Comparison of the memory usage of $Ensemble_4$ and individual models during training.

to the shared weights, when adding a new model to the ensemble, the memory usage only increases due to the additional parameters of the batch normalization layer. The reason is that we do not share these layers with the other models. The batch normalization layers for one model contain a total number of 9734 parameters for the *ResNetE-18*. Since there are four models in $Ensemble_4$ the total number of parameters increases by 29,202. The increase reflects the additional batch normalization for three models. This increase in the number of parameters is negligible compared to the increase in parameters when training multiple models without shared weights. Therefore, training the models in an ensemble can be considered an efficient way to reduce memory usage during training.

Training Speed We assess the training speed of our ensemble compared to training each model individually. To achieve this, we compare training our ensemble to training each model in the ensemble one after another. The test is conducted on the CIFAR-100 dataset. We trained the models for 60 epochs without applying knowledge distillation and measured the training time. We train on two GPUs, using a batch size of 64 per GPU. The GPU used is an NVIDIA GeForce 1080 TI. Results are shown in Table 7.4.

Model	Training Time
Single Model	1h 37min
All models	6h 28min
$Ensemble_4$	5h 4min

Table 7.4: Comparison of the training speed of $Ensemble_4$ and individual models. The variations in the training speeds of the individual models were within one minute.

The results show that training the ensemble is faster than training all four models one after another. The time difference is 1h 24min, which is 21.65% of the total training time. We identified two main reasons for this difference in training time. The first reason is the data loading. When training in an ensemble, a batch of data is only loaded once for inference. When training one model after another, the data must be loaded once for each model. Prefetching batches of data can circumvent this problem during training. However, prefetching increases memory usage. The second reason is that when training the ensemble, the gradients of all models are accumulated on the shared weights. Afterward, the weights are updated. When training individual models, the weights of each model have to be updated separately. The additional parameter updates may increase training time.

Inference Speed To access the influence quantization has on the inference speed, we calculated the number of bit operations during inference for each model in the ensemble. To achieve this, we implemented the method proposed by Jin et al.[19]. The method calculates the number of multiply-accumulate instruction operations for each layer and multiplies it with the bit-width of the layer. In addition to the number of bit operations, we also provide the accuracies of the model. They are added because quantization usually decreases inference time while also decreasing the model's accuracy. The decrease in model accuracy is caused by the quantization error as described in Section 2.2. We focus our evaluation on the CIFAR-100 dataset since we reached higher accuracies on it. Additionally, we added data for a comparison on ImageNet in Appendix A.5.

The best models trained on CIFAR-100 were used for the comparison. For $Ensemble_1$ the ensemble was trained using the "Adapted Real-to-Binary Training with continued training". $Ensemble_4$ was trained using the "Two-Stage Training". Both trainings used an untrained *ResNet-110* teacher and "Progressive knowledge distillation within the ensemble". The experiments that resulted in these models are described in Section 5.1.5. The results of our comparison are shown in Table 7.5. We added a comparison on ImageNet in Appendix A.5.

Bit	BitOPs	Acc.($Ensemble_1$)	Acc.($Ensemble_4$)
32	569.5 B	72.73	73.35
8	36.3 B	72.76	73.21
4	9.64 B	72.7	73.13
2	2.97 B	72.36	72.61
1	1.31 B	70.93	-

Table 7.5: Comparison of the number of bit operations during inference using the quantized models and their respective accuracies on CIFAR-100. The accuracies are stated in %.

The analysis shows that quantization results in a significant reduction of bit operations during inference. For example, when quantizing the model to eight bits, only 6.37% of the bit operations of the full precision model are needed to classify an input. When quantizing to binary weights and activations, which is the lowest quantization level, only 0.23% of the bit operations are required.

While the number of bit operations decreases when quantizing, we observed no significant decrease in accuracy for both ensembles when comparing the bit-widths 32, 8, and 4. When quantizing to a bit-width of two, we observe a small drop in accuracy of 0.34% for $Ensemble_1$ and 0.48% for $Ensemble_4$. Binarization results in even lower accuracy, dropping by 1.43% compared to the two-bit model. However, the number of bit operations is also lowered by 56%, making binarization of the model an efficient solution for devices with low computational resources.

7.2 Comparison with the State of the Art

This section compares the accuracy of the ensemble developed in this work with the current state of the art. We focus our comparison on ImageNet because most related work on jointly training models with multiple bit-width only provides results for ImageNet. ImageNet is also used more often in other related work on quantization or knowledge distillation. A short comparison for CIFAR-100 is given at the end of this section.

7.2.1 ImageNet

This section compares our results to other related work on the ImageNet dataset. To achieve a fair comparison, we only compare our results to other results that were obtained using a *ResNet-18* architecture as the trained model. We only compare these results because we only used *ResNet-18* in our experiments, and the size of the model can have a significant impact on the final accuracy. The ensembles in this comparison that we trained use "Two-stage Training" and a pretrained *ResNet-18* teacher since we achieved the best results with this training scheme. The experiment to train these models is described in Section 5.2.4. The comparison with related work can be seen in Table 7.6

Method	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
Real-to-Binary [29]	**65.4**	-	-	-	-
DoReFa [51]	-	62.6	68.1	-	**70.4**
From Quantized DNNs to Quantizable DNNs[10]	-	62.91	66.94	-	67.27
CoQuant [42]	-	57.1	66.6	**67.9**	-
Adabits [19]	-	**65.1**	**69.2**	-	-
Bitmixer [4]	-	64.4	69.1	-	69.6
$Ensemble_1$ (Ours)	55.71	64.93	66.81	66.84	66.86
$Ensemble_4$ (Ours)	-	65.04	67.54	67.63	67.59

Table 7.6: Comparison of the best results of $Ensemble_1$ and $Ensemble_4$ with the current state of the art on ImageNet. Shown are the accuracies in %. Missing values indicate that the approach did not train a model with this bit-width. All trained models use *ResNet-18* architecture. We used the "Two-Stage Training" with a *ResNet-18* teacher to train the ensembles. The results of DoReFa and Adabits are reported by Bulat et al.[4]

Table 7.6 shows that results from this work can compare to some of the related work but do not outperform the current state of the art. The comparison contains four works that simultaneously train models with multiple bit-widths, similar to the ensemble with shared weights used in this work. Those are the works of Du et al.[10], Sun et al.[42], Jin et al.[19] and Bulat et al.[4].

Compared to Du et al.[10] and Sun et al.[42] our ensembles show good performance. $Ensemble_4$ outperforms their models on all bit-widths, except the eight-bit model of Sun et al.[42]. $Ensemble_1$ performs slightly worse. The ensembles also compare well against the individually trained models of Zhou et al[51], except for the 32-bit model.

The results look less positive when comparing them to Jin et al.[19] and Bulat et al.[4]. Jin et al.[19] trained models with the bit-widths two and four, among others. While the results of the two-bit model are comparable to our results, the performance of the four-bit model of Jin et al.[19] is much higher. The accuracy difference is 2.39% when comparing to $Ensemble_1$ and 1.66% when compared to $Ensemble_4$. The result of Bulat et al.[4] is similar to the results of Jin et al.[19]. Additionally, they also provided results for a 32-bit model. This model outperforms the 32-bit models trained in our ensembles. It is 2.01% more accurate than the 32 bit model of $Ensemble_4$. Important differences between these works and ours are the quantization functions used. We used the quantization functions proposed by Zhou et al.[51]. In contrast, Jin et al.[19] uses a modified version of PACT [8] to quantize activations and Bulat et al.[4] uses LSQ [11]. The works on these quantization functions support their usefulness. Therefore, we suspect the usage of these functions may be the reason for the higher accuracies of Bulat et al.[4], and Jin et al.[19]. It should also be noted that the number of training epochs chosen by Bulat et al.[4] and Jin et al.[19] is significantly bigger than our epoch number. While we train our models for 60 epochs, Jin et al.[19] trains them for 150 epochs, and Bulat et al.[4] trains for 160 epochs. The longer training may improve the accuracy of the ensemble. However, our training time was limited by resource constraints.

No approach that simultaneously train models with multiple bit-width reported results for a binary model on ImageNet. Therefore, we compare our results to the results of Martinez et al.[29], which can be considered state of the art for accurate binary neural networks. We can see that the model trained in Martinez et al.[29] surpasses the accuracy of our model by a large margin. The performance difference is 9.69%. In Section 6.1 we discussed why the optimization of the binary model in the ensemble might be challenging. The reason is a difference in the distributions of weights between the binary and the other models. Since the model of Martinez et al.[29] is trained alone, this problem does not appear. We suspect that this may be a major reason for its better performance. Additionally, Martinez et al.[29] employs a variety of other techniques, such as "Real-to-binary attention matching" and "Data-driven channel rescaling", which improve its performance.

7.2.2 CIFAR-100

In this section, we compare the performance of our ensemble on CIFAR-100 to the state of the art. The only work on jointly training models with multiple bit-width that provided results on CIFAR-100 was the work of Du et al.[10]. The comparison of our models with their results is shown in Table 7.7. Our ensembles were trained using "Two-stage Training", an untrained *ResNet-110* teacher, and "Progressive knowledge distillation within the ensemble" as described in Section 5.1.5. Comparing our results to other work on CIFAR-100 is not useful since the majority do not train models with *ResNet-18* architectures.

Method	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
From Quantized DNNs to Quantizable DNNs[10]	64.95	70.5	71.25	-	71.37
$Ensemble_1$ (Ours)	**70.19**	71.88	72.28	72.31	72.2
$Ensemble_4$ (Ours)	-	**72.61**	**73.31**	**73.21**	**73.35**

Table 7.7: Comparison of the best results of $Ensemble_1$ and $Ensemble_4$ with the results of Du et al.citefromquantoquan on CIFAR-100. Shown are the accuracies in %. Missing values indicate that the approach did not train a model with this bit-width. All trained models use *ResNet-18* architecture. We used the "Two-Stage Training" with a *ResNet-18* teacher to train the ensembles.

The comparison shows that our trained models are more accurate than the models trained by Du et al.[10]. The biggest difference can be seen when comparing the binary models. Here, our ensemble is 5.24% more accurate. Du et al.[10] trains their models using knowledge distillation. The approach is comparable to using an untrained *ResNet-18* teacher in our work. Our ensemble is trained using a *ResNet-110* teacher and "Progressive knowledge distillation within the ensemble". Therefore, we suspect those are the main reasons our ensemble performs better. The *ResNet-110* teacher is more accurate, than the *ResNet-18* teacher. The progressive knowledge distillation adds additional regularization.

8 Conclusion and Future Work

In this final chapter, we summarize the findings of our work and outline some limitations. Afterward, we give an overview of interesting future work and give an outlook on the field of research.

The purpose of this work is to solve our research questions, which is to study how knowledge distillation can be used to improve the accuracy of the individual models in an ensemble of models with shared weights, where each model is quantized to a different bit-width. To achieve this, we introduced the ensemble we wanted to train, as well as, the knowledge distillation techniques and training schemes we tested. Next, we conducted experiments to assess the effectiveness of the different approaches on the CIFAR-100 and ImageNet datasets. The results were analyzed and discussed afterward. The experiments resulted in the following key findings:

- Application of knowledge distillation using a pretrained teacher improves the results of the tested ensembles on both tested datasets.
- Using an untrained teacher provides model regularization. Additionally, it improves the results when training small students with high-capacity teachers.
- We showed that the accuracy gains of the binary model when trained in an ensemble were only small on CIFAR-100. On ImageNet, the training in an ensemble degraded the accuracy of the binary model. We identified the main reasons for this to be differences in the distribution of weights between the binary model and the other models and the capacity difference between the binary model and the teacher.
- Including a binary model in the ensemble degrades the performance of the other models in the ensemble due to the negative influence the binary model has on the shared weights.
- We proposed "Progressive knowledge distillation within the ensemble", which provided better regularization than the "Simple knowledge distillation between teacher and ensemble". However, it did not perform well on ImageNet, where regularization is less important in the context of our experiments.
- Training schemes in two stages that use the parameters of intermediate models with quantized activations and full precision weights to initialize the fully quantized model show the best performance on both datasets.
- Using an ensemble with quantized activations and full precision weights as a teacher is inferior to using a single, full precision teacher in most cases. We suspect problems when optimizing conflicting predictions of the teachers on the shared weights to be the reason.
- Models trained in the ensemble achieve higher accuracies than models trained alone when no binary model is trained.

The ensemble with the highest accuracy of the binary model on CIFAR-100 was trained using the "Adopted Real-to-Binary training with continued training" with an untrained *ResNet-110* teacher in the first phase and "Progressive knowledge distillation within the ensemble". As a result, it reached an accuracy of 70.93%. The ensemble with the best performing four-bit model on CIFAR-100 used "Two-Stage Training", an untrained *ResNet-110* teacher, and "Progressive knowledge distillation within the ensemble". This ensemble reached an accuracy of 73.13%. On ImageNet, the best binary model reached an accuracy of 55.71% and was trained using the "Two-Stage Training" with a pretrained *ResNet-18* teacher. The same training scheme was used to train an ensemble with a four-bit accuracy of 67.54%. These models outperform their individually trained counterparts. The only exception to this is the binary model trained on ImageNet. As a result, our ensembles reach accuracies close to the state of the art.

These results show that knowledge distillation can be used to improve the performance of ensembles of quantized models with shared weights. They also show that the decision which knowledge distillation technique to use depends on the dataset used for training. We also confirmed that training models in an ensemble with shared weights is an efficient way to reduce memory usage during training and training time.

Limitations: Besides the promising results of our work, it has some limitations. First, we account for the positive results of the knowledge distillation on CIFAR-100 primarily to regularization effects. It would have been interesting to test the knowledge distillation techniques on models that are already stronger regularized, for example, through additional image augmentation. This test could assess if the tested techniques provide a benefit besides regularization on CIFAR-100. Second, the training on ImageNet has some limitations resulting from computational constraints. We only trained the ensembles for 60 epochs. Other works on this topic train their models for 150[19] or 160[4] epochs. The increased training time may have resulted in a better performing model and may also positively influence the effect of knowledge distillation.

Future Work: Our work offers some opportunities for future research. One approach would be the use of different quantization functions. In our work, we quantized weights and activations as proposed by Zhou et al.[51]. However, many different quantization functions exist, some of which have already been used to train quantized models with shared weights. This includes *PACT* [19] or *Than-Based Quantization* [13]. Other methods, such as *LSQ*[11] may also be suitable. Combining these quantization functions with the knowledge distillation techniques of this work could further increase the accuracy of the ensemble.

We showed that knowledge distillation could be used to improve the models' performances. This result motivates to try new, more complex distillation approaches. For example, one could use a teacher assistant to bridge the gap between student and teacher as proposed by Mirzadeh et al.[30]. Another promising approach was proposed by Yuan et al.[48]. They developed a teacher-free knowledge distillation framework in which students train each other. Using these approaches for our proposed ensembles could be interesting because the different quantized models may be good teachers for each other.

This would eliminate the need for an additional teacher model, which would make the training simpler and faster.
Another field of research for the ensemble could be improving the training algorithm. During training, each model predicts a sample to calculate the gradients for the parameter update. Predicting a sample with each model negatively affects training time. One could vary how samples are passed through the ensemble, for example, by passing samples through a subset of the models at each step. This adjustment to the training makes it faster.

Outlook: Quantization and knowledge distillation are topics where a lot of promising research is done, and its importance may keep growing as the need for fast, lightweight models increases. This growing demand is caused, for example, through applications of these models in the internet of things. We used these techniques to train models with shared weights and different quantization levels. This is an interesting approach to train flexible models that can be deployed to devices with varying resource constraints. Since this provides some practical advantages for the industries that use these models, it is likely, that there will be further research into this topic.

Bibliography

[1] Joseph Bethge, Marvin Bornstein, Adrian Loy, Haojin Yang, and Christoph Meinel. "Training Competitive Binary Neural Networks from Scratch". In: *CoRR* abs/1812.01965 (2018). arXiv: 1812.01965. URL: http://arxiv.org/abs/1812.01965.

[2] Joseph Bethge, Haojin Yang, Christian Bartz, and Christoph Meinel. "Learning to Train a Binary Neural Network". In: *CoRR* abs/1809.10463 (2018). arXiv: 1809.10463. URL: http://arxiv.org/abs/1809.10463.

[3] Alexandre Boulch. "ShaResNet: reducing residual network parameter number by sharing weights". In: *CoRR* abs/1702.08782 (2017). arXiv: 1702.08782. URL: http://arxiv.org/abs/1702.08782.

[4] Adrian Bulat and Georgios Tzimiropoulos. "Bit-Mixer: Mixed-precision networks with runtime bit-width selection". In: *CoRR* abs/2103.17267 (2021). arXiv: 2103.17267. URL: https://arxiv.org/abs/2103.17267.

[5] Han Cai, Chuang Gan, and Song Han. "Once for All: Train One Network and Specialize it for Efficient Deployment". In: *CoRR* abs/1908.09791 (2019). arXiv: 1908.09791. URL: http://arxiv.org/abs/1908.09791.

[6] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. "A Survey of Model Compression and Acceleration for Deep Neural Networks". In: *CoRR* abs/1710.09282 (2017). arXiv: 1710.09282. URL: http://arxiv.org/abs/1710.09282.

[7] Jang Hyun Cho and Bharath Hariharan. "On the Efficacy of Knowledge Distillation". In: *CoRR* abs/1910.01348 (2019). arXiv: 1910.01348. URL: http://arxiv.org/abs/1910.01348.

[8] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. "PACT: Parameterized Clipping Activation for Quantized Neural Networks". In: *CoRR* abs/1805.06085 (2018). arXiv: 1805.06085. URL: http://arxiv.org/abs/1805.06085.

[9] Xiang Deng and Zhongfei Zhang. *Can Students Outperform Teachers in Knowledge Distillation based Model Compression?* 2021. URL: https://openreview.net/forum?id=XZDeL25T12l.

[10] Kunyuan Du, Ya Zhang, and Haibing Guan. "From Quantized DNNs to Quantizable DNNs". In: *CoRR* abs/2004.05284 (2020). arXiv: 2004.05284. URL: https://arxiv.org/abs/2004.05284.

[11] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. "Learned Step Size Quantization". In: *CoRR* abs/1902.08153 (2019). arXiv: 1902.08153. URL: http://arxiv.org/abs/1902.08153.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[13] Luis Guerra, Bohan Zhuang, Ian Reid, and Tom Drummond. "Switchable Precision Neural Networks". In: *CoRR* abs/2002.02815 (2020). arXiv: 2002.02815. URL: https://arxiv.org/abs/2002.02815.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML].

[16] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *CoRR* abs/1704.04861 (2017). arXiv: 1704.04861. URL: http://arxiv.org/abs/1704.04861.

[17] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations". In: *CoRR* abs/1609.07061 (2016). arXiv: 1609.07061. URL: http://arxiv.org/abs/1609.07061.

[18] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167.

[19] Qing Jin, Linjie Yang, and Zhenyu Liao. "AdaBits: Neural Network Quantization with Adaptive Bit-Widths". In: *CoRR* abs/1912.09666 (2019). arXiv: 1912.09666. URL: http://arxiv.org/abs/1912.09666.

[20] Sangil Jung, Changyong Son, Seohyung Lee, JinWoo Son, Youngjun Kwak, Jae-Joon Han, and Changkyu Choi. "Joint Training of Low-Precision Neural Network with Quantization Interval Parameters". In: *CoRR* abs/1808.05779 (2018). arXiv: 1808.05779. URL: http://arxiv.org/abs/1808.05779.

[21] Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. "QKD: Quantization-aware Knowledge Distillation". In: *CoRR* abs/1911.12491 (2019). arXiv: 1911.12491. URL: http://arxiv.org/abs/1911.12491.

[22] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: 2009.

[23] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (Dec. 1989), pages 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541. eprint: https://direct.mit.edu/neco/article-pdf/1/4/541/811941/neco.1989.1.4.541.pdf. URL: https://doi.org/10.1162/neco.1989.1.4.541.

[24] Zewen Li, Wenjie Yang, Shouheng Peng, and Fan Liu. "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects". In: *CoRR* abs/2004.02806 (2020). arXiv: 2004.02806. URL: https://arxiv.org/abs/2004.02806.

[25] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. "On the Variance of the Adaptive Learning Rate and Beyond". In: *CoRR* abs/1908.03265 (2019). arXiv: 1908.03265. URL: http://arxiv.org/abs/1908.03265.

[26] Zechun Liu, Zhiqiang Shen, Shichao Li, Koen Helwegen, Dong Huang, and Kwang-Ting Cheng. "How Do Adam and Training Strategies Help BNNs Optimization?" In: *CoRR* abs/2106.11309 (2021). arXiv: 2106.11309. URL: https://arxiv.org/abs/2106.11309.

[27] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. "Bi-Real Net: Enhancing the Performance of 1-bit CNNs With Improved Representational Capability and Advanced Training Algorithm". In: *CoRR* abs/1808.00278 (2018). arXiv: 1808.00278. URL: http://arxiv.org/abs/1808.00278.

[28] Dhruv Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. "Exploring the Limits of Weakly Supervised Pretraining". In: *CoRR* abs/1805.00932 (2018). arXiv: 1805.00932. URL: http://arxiv.org/abs/1805.00932.

[29] Brais Martínez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. "Training Binary Neural Networks with Real-to-Binary Convolutions". In: *CoRR* abs/2003.11535 (2020). arXiv: 2003.11535. URL: https://arxiv.org/abs/2003.11535.

[30] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. "Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher". In: *CoRR* abs/1902.03393 (2019). arXiv: 1902.03393. URL: http://arxiv.org/abs/1902.03393.

[31] Asit K. Mishra and Debbie Marr. "Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy". In: *CoRR* abs/1711.05852 (2017). arXiv: 1711.05852. URL: http://arxiv.org/abs/1711.05852.

[32] Yurii Nesterov. "A method for solving the convex programming problem with convergence rate $O(1/k^2)$". In: *Proceedings of the USSR Academy of Sciences* 269 (1983), pages 543–547.

[33] Neural Network by sachin modgekar from NounProject.com.

[34] Pc by Icons Producer from NounProject.com.

[35] Phone by Vincencio from NounProject.com.

[36] Raspberry Pi by Ben Davis from NounProject.com.

[37] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks". In: *CoRR* abs/1603.05279 (2016). arXiv: 1603.05279. URL: http://arxiv.org/abs/1603.05279.

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pages 211–252. DOI: 10.1007/s11263-015-0816-y.

[39] Mingzhu Shen, Feng Liang, Chuming Li, Chen Lin, Ming Sun, Junjie Yan, and Wanli Ouyang. "Once Quantized for All: Progressively Searching for Quantized Efficient Models". In: *ArXiv* abs/2010.04354 (2020).

[40] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A. Alemi, and Andrew Gordon Wilson. "Does Knowledge Distillation Really Work?" In: *CoRR* abs/2106.05945 (2021). arXiv: 2106.05945. URL: https://arxiv.org/abs/2106.05945.

[41] Farhana Sultana, Abu Sufian, and Paramartha Dutta. "Advancements in Image Classification using Convolutional Neural Network". In: *CoRR* abs/1905.03288 (2019). arXiv: 1905.03288. URL: http://arxiv.org/abs/1905.03288.

[42] Ximeng Sun, Rameswar Panda, Chun-Fu Chen, Naigang Wang, Bowen Pan, Kailash Gopalakrishnan, Aude Oliva, Rogerio Feris, and Kate Saenko. *Improved Techniques for Quantizing Deep Networks with Adaptive Bit-Widths*. 2021. arXiv: 2103.01435 [cs.CV].

[43] Adapted from Rikiya Yamashita, Mizuho Nishio, Richard Do, and Kaori Togashi. "Convolutional neural networks: an overview and application in radiology". In: *Insights into Imaging* 9 (June 2018). DOI: 10.1007/s13244-018-0639-9.

[44] Haojin Yang, Martin Fritzsche, Christian Bartz, and Christoph Meinel. "BMXNet: An Open-Source Binary Neural Network Implementation Based on MXNet". In: *CoRR* abs/1705.09864 (2017). arXiv: 1705.09864. URL: http://arxiv.org/abs/1705.09864.

[45] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas S. Huang, Xiaodan Song, Ruoming Pang, and Quoc V. Le. "BigNAS: Scaling Up Neural Architecture Search with Big Single-Stage Models". In: *CoRR* abs/2003.11142 (2020). arXiv: 2003.11142. URL: https://arxiv.org/abs/2003.11142.

[46] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. "Slimmable Neural Networks". In: *CoRR* abs/1812.08928 (2018). arXiv: 1812.08928. URL: http://arxiv.org/abs/1812.08928.

[47] Li Yuan, Francis E. H. Tay, Guilin Li, Tao Wang, and Jiashi Feng. "Revisit Knowledge Distillation: a Teacher-free Framework". In: *CoRR* abs/1909.11723 (2019). arXiv: 1909.11723. URL: http://arxiv.org/abs/1909.11723.

[48] Li Yuan, Francis E. H. Tay, Guilin Li, Tao Wang, and Jiashi Feng. "Revisit Knowledge Distillation: a Teacher-free Framework". In: *CoRR* abs/1909.11723 (2019). arXiv: 1909.11723. URL: http://arxiv.org/abs/1909.11723.

[49] Sergey Zagoruyko and Nikos Komodakis. "Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer". In: *CoRR* abs/1612.03928 (2016). arXiv: 1612.03928. URL: http://arxiv.org/abs/1612.03928.

[50] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.

[51] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients". In: *CoRR* abs/1606.06160 (2016). arXiv: 1606.06160. URL: http://arxiv.org/abs/1606.06160.

[52] Bohan Zhuang, Jing Liu, Mingkui Tan, Lingqiao Liu, Ian D. Reid, and Chunhua Shen. "Effective Training of Convolutional Neural Networks with Low-bitwidth Weights and Activations". In: *CoRR* abs/1908.04680 (2019). arXiv: 1908.04680. URL: http://arxiv.org/abs/1908.04680.

[53] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian D. Reid. "Towards Effective Low-bitwidth Convolutional Neural Networks". In: *CoRR* abs/1711.00205 (2017). arXiv: 1711.00205. URL: http://arxiv.org/abs/1711.00205.

A Appendix

A.1 Standard Deviation of CIFAR-100 Experiments

A.1.1 Standard Deviations of Ensemble Setup for *Ensemble$_1$*

bit-widths	1	2	4	8	16	32
1, 2, 4, 8, 16, 32	64.97±0.38	66.19±0.44	66.38±0.45	66.5±0.36	**66.36** ±0.27	66.27±0.56
1, 2, 4, 8, 32	**65.30**±0.04	**66.35** ±0.28	**66.46** ±0.23	**66.54** ±0.11	-	66.31±0.29
1, 2, 8, 32	64.90±0.10	66.14±0.30	-	66.05±0.33	-	66.18±0.33
1, 2, 4, 32	64.94±0.14	66.23±0.38	66.44±0.30	-	-	66.12±0.41
1, 2, 4, 8	65.03±0.31	66.11±0.30	66.45±0.39	66.45±0.44	-	-
1, 4, 8, 32	65.29±0.13	-	66.5±0.21	66.35±0.31	-	**66.57** ±0.39

Table A.1: Accuracies and standard deviation when comparing ensembles with models of different bit-widths to reach high binary accuracy. Values are stated in the format "Accuracy in % ± Standard Deviation * 10^{-1}".

A.1.2 Standard Deviations of Ensemble Setup for *Ensemble$_4$*

bit-widths	WD	1	2	4	8	32
1, 2, 4, 8, 32	No	65.30±0.04	66.35±0.28	66.46±0.23	66.54±0.11	66.31±0.29
2, 4, 8, 32	No	-	65.57±0.33	66.13±0.32	65.89±0.15	65.86±0.29
2, 4, 8, 32	Yes	-	**67.21** ±0.52	**67.55** ±0.70	**67.4** ±0.55	**68.41** ±0.58

Table A.2: Standard Deviation when comparing ensembles with models of different bit-widths with and without weight decay to reach a four-bit accuracy. Values are stated in the format "Accuracy in % ± Standard Deviation".

A.1.3 Standard Deviations of *Ensemble$_1$* for the Training Level of the Teacher

Teacher	Train	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	65.2±0.04	66.35±0.28	66.46±0.23	66.54±0.11	66.31±0.29
ResNet-20	Pretr.	68.56±0.17	70.39±0.13	70.76±0.15	70.74±0.15	70.77±0.14
ResNet-20	Untr.	68.69±0.38	70.34±0.39	70.71±0.25	70.69±0.15	70.70±0.17
ResNet-56	Pretr.	66.24±0.30	68.05±0.40	68.37±0.44	68.4±0.44	68.32±0.47
ResNet-56	Untr.	69.3±0.17	71.0±0.10	71.51±0.23	71.5±0.27	71.6±0.23
ResNet-110	Pretr.	65.66±0.06	67.49±0.10	67.78±0.44	67.79±0.41	67.71±0.47
ResNet-110	Untr.	**69.51**±0.02	**71.48**±0.20	**71.86**±0.23	**71.89**±0.32	**71.91**±0.24

Table A.3: Standard deviations when comparing models included in *Ensemble$_1$* using pretrained (Pretr.) and untrained (Untr.) teachers. Values are stated in the format "Accuracy in % ± Standard Deviation".

A.1.4 Standard Deviations of *Ensemble$_4$* for the Training Level of the Teacher

Teacher	Train	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	67.21±0.52	67.55±0.70	67.4±0.55	68.41±0.58
ResNet-20	Pretr.	69.46±0.26	70.17±0.29	70.23±0.29	70.88±0.23
ResNet-20	Untr.	68.92±0.17	70.02±0.20	70.08±0.26	70.55±0.36
ResNet-56	Pretr.	67.5±0.11	68.19±0.38	68.21±0.31	69.13±0.48
ResNet-56	Untr.	**69.85**±0.05	**70.79**±0.08	**70.8**9±0.14	**71.66**±0.05
ResNet-110	Pretr.	66.57±0.14	67.68±0.26	67.56±0.16	68.8±0.19
ResNet-110	Untr.	69.52±0.11	70.57±0.13	70.65±0.16	71.48±0.03

Table A.4: Standard deviations when comparing models included in *Ensemble$_4$* using pretrained (Pretr.) and untrained (Untr.) teachers. Values are stated in the format "Accuracy in % ± Standard Deviation".

A.1.5 Standard Deviations of *Ensemble$_1$* for the Knowledge Distillation within the Ensemble

Teacher	KD	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	65.2±0.04	66.35±0.28	66.46±0.23	66.54±0.11	66.31±0.29
ResNet-20	Simple	68.69±0.38	70.34±0.39	70.71±0.25	70.69±0.15	70.70±0.17
ResNet-20	Prog.	67.61±0.16	69.1±0.19	69.75±0.06	70.14±0.15	70.24±0.22
ResNet-56	Simple	69.3±0.17	71.0±0.10	71.51±0.23	71.5±0.27	71.6±0.23
ResNet-56	Prog.	69.62±0.23	71.25±0.17	71.69±0.19	71.7±0.22	71.66±0.27
ResNet-110	Simple	69.51±0.02	71.48±0.20	71.86±0.23	71.89±0.32	71.91±0.24
ResNet-110	Prog.	**70.1**±0.34	**71.68**±0.45	**72.17**±0.35	**72.14**±0.24	**72.04**±0.25

Table A.5: Standard deviations when comparing models included in *Ensemble$_1$* using simple KD and progressive KD (Prog.) within the ensemble. Values are stated in the format "Accuracy in % ± Standard Deviation".

A.1.6 Standard Deviations of *Ensemble$_4$* for the Knowledge Distillation within the Ensemble

Teacher	KD	2 Bit	4 Bit	8 Bit	32 Bit
No Teacher	-	67.21±0.52	67.55±0.70	67.4±0.55	68.41± 0.58
ResNet-20	Simple	68.92±0.17	70.02±0.20	70.08±0.26	70.55±0.36
ResNet-20	Prog.	69.29±0.10	69.95±0.23	70.12±0.23	70.45±0.10
ResNet-56	Simple	69.85±0.05	70.79±0.08	70.89±0.14	71.66±0.05
ResNet-56	Prog.	71.26±0.37	72.11±0.25	72.12±0.31	72.12±0.31
ResNet-110	Simple	69.52±0.11	70.57±0.13	70.65±0.16	71.48±0.03
ResNet-110	Prog.	**71.95**±0.15	**72.76**±0.09	**72.69**±0.13	**72.59**±0.11

Table A.6: Standard deviations when comparing models included in *Ensemble$_4$* using simple KD and progressive KD (Prog.) within the ensemble. Values are stated in the format "Accuracy in % ± Standard Deviation".

A.1.7 Standard Deviations of *Ensemble$_1$* for Trainings in Multiple Stages

Training Scheme	1 Bit	2 Bit	4 Bit	8 Bit	32 Bit
Best single stage	70.1±0.34	71.68±0.45	72.17±0.35	72.14±0.24	72.04±0.25
First stage	71.31±0.1	72.06±0.07	72.06±0.14	72.17±0.12	72.05±0.26
Two-stage Training	70.19±0.44	71.88±0.25	72.28±0.31	72.31±0.30	72.2±0.26
Real-to-bin.	68.98±0.36	70.42±0.53	70.72±0.57	70.63±0.54	70.46±0.54
Real-to-bin. + Attention matching	69.24±0.35	71.44±0.30	71.63±0.42	71.58±0.45	71.28±0.34
Real-to-bin. + cont. training	**70.93**±0.21	**72.36**±0.09	**72.7**±0.12	**72.76**±0.11	**72.73**±0.16
Joint training of all stages	70.03±0.33	71.78±0.17	72.3±0.18	72.22±0.01	72.08±0.18

Table A.7: Standard deviations when comparing models included in *Ensemble$_1$* using training schemes in multiple stages. Values are stated in the format "Accuracy in % ± Standard Deviation".

A.1.8 Standard Deviations of *Ensemble$_4$* for Trainings in Multiple Stages

Training Scheme	2 Bit	4 Bit	8 Bit	32 Bit
Best single stage	71.95±0.15	72.76 ± 0.09	72.69 ± 0.13	72.59 ± 0.11
First stage	74.93±0.45	75.15±0.58	75.18±0.49	75.37±0.43
Two-stage Training	**72.61**±0.25	**73.13**±0.16	**73.21**±0.21	**73.35**±0.21
Real-to-bin.	70.67±0.48	71.26±0.38	71.35±0.37	70.95±0.24
Real-to-bin. + Attention matching	70.79±0.44	71.66±0.4	71.65±0.3	71.10±0.21
Real-to-bin. + cont. training	72.21±0.16	72.92±0.17	72.93±0.17	71.92±0.11
Joint training of all stages	71.39±0.16	72.21±0.17	72.25±0.17	72.27±0.06

Table A.8: Standard deviations when comparing models included in *Ensemble$_4$* using training schemes in multiple stages. Values are stated in the format "Accuracy in % ± Standard Deviation".

A.2 Training the First Stage of $Ensemble_4$ Different Weight Decay Values on CIFAR-100

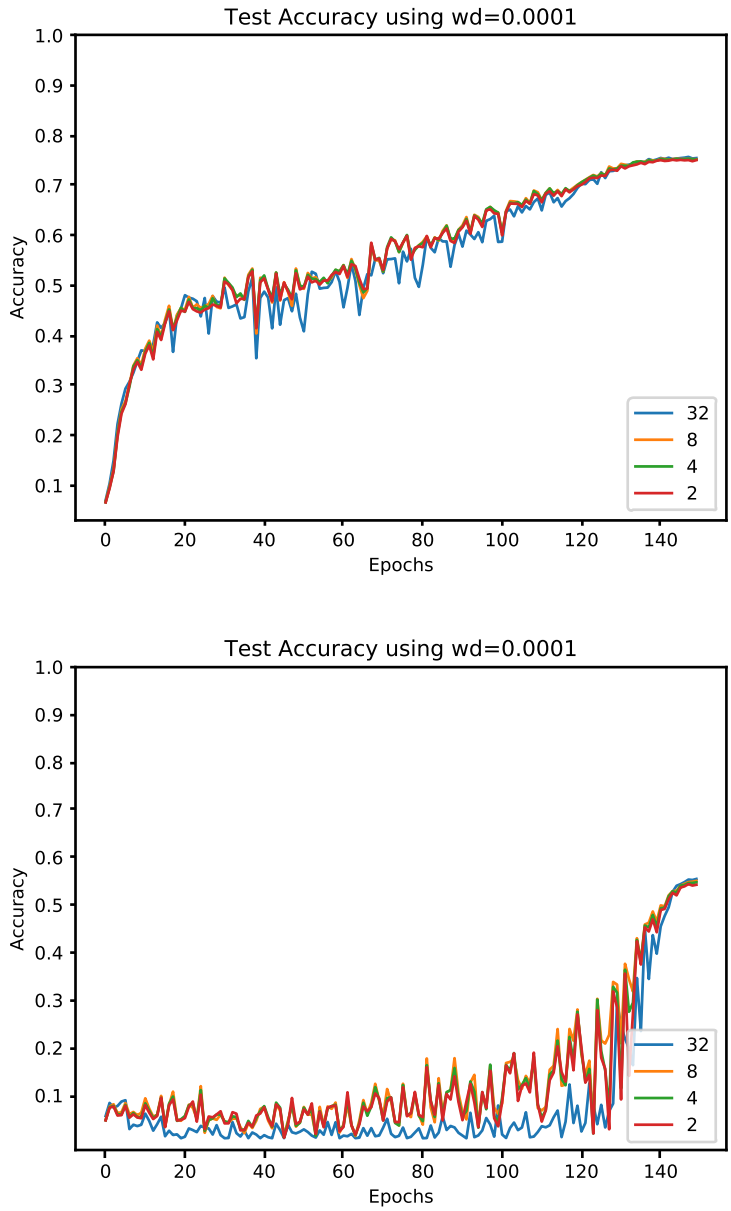

Figure A.1: Learning curve of $Ensemble_4$ with quantized activations and full precision weights on CIFAR-100 using different weight decays. We tested a weight decay of 0.0001 (top) as it is used in the other trainings of the ensemble and weight decay of 0.000002 (bottom).

Weight Decay	2 Bit	4 Bit	8 Bit	32 Bit
No WD	69.53	70.48	70.67	71.19
WD=0.000002	**74.93**	**75.15**	**75.18**	**75.37**
WD=0.0001	55.47	55.06	54.76	54.39

Table A.9: Final results after training $Ensemble_4$ with quantized activations and full precision weights using different weight decay values.

A.3 Comparison of the Ensembles' Train and Test Accuracy with *ResNet-56* Teacher on CIFAR-100

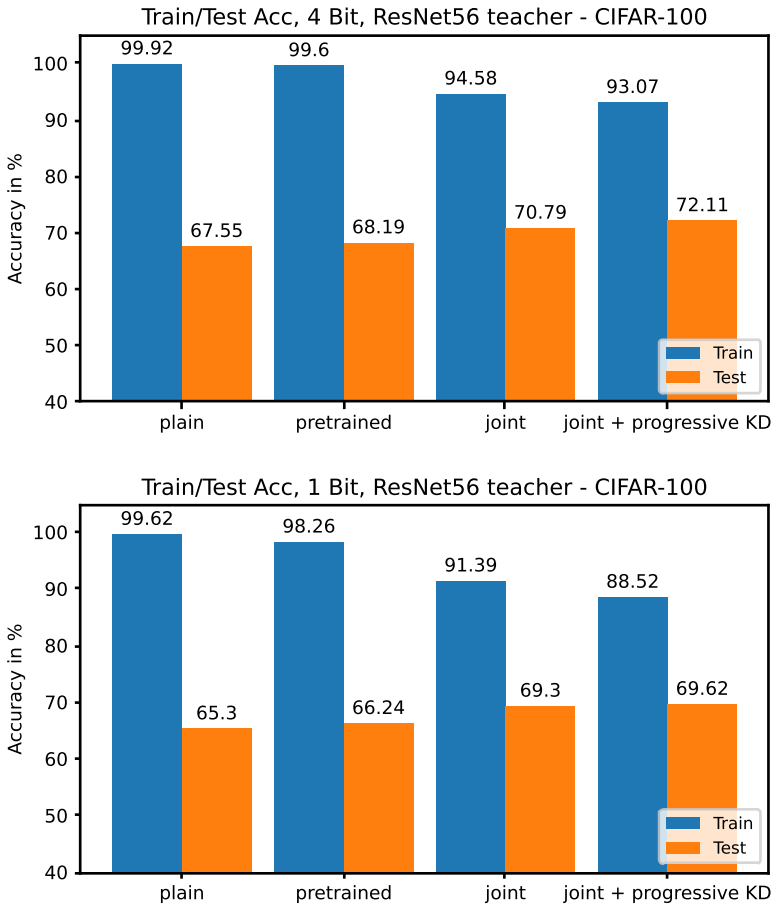

Figure A.2: Comparison of train and test accuracies for the target bit width of $Ensemble_1$ (left) and $Ensemble_4$ (right) when applying different knowledge distillation strategies on CIFAR-100. The results were obtained using a *ResNet-56* teacher. It can be seen, that the different techniques result in an increase in test accuracy, while the train accuracy decreases.

A.4 Weight Decay for *ResNet-18* Teacher when Training the Ensemble

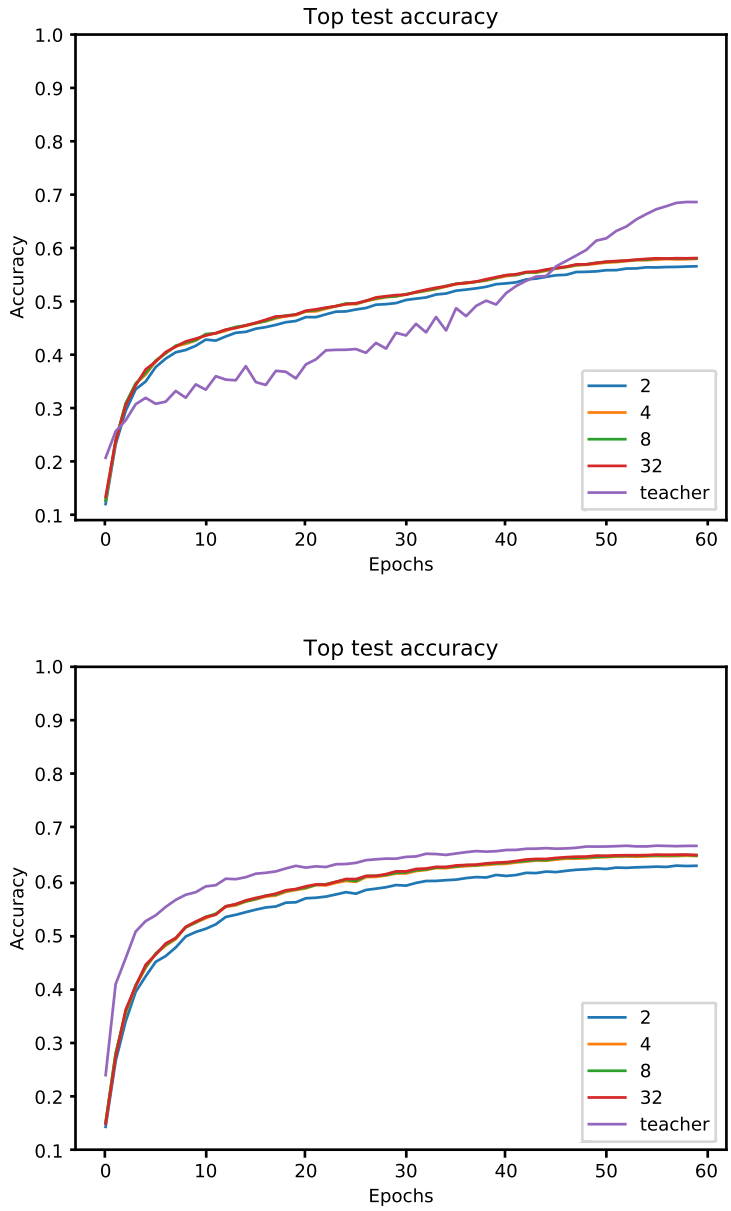

Figure A.3: Learning curve of $Ensemble_4$ and the *ResNet-18* teacher on ImageNet using different weight decays for the teacher. We tested a weight decay of 0.0001 (top) and no weight decay (bottom). Adding weight decay improves the teacher's accuracy, but causes it to converge slower.

Teacher Weight Decay	2 Bit	4 Bit	8 Bit	32 Bit	*Teacher*
No WD	**62.98**	**64.83**	**64.97**	**65.02**	66.67
WD=0.0001	56.67	58.06	58.13	58.21	**68.71**

Table A.10: Final results after training $Ensemble_4$ with a *ResNet-18* teacher with different weight decay values for the teacher.

A.5 Evaluation of Inference Speed and Accuracy on ImageNet

Bit	BitOPs	Acc.($Ensemble_1$)	Acc.($Ensemble_4$)
32	1860.35	66.91	-
8	119.33	66.92	67.61
4	32.28	66.85	67.67
2	10.52	64.93	67.54
1	5.08	55.71	65.21

Table A.11: Comparison of the number of bit operations during inference using the quantized models and their respective accuracies on ImageNet. The accuracies are stated in %. Model were trained using "Two Stage Training", a pretrained *ResNet-20* teacher and "Simple knowledge distillation between ensemble and teacher".

Eidesstattliche Erklärung

Hiermit versichere ich, dass meine Masterarbeit “Training of Ensembles of Quantized Neural Networks with Shared Weights Using Knowledge Distillation” (“Trainieren von Ensembles quantisierter neuronaler Netze mit geteilten Gewichten mittels Wissensdestillation”) selbständig verfasst wurde und dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt wurden. Diese Aussage trifft auch für alle Implementierungen und Dokumentationen im Rahmen dieses Projektes zu.

Potsdam, den 27. Januar 2022,

(Alexander Kromer)