A Study of Response-Based Knowledge Distillation and the Effects of Alpha on Model Accuracy

Nicole Hao

August 2023

Abstract

This report is the culmination of an independent project under Prof. Yunan Yang's guidance. It is a relatively comprehensive study that covers most knowledge distillation (KD) key concepts and some general machine learning concepts. Our exploration spans foundational definitions, papers, and applications. This report also covers an investigation of the effects of alpha in response-based knowledge distillation.

Contents

1 Introduction of Knowledge Distillation (KD)

Before research problem formulation and experimental results, I will first list the preliminary content on knowledge distillation. Some of the following definitions, I will refer to frequently throughout the report; some are just good to know generally, especially when reviewing literature on KD.

1.1 Preliminaries

Definition 1.1 (Knowledge Distillation). In machine learning, knowledge distillation is the process of transferring knowledge from a large complex model (teacher model) or set of models to a single smaller model (the student(s)) that can be practically deployed under real-world constraints [HVD15]. The smaller models can be deployed on mobile phones [PL21]. They are less computationally expensive, and the knowledge distilled from the teacher model, which is better at making accurate predictions, boosts the performance of the student models.

Definition 1.2 (Knowledge). In a neural network context, knowledge typically refers to the learned weights and biases [PL21].

Definition 1.3 (Standard Softmax Function). $\sigma : \mathbb{R}^K \to (0,1)^K$ where $K \geq 1$ is defined by the formula

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

In words, it applies the standard exponential function to each element z_i of the input vector z and normalizes these values by dividing by the sum of all these exponentials. The normalization ensures that the sum of the components of the output vector $\sigma(z)$ is 1 [HVD15].

Example 1.1. For example, the standard softmax of (1,2,8) is approximately (0.001,0.002,0.997), which amounts to assigning almost all of the total unit weight in the result to the position of the vector's maximal element (of 8).

Definition 1.4 (One-hot Coding Labels). One-hot coding and integer labeling are two common methods used to represent categorical data. Each categorical value is converted into a binary vector. For a feature with N unique labels, each label is represented by an N-dimensional vector with one element set to 1 and the rest set to 0.

Example 1.2. For categories Cat, Dog, Bird:

- Cat: [1, 0, 0]
- Dog: [0, 1, 0]
- Bird: [0, 0, 1]

Definition 1.5 (Integer Labels). Each category is assigned a unique integer. Unlike one-hot coding, it uses a single integer value to represent each category.

Example 1.3. For the same categories:

- Cat: 1
- Dog: 2
- Bird: 3

Definition 1.6 (Logit). In a classification neural network, the final layer (before applying softmax) typically consists of linear units that generate scores for each class. These scores are called logits. They can be thought of as the network's confidence in each class before this confidence is converted into a probability distribution [PL21]. In knowledge distillation, logits play a crucial role. The teacher model's logits, which contain rich information about the relationships between different classes, are used to guide the training of the student model. The student model learns to mimic these logits of the teacher model.

Definition 1.7 (Optimum). The best solution among a set of possible solutions. It is often used in the context of optimization problems, where the goal is to find the best solution according to some defined criteria or objective function. If the problem is a minimization problem, the optimum is the minimum solution; if the problem is a maximization problem, the optimum is the maximum solution.

Definition 1.8 (Accuracy Score)**.** An accuracy score is a metric used to evaluate the performance of a classification model. It is defined as the proportion of correct predictions made by the model compared to the total number of predictions. Mathematically, it is expressed as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

This score ranges from 0 to 1, where a score of 1 indicates perfect accuracy (all predictions are correct), and a score of 0 indicates complete inaccuracy (all predictions are incorrect). Accuracy is a straightforward and intuitive measure, often used as a starting point for model evaluation, but it may not always provide a complete picture. Normally accuracy score is used with F1 score, precision, and recall to evaluate the performance of a machine learning model.

Definition 1.9 (Cross-entropy)**.** In the context of classification problems, such as binary or multiclass classification, cross-entropy is often used as a loss function to measure how well a model's predicted probabilities align with the true labels or ground truth. It is used to evaluate the dissimilarity between the predicted probability distribution (often denoted as "p") and the true distribution (often denoted as "y") [PL21].

$$\text{Binary Cross-Entropy Loss: } H(y,p) = -\sum_i (y_i \cdot \log(p_i) + (1-y_i)\cdot \log(1-p_i))$$

Definition 1.10 (Self-distillation)**.** This is the case when the teacher's and student's architectures are identical. The idea builds on the traditional concept of knowledge distillation, where a smaller "student" model learns to replicate the behavior of a larger "teacher" model. In self-distillation, however, the same model acts both as the teacher and the student.

2 Literature Review - Latest Understanding of KD

There are not many papers in applied math about KD (most papers are in the CS category, performing numerical experiments, and so on). I listed some of the latest and most cited papers about KD in or adjacent to applied math, summarized their content, and evaluated their relevance to our project. It turned out that the 2015 Hinton paper was the main one I relied on.

2.1 Distilling the Knowledge in a Neural Network (Hinton et. al, 2015)

The main idea of knowledge distillation, proposed by Geoffery Hinton, Oriol Vinyals, and Jeff Dean in their paper "Distilling the Knowledge in a Neural Network", is to compress the information contained in a large network into a smaller one. This is done by training the smaller network to mimic the output distributions of the larger network. This paper showed that the predictive performance of a model can be improved if it is trained to match the soft targets produced by a large and accurate model.

Essentially, distillation allows for the training of smaller models that are more suitable for deployment in resource-constrained environments, like mobile devices, without a significant loss in accuracy [GYMT21] [PL21]. The experimental results of the paper show that student networks trained with knowledge distillation can outperform those trained conventionally on hard targets, and in some cases, can even approach the performance of the teacher network.

2.1.1 Response-Based KD Architecture

The figure above illustrates the architecture of a response-based knowledge distillation, which is one among many variations of knowledge distillation, such as multi-teacher distillation (one student model and many teacher models) and feature-based knowledge (train the student model to learn the same feature activations as the teacher model) (we list some different types of KD architecture in Section 2.5) [Sun23]. Response-based KD focuses on the final output layer of the teacher model. The hypothesis is that the student model will learn to mimic the predictions of the teacher model. As illustrated in the figure above, this can be achieved by using a loss function, termed the distillation loss, that captures the difference between the logits of the student and the teacher model respectively. As this loss is minimized overtraining, the student model will become better at making the same predictions as the teacher [GYMT21].

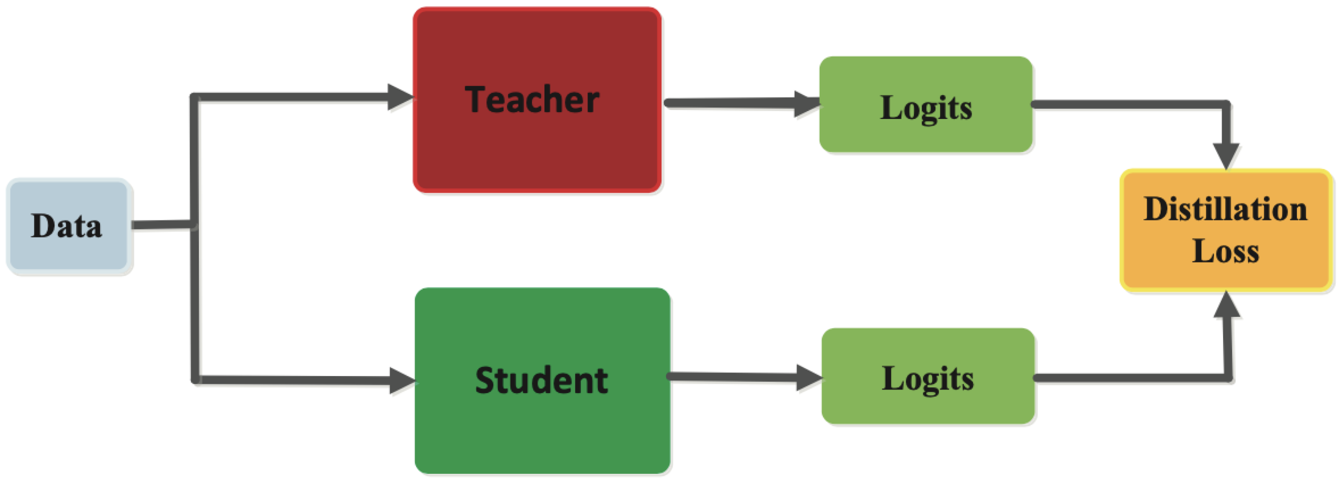

Figure 1: Architecture/Structure of A Response-Based KD System

2.1.2 Teacher Model

The teacher model is a pre-trained large and often complex neural network. This model is usually high-performing, having been trained on a large amount of data and potentially having a very deep or wide architecture. The key characteristic of the teacher model is that it has already learned the task at hand with a high degree of accuracy. It contains valuable knowledge and nuanced understandings of the data, captured within its parameters and activations.

2.1.3 Student Model

The student model is typically a smaller, less complex neural network. The goal of this model is to learn the task by mimicking the teacher model's behavior , but with fewer parameters and a simpler structure. This makes the student model more efficient in terms of computational resources and quicker at making predictions, which is beneficial for deployment in resource-constrained environments like mobile devices or embedded systems.

2.1.4 Hard Targets (labels) & Soft Targets (labels)

In the paper, Hinton proposed using the class probabilities (soft targets) generated by the teacher network as training data for the student network.

Hard targets are the actual labels or ground truth for the training data. In a classification task, they are typically represented as discrete labels indicating the correct class for each input (e.g., "dog", "cat", in a classification task). Hard targets are definitive. They don't convey any information about the model's confidence in its prediction or the relative differences between classes.

Soft targets contain more information per example than hard targets (the actual class labels), leading to more efficient learning. These soft targets are produced by applying the softmax function on the logits (see def 1.6). using a "softmax" output layer that converts the logit, z_i, computed for each class into a probability, q_i, by comparing z_i with the other logits. Here T is the temperature, which is normally set to 1. Using a higher value for T generates a smoother distribution.

$$q_i = \frac{exp(z_i/T)}{\sum_j (z_j/T)} \tag{2.1}$$

In the code implementation, normally the logits of both the teacher model and the student model get converted into soft targets by a final softmax layer. Then the soft targets or softmax predictions of the teacher get converted into hard targets or hard labels (see Section 4).

2.1.5 Mathematical Fomulation

The following is the mathematical formulation of the optimization problem in the Hinton paper. I summarized everything in the form of bullet points. In Section 3 you can see that I adapted this formulation to my own notation.

- Given a finite dataset $\{(a_n, b_n) | n = 1, 2, ..., N\}$, where inputs $a_n \in \mathcal{A}$ (e.g., vectors from $\mathbb{R}^d$ and outputs $b_n \in \mathcal{B}$ (e.g., categorical labels or real numbers).
- A set of models $\mathcal{F} = \{\phi_x : \mathcal{A} \to \mathcal{B} | x \in \mathcal{P} \subseteq \mathbb{R}^d\}$ with fixed neural network architecture parametrized by vector x.
- Loss function: $l : \mathcal{B} \times \mathcal{B} \to \mathbb{R}^+$. The loss associates with a data point (a_n, b_n) and model $\phi_x \in \mathbb{F}$ would be $l(\phi_x(a_n), b_n)$.
- Empirical Risk Minimization (ERM) for this problem is:

$$min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{n=1}^{N} l(\phi_x(a_n), b_n) \tag{2.2}$$

- Teacher model's outputs: $\Phi_\theta(a_n) \in \mathcal{B}$ for each input $a_n \in \mathcal{A}$
- Teacher model $\Phi_\theta : \mathcal{A} \to \mathcal{B}$ but can have different architecture, more layers and parameters.
- Knowledge Distillation with parameter $\lambda \in [0, 1]$ from teacher model Φ_θ to student model ϕ_x is the following modification to 2.2:

$$min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{n=1}^{N} [(1 - \lambda) l(\phi_x(a_n), b_n) + \lambda l(\phi_x(a_n), \Phi_\theta(a_n))] \tag{2.3}$$

- Self-Distillation loss

$$min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{n=1}^{N} [(1 - \lambda) l(\phi_x(a_n), b_n) + \lambda l(\phi_x(a_n), \phi_\theta(a_n))] \tag{2.4}$$

2.2 Generalization bounds in the case of linear and deep linear models: Towards Understanding Knowledge Distillation (Phuong & Lampert, 2019)

- In this paper, Phuong and Lampert derived a characterization of the solution learned by the student. They also proved a **generalization bound** that establishes extremely fast convergence of the risk of distillation-trained classifiers (specifically, it's a bound on the transfer risk, meaningful even in the low-data regime)
- Three key factors that determine the success of distillation:
  1. Data geometry: geometric properties of the data distribution, in particular class separation
  2. Optimization bias: gradient descent optimization finds a very favorable minimum of the distillation objective
  3. Strong monotonicity: the expected risk of the student classifier always decreases when the size of the training set grows
- This paper was more on the confusing side to me, and I didn't end up using any of its content. Still an interesting read, though.

There are several other papers that are worth exploring, however, the focus of this research project primarily requires a close examination of the 2015 Hinton paper, and the other papers were not highly relevant.

2.3 The Impact of Distillation on Stochastic (SGD-based) Optimization: Knowledge Distillation Performs Partial Variance Reduction (Safaryan et al., 2023)

- The paper sheds light on the inner mechanisms of KD by examining it from a mathematical optimization perspective.
- It shows that in the context of linear and deep linear models KD can be interpreted as a novel type of **stochastic variance reduction mechanism** [SPA23].
- Contribution: They provide a detailed **convergence analysis** of the resulting dynamics, which hold under standard assumptions for both **strongly-convex and non-convex losses.**

- This shows that KD acts as a form of partial variance reduction, which can **reduce the stochastic gradient noise**, but not completely depending on the properties of the "teacher" model (what properties?). Because the teacher model may not be an optimum, it might even introduce bias or even increase variance for certain parameter values [SPA23].
- Takeaway: we need to be careful with the parametrization of KD, in particular, w.r.t the weighting of the distillation loss, and is validated empirically on both linear models and deep neural networks. Distillation weight should be used as a means to maximize the positive effects of distillation. For linear models, they can even derive a closed-form solution for the optimal distillation weight. They validate their findings experimentally for both linear models and deep networks, confirming the effects predicted by the analysis.
- Results apply to
  - Self-distillation
  - Distillation for compression (a compressed model leverages outputs from an uncompressed one during training)
- Approach: Re-formulation of distillation in reduction techniques, such as **SVRG** (stochastic optimization).
- This paper was reviewed but not used in the final research problem formulation.

2.4 Distillation and Privileged Information (Generalized Distillation): Unifying distillation and privileged information [LPBSV16]

Although I didn't use this paper in my investigation I thought it was an interesting read. It presents a novel framework based on KD (I think you can say that it's a new type of KD), named Generalized Distillation, which integrates the concepts of KD and learning using privileged information. The key ideas of the paper are as follows:

- Generalized Distillation: The authors introduce Generalized Distillation as a method that allows a student model not only to learn from the teacher's outputs (as in traditional distillation) but also to leverage additional privileged information during training. This approach aims to enhance the learning efficiency and performance of the student model [LPBSV16].
- Theoretical Foundations: The paper provides a theoretical grounding for this approach, discussing how it extends the scope of knowledge distillation. It emphasizes the potential of Generalized Distillation to make learning more effective by using extra information available during the training phase.
- Empirical Evaluation: The paper includes empirical evaluations to demonstrate the effectiveness of Generalized Distillation. These evaluations involve comparing the performance of student models trained with this new method against those trained with standard distillation or without any distillation.

2.5 Other Types of Knowledge Distillation [Sun23]

1. Generalized Distillation: As mentioned in the previous section.
2. Self-Distillation: In this approach, as we mentioned previously in Section 1.1 Preliminaries, the same model acts as both the teacher and the student, learning iteratively from its own softened predictions.
3. Feature-Based Distillation: The student model learns to replicate intermediate representations (features) of the teacher model, not just the final output.
4. Relational Knowledge Distillation: This method focuses on distilling knowledge of the relationships between data points or features, rather than individual predictions.
5. Online Knowledge Distillation: Multiple models are trained simultaneously, and they learn collaboratively by sharing knowledge during training, in contrast to having a pre-trained fixed teacher model.
6. Multi-Teacher Distillation: The student model learns from multiple teacher models, gaining a more comprehensive understanding from diverse knowledge sources.

3 Research Question Formulation & Hypothesis

3.1 Loss Function of the Student Model

Using my own notation, the loss function for the student model with knowledge distillation can be summarized as the following:

$$\min \alpha \underbrace{\left(-\sum_{i=1}^{N} \bar{p_i} \cdot log(y_i)\right)}_{\text{Student loss}} + (1-\alpha) \cdot \underbrace{\left(-\sum_{i=1}^{N} p_i \cdot log(q_i)\right)}_{\text{Distillation loss}} \tag{3.1}$$

- α: confidence
- p_i: ith entry of the softmax predictions of student
- q_i: ith entry of the softmax predictions of teacher
- $\bar{p_i}$: ith entry of the hard prediction of the student (take the highest entry value of the vector p and send it to 1, send the rest to 0.
- y_i: ith entry of the hard true labels
- Distillation loss: measures the difference between soft student predictions, p, and soft teacher labels, q. See cross-entropy formula 1.9.
- Student loss: measures the difference between soft student predictions v and ground truth t

Without Knowledge Distillation ($\alpha = 1$), the loss function for the student model becomes:

$$\min \underbrace{-\sum_{i=1}^{N} \bar{p_i} \cdot log(y_i)}_{\text{Student loss}} \tag{3.2}$$

This means that the model is simply trained on the difference between the predictions of the student model and the hard true labels.

3.2 Research Question

It is observed that this loss function is a linear combination of the distillation loss and student loss, so it acts like interpolation. So our research question is the following:

How does varying the parameter alpha in knowledge distillation affect the accuracy of a student model compared to a model trained from scratch and a model trained with distilled data from a teacher model, especially when using datasets like MNIST for experimentation?

3.3 Hypothesis

We hypothesized that there exists an optimal range for the alpha parameter in knowledge distillation that balances the influence of the teacher's knowledge (distillation loss) and the student's direct learning from data (student loss). This balance, when achieved, is expected to enhance the student model's accuracy, potentially surpassing that of a model trained from scratch and approaching the accuracy of the teacher model.

Based on our analysis of the loss function 3.1, we expect the following results:

- $\alpha = 0.0$: When alpha is set to 1.0, the loss function only considers the distillation loss, and the student loss is completely ignored. This means that the student model will mainly try to mimic the teacher model without much regard for the true labels. As a result, the student model might achieve high accuracy in mimicking the teacher but not achieve the best performance on test data.

- $\alpha = 1.0$: When alpha is set to 0.0, the loss function focuses solely on the student loss, and the soft distillation loss is disregarded. In this case, the student model behaves just like a regular model trained on true labels. It will not benefit from the teacher's knowledge.
- $0.0 < \alpha < 1.0$: Intermediate values of alpha strike a balance between the soft distillation loss and the hard loss. These values allow the student model to leverage knowledge from the teacher while still being influenced by the true labels. The choice of alpha should reflect how much you trust the teacher's knowledge compared to the true labels.

So, if we plot the accuracy scores across different values of alpha, we should observe an initial increase in accuracy and a significant decrease as the alpha values surpass around 0.5.

4 Numerical Experiments

4.1 Data Generation, Processing, and Noise in Training Data

At the beginning stages of this investigation, we tried randomly generating data and then adding Gaussian noise to the labels. However, randomly generating data creates a problem: there is no defined underlying pattern for our deep learning models to pick up. As for the Noise in the training data, we added Gaussian noise initially because it is a common type of noise that follows a Gaussian distribution. Later, we decided to focus on the MNIST data because they have more real-life significance and have an underlying pattern in the training data (see Section 4.2 for an introduction of the MNIST data). We added noise by perturbing the image labels for the training data y_i for all i. For example, if an image has label 4, which means that the image is the number 4, we perturb its label by changing it to 3. There are 10,000 images in our training data, because we still want the teacher model to make accurate predictions, only 500 images, about 5% of the labels of the training data, were perturbed. These perturbed data were all randomly selected from the training set. We eventually evaluated our models on unperturbed test data.

4.2 MNIST Data

In our numerical experiments, we focused on MNIST data. The MNIST dataset is a large collection of handwritten digits commonly used for training and testing in the field of machine learning and computer vision. Comprising 70,000 images, it is divided into a training set of 60,000 examples and a test set of 10,000 examples [Den12]. Each image in the MNIST dataset is a 28x28 pixel grayscale picture of a digit ranging from 0 to 9. These images are simple yet varied, representing a wide range of handwriting styles. This diversity makes MNIST an ideal benchmark for algorithms focusing on image processing, classification, and recognition tasks. The dataset's simplicity and comprehensiveness have made it a standard for those entering the field of machine learning, providing a straightforward way to test algorithms and neural network models. Below are examples of how these digits appear in the dataset: Each image is

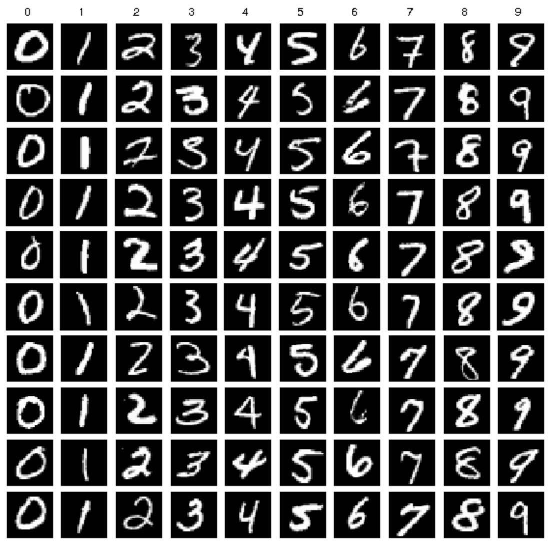

Figure 2: Examples of the MNIST Data, [Den12]

a single digit, and the goal of a typical machine learning model trained on MNIST is to correctly identify these digits. This task tests the model's ability to process visual information and learn patterns in data.

4.3 Code Implementation

This section goes over the structure of the code I used. The code is from the Keras Knowledge distillation tutorial [Tea20], I adapted it to perform numerical experiments by varying alpha values.

4.3.1 The Distiller Class

I defined the `Distiller` class for the process of knowledge distillation in machine learning. The class structure is as follows:

```
class Distiller(keras.Model):
    def __init__(self, student, teacher):
        super().__init__()
        self.teacher = teacher
        self.student = student
```

This initial part defines the constructor of the `Distiller` class. It initializes the class with a student model and a teacher model. The student model is the one being trained, and the teacher model is the pre-trained model from which knowledge is distilled.

```
    def compile(
        self,
        optimizer,
        metrics,
        student_loss_fn,
        distillation_loss_fn,
        alpha=0.1,
        temperature=3,
    ):
        # ... (function definition)...
        super().compile(optimizer=optimizer, metrics=metrics)
        self.student_loss_fn = student_loss_fn
        self.distillation_loss_fn = distillation_loss_fn
        self.alpha = alpha
        self.temperature = temperature
```

The `compile` method configures the distiller. It includes parameters for the optimizer, metrics, student loss function, and distillation loss function. The `alpha` parameter balances the student and distillation loss, and `temperature` is used to soften probability distributions.

```
    def compute_loss(
        self, x=None, y=None, y_pred=None, sample_weight=None, allow_empty=False
    ):
        teacher_pred = self.teacher(x, training=False)
        student_loss = self.student_loss_fn(y, y_pred)

        distillation_loss = self.distillation_loss_fn(
            ops.softmax(teacher_pred / self.temperature, axis=1),
            ops.softmax(y_pred / self.temperature, axis=1),
        ) * (self.temperature**2)

        loss = self.alpha * student_loss + (1 - self.alpha) * distillation_loss
        return loss
```

The `compute_loss` method calculates the loss for the student model. It computes the teacher model's predictions, student loss, and distillation loss. The final loss is a weighted sum of the student and distillation losses as in eqn 3.1.

4.3.2 Teacher Model

The following Python code snippets demonstrate the creation of the teacher model we used for numerical experiments:

```
# Create the teacher
teacher = keras.Sequential(
    [
        keras.Input(shape=(28, 28, 1)),
        layers.Conv2D(256, (3, 3), strides=(2, 2), padding="same"),
        layers.LeakyReLU(negative_slope=0.2),
        layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"),
        layers.Conv2D(512, (3, 3), strides=(2, 2), padding="same"),
        layers.Flatten(),
        layers.Dense(10),
    ],
    name="teacher",
)
```

This code defines the teacher model as a convolutional neural network (CNN) using Keras's Sequential API. The model consists of convolutional layers, activation functions (LeakyReLU), max-pooling layers, and a dense layer. The teacher model is designed to be more complex, with a higher number of convolutional filters (256 and 512).

4.3.3 Student Model

The following Python code snippets demonstrate the creation of the student model we used for numerical experiments:

```
# Create the student
student = keras.Sequential(
    [
        keras.Input(shape=(28, 28, 1)),
        layers.Conv2D(16, (3, 3), strides=(2, 2), padding="same"),
        layers.LeakyReLU(negative_slope=0.2),
        layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"),
        layers.Conv2D(32, (3, 3), strides=(2, 2), padding="same"),
        layers.Flatten(),
        layers.Dense(10),
    ],
    name="student",
)
```

The student model is also a CNN created using Keras's Sequential API. It is less complex than the teacher, with fewer convolutional filters (16 and 32). This reduced complexity makes the student model more computationally efficient, which matches the goal of knowledge distillation to train a lighter model that mimics the performance of a more complex teacher model.

5 MNIST Data Numerical Results

We trained the teacher model on all of the MNIST for 1000 epochs. So the accuracy of the teacher model is 0.988, which means that for 100 images, the teacher model can predict about 98 out of 100 correctly. Due to limited computing power and time, I trained the student model with alpha value 1 for only 100 epochs and achieved a final accuracy of 0.931.

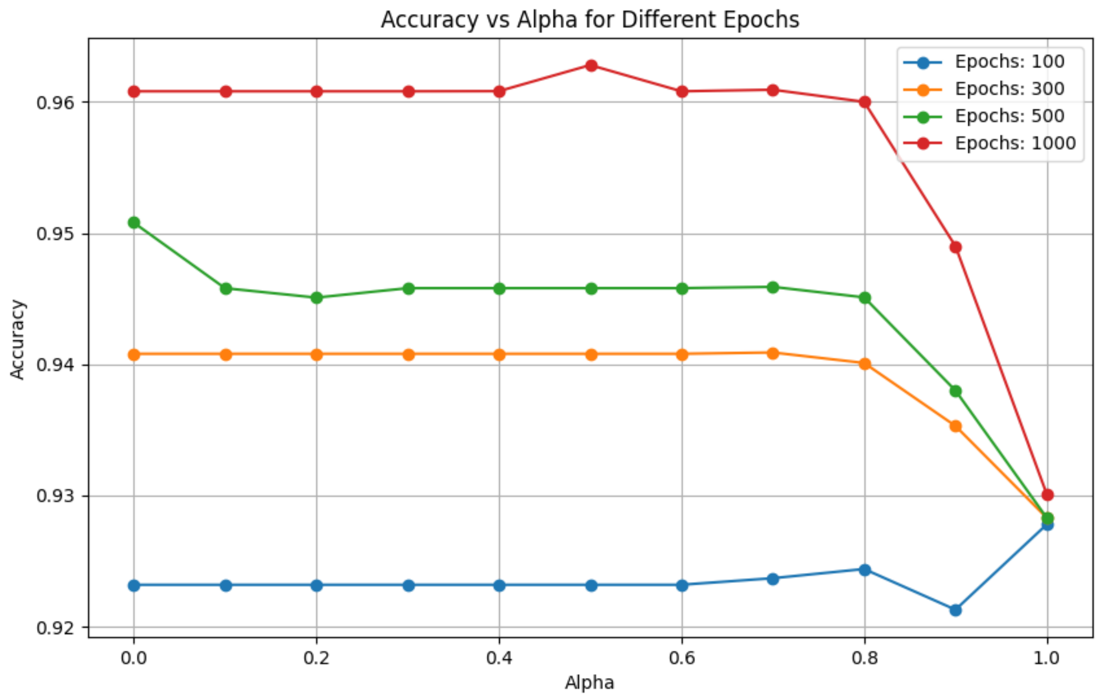

Figure 3: Accuracy Scores across 10 Alpha Values of the Student Model trained with Different Number of Epochs

6 Discussion & Conclusions

As we mentioned in Section 3.3 Hypothesis, as alpha varies, it's supposed to adjust the weight given to the student loss relative to the distillation loss.

A higher alpha value places more emphasis on the student model directly learning from the training data (higher student loss weight), whereas a lower alpha value stresses learning from the teacher model's outputs (higher distillation loss weight). This balance is essential because overly focusing on either can be detrimental: too much weight on the distillation loss might cause the student to replicate the teacher's errors or peculiarities without adequately learning the task itself, while too much weight on the student loss might not fully leverage the teacher's knowledge.

From our results, we can see that as we increase the alpha value past 0.8, the accuracy score decreases significantly, and they seem to converge to a certain value. This supports our hypothesis because if alpha is 1.0, the student model entirely depends on the student loss (see eqn 2.4), which achieved a final accuracy of 0.931 with 100 epochs. Even if we trained the student model with the student loss only for 1000 epochs, the accuracy shouldn't be significantly higher than 0.931 because the student model is a simple model.

Another observation that's worth noting is that the accuracy scores do not experience an initial increase as we hypothesized. For 1000 epochs, the accuracy peaks at alpha = 0.5 but that could be a one-time occurrence. It is hard to us to make any conclusion based on it.

The stability of accuracy also varies across alpha values. For lower epoch counts (100 and 300), the accuracy remains remarkably stable across different alpha values. This suggests that in the early stages of training, the model's performance is less sensitive to the alpha parameter.
Additionally, as the epoch count increases (500 and 1000), there's a noticeable divergence in accuracy across different alpha values. Particularly at 1000 epochs, the model shows a more distinct trend where accuracy peaks at an alpha value of 0.5 and gradually decreases as alpha approaches 1.0. This indicates that, over time, the choice of alpha becomes more critical to the model's performance.

Our theoretical expectation that mid-range alpha value will yield the best results doesn't exactly align with our findings. This discrepancy can arise due to several reasons:

- Model Complexity: If the student model's complexity is not sufficient, it may not effectively learn from the teacher, irrespective of alpha.
- Data Distribution: The specific characteristics of the training data can influence the effectiveness of knowledge transfer.
- Teacher Model's Limitations: If the teacher model has inherent biases or limitations, these can be transferred to the student model.

7 Future Directions

This section outlines potential future directions, building on the findings and observations from the current study:

1. **Experimenting with Varied Teacher Model Quality**: Investigating the impact of the teacher model's quality, such as using less perfectly trained models, on the effectiveness of knowledge distillation and the optimal setting for the alpha parameter. For the experiments, we used a good teacher model that achieved an accuracy score of 0.988. To further verify our hypothesis that the performance converges to a low accuracy score as we increase alpha to 1, we can also try a poor teacher model. For example, we can try a teacher model that's trained for only 2 epochs and run the same experiment we did on the student model.

2. **Trying Different Evaluation Metrics**: We only tried accuracy scores to evaluate model performance, as it is the most commonly used metric in the field of machine learning. It is also worth experimenting with metrics like the value of the loss function since the accuracy score for multi-class classification is weighed by the number of data points in each class. Computing categorical accuracy scores for each class is also a good alternative option.

3. **Exploring Different Data Sets**: Extending the research to include a variety of datasets, particularly those with higher complexity or unique characteristics. We could try different image data sets like fashion MNIST, or generating non-image data using defined functions.

4. **Denoising Effects of Knowledge Distillation and Different Noise Levels**: Analyzing the influence of different types and levels of noise in the training data on the distillation process. When conducting the investigation, I only perturbed about %5 of the training data considering we wanted the teacher model to be good. Alternatively, we can perturb more labels, maybe 10% and 20%, and track these labels to see if the student model, with the distilled knowledge from the teacher, corrects the perturbed labels. If so, KD has a denoising effect on the input data and labels.

5. **Advanced Model Architectures**: We need to further assess the role of alpha in knowledge distillation using more complex neural network architectures, including convolutional and recurrent neural networks, to explore if the findings hold across different model types.

6. **Theoretical Analysis**: Developing a deeper theoretical understanding of how alpha influences learning in knowledge distillation, aiming to uncover new insights or more effective training methodologies. For example, for mean-squared linear regression, we can incorporate ϵ into the loss function for the student model (eqn 3.1). Let $\hat{y_i} = q_i + \epsilon_i \forall i \in N$, the loss function can be written as:

$$\alpha \cdot \underbrace{\left(\frac{1}{n}\sum_{i=1}^{N}(p_i - q_i)^2\right)}_{\text{distillation loss MSE}} + (1-\alpha) \cdot \underbrace{\left(\frac{1}{n}\sum_{i=1}^{N}(p_i - (q_i + \epsilon_i))^2\right)}_{\text{student loss MSE}} \tag{7.1}$$

$$\tag{7.2}$$

Using this formulation will help us better understand the effects of noise on the model performance mathematically.

7. **Computation Power**: When training the student models, because of the size of the MNIST dataset, and the number of epochs chosen, training took very long using my computer's CPU. I was able to use a friend's GPU to increase the time it took. If it were possible to have access to a GPU (from the CS department perhaps), more numerical experiments could be been conducted.

8 References (Honorable Mentions)

Not directly cited in the report, but the following resources help me get a general idea of KD at the beginning stages of the project.

- MIT Han Lab, Lecture 10 - Knowledge Distillation — MIT 6.S965, `https://youtu.be/IIqf-oUTHeO?si=u9y1pmMxSu8yAp6Z`
- Dingu Sagar, Knowledge Distillation in Deep Learning - Basics: `https://youtu.be/gADXP5daZeM?si=7M7H3VT5SPZqgHXK`
- Andrew Ng, Deep Learning Specialization Course `https://www.deeplearning.ai/courses/deep-learning-specializati` `?utm_medium=referral&utm_source=andrew-website`
- Knowledge Distillation: Simplified, Prakhar Ganesh `https://towardsdatascience.com/knowledge-distillation-simpli`
- Chen, Guobin; Choi, Wongun; Yu, Xiang; Han, Tony; Chandraker, Manmohan (2017). "Learning efficient object detection models with knowledge distillation". Advances in Neural Information Processing Systems: 742–751.

References

[Den12] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[GYMT21] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6):1789–1819, June 2021. arXiv:2006.05525 [cs, stat].

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, March 2015. arXiv:1503.02531 [cs, stat].

[LPBSV16] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information, February 2016. arXiv:1511.03643 [cs, stat].

[PL21] Mary Phuong and Christoph H. Lampert. Towards Understanding Knowledge Distillation, May 2021. arXiv:2105.13093 [cs, stat].

[SPA23] Mher Safaryan, Alexandra Peste, and Dan Alistarh. Knowledge Distillation Performs Partial Variance Reduction, December 2023. arXiv:2305.17581 [cs, math].

[Sun23] Sundeep Teki. Knowledge Distillation: Principles, Algorithms, Applications, September 2023.

[Tea20] Keras Team. Keras documentation: Knowledge Distillation, September 2020.