# Investigating Knowledge Distillation

**Sean J. Parker**
Department of Computer Science
University of Cambridge
sjp240@cam.ac.uk

## Abstract

Deep Neural Networks have been developed that can perform image classification with remarkable accuracy, however, such models are often extremely deep, cumbersome models that contain millions of parameters. The models are often too large to be used on mobile and edge devices which have extremely limited resources for performing inference. Some attempts to reduce the sizes of models have been developed such as model compression and quantisation, however, these methods do not reduce the size of the network, rather they reduce the number of weights or number of bits that represent the weights. Knowledge distillation (KD) on the other hand, trains a different, smaller network that can use the 'dark knowledge' from the teacher during training. This paper focuses on investigating the traditional results and implementing a recently proposed method for improving KD when the two models have a vastly different depths. Additionally, we show experimentally, using the MNIST and CIFAR-10 datasets, that the choice of hyperparameters for KD is imperative in distilling the knowledge from the teacher.

## 1 Introduction

With the increased prevalence of machine learning (ML) being used in mobile and lightweight devices, it has become ever more important to reduce the sizes of ML models which brings two key benefits. Firstly, a smaller footprint in storage requirements, and secondly, a less resource-intensive computation when performing a forward propagation. For example, the popular Inception V3 model contains $\sim$24 million params taking up almost 100MB [1]. Following the increased adoption of mobile and IoT devices which demand low energy and compute requirements, inference on large, cumbersome models could prove too costly for such small devices needing to perform object or speech recognition.

Over the past decade, across multiple disciples, including the likes of computer vision and NLP, there is a trend of increasing both model size and complexity. It follows the introduction of affordable compute capability driven by easier access to GPUs and TPUs in cloud services. In turn, a variety of techniques were developed which aim to reduce the sizes of deep learning (DL) models. Some popular techniques such as Quantisation [2] and model pruning [3] show promising results for reducing the size of models, especially important given the rise in popularity of mobile machine learning with the likes of Apple's Siri, Amazon Alexa and Google Predictive Keyboard.

This paper primarily focuses on investigating a field of model compression called Knowledge Distillation which was introduced by Hinton et al. [4] in their seminal paper which described the key motivations and provided a mathematical grounding for KD of neural networks. The main aims of the project are twofold. Primarily, this paper aims to replicate the method described by Hinton et al. [4], for KD of deep neural networks. Secondly, it aims to investigate more recent, advanced, techniques for KD, such as Teacher-Assistant-Student KD, introduced by Mirzadeh et al, [5]. They proposed a method that uses intermediate models to aid in transferring knowledge from large, deep models to far smaller 'Student' models via so-called 'Assistants'.

The rest of the paper is organised as follows. Section 2 provides a brief summary of the background for KD, followed by Section 3 which provides further technical details on the mathematics. Section 4 describes briefly the implementation details for this project which provides sufficient detail for reproduction of results. Section 5 includes an analysis of experimental results, and finally, a brief conclusion is provided in Section 6.

## 2 Background

The concept of Knowledge Distillation (KD) was first introduced by Buciluǎ et al. [6], where the authors described novel model compression techniques. The authors demonstrated that knowledge can be transferred from a large, possibly ensemble, model to a smaller model. KD was further popularised by Hinton et al. [4], where they provided a generalization of model compression. It was a method for distilling learnt information from a large neural network to a smaller network. They found that by the logits of the teacher can be leveraged to guide the smaller network to produce logits matching that of the teacher. Furthermore, they performed experiments in a variety of domains such as image classification on MNIST [7] and JFT [8], as well as speech recognition tasks. In all experiments they found their method learnt more efficiently than training the smaller model from scratch, showing it has learnt the generalised, distilled knowledge from the teacher.

Following the introduction of the Knowledge distillation method, it evolved and was applied to a range of topics, for example, object detection, acoustic models, natural language processing and graph neural networks. KD provides a promising avenue for further research for its application to reducing the size of deep neural networks (DNN) for use on edge devices with a limited compute and require low power consumption.

Furthermore, there has been significant effort applied to aid in stabilizing the student learner, it is often prone to overfitting on the training data. For example, Romero et al. [9], proposed a method for using logits from early layers in the student and combining them in a way that provides more stable gradients. Sau et al. [10] found that by injecting noise in the logits of the teacher, it provides more stable knowledge distillation to the student. Tarvainen et al. [11] described an approach by which the weights of the target predictions can be averaged to improve the accuracy of the student. Finally, there have been many adversarial methods developed by various authors in which a classifier is trained, adversarially, against a discriminator [12, 13].

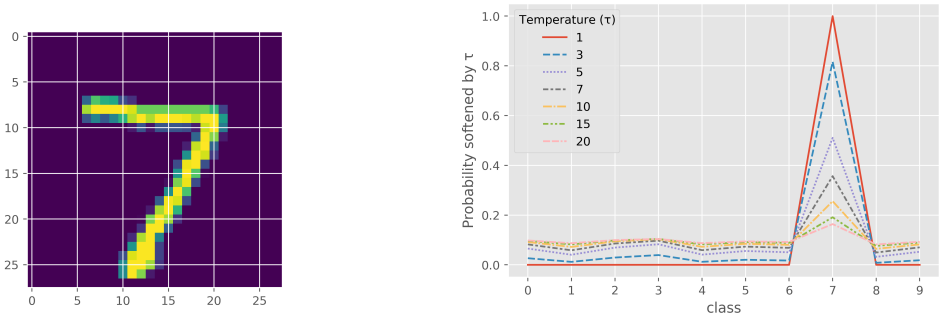
Following the introduction of KD using a 'teacher' and 'student' approach, other researchers investigated the possibility of using intermediate models to bridge the gap between large models and the smaller students. Mirzadeh et al. [5] describe one such approach using the Teacher-Assistant-Student architecture and evaluating the approach using CIFAR-10/100 and ImageNet datasets. They found by carefully choosing the size of a TA (in a 1-step setting), they could achieve an $\sim$5% improvement compared to directly training the student from the teacher, not an insignificant amount, especially in regards to datasets such as ImageNet and CIFAR-100 that are notoriously difficult to achieve high accuracy, even with fine tuned, deep networks.

## 3 Knowledge Distillation

First, we introduce the idea of temperature, in this paper, denoted by $\tau$, which is used to generate so-called 'dark knowledge' that is captured in the logits produced by a deep network. The dark knowledge can be viewed in the small differences between the logits of different classes, it is vitally important for knowledge distillation.

Figure 1 shows how changing the temperature in the softmax function (shown below in Equation (1)) changes the probability distribution across the classes. In Figure 1b below, we can see how this dark knowledge is represented. While for the hard targets, $\tau = 1$, the model is certain of the classification of the digit, when using a higher temperature, the predictions are softer. Additionally, we can also see the rich, similarity structure that exists in the logits. The model has determined that a '7' shares features similar to those of a '3', '0' and '9'. For performing KD, this type of information is critical as we leverage these small differences in probability to generalise the student model.

Knowledge Distillation is a generalised approach to model compression and it can be shown that for large temperatures, model compression is equivalent to KD as both approaches match the logits




(a) MNIST test image of a '7'

(b) Effect of temperature on model predictions

Figure 1: The figure on the left shows a test image of the number 7. The image on the right shows the effect of temperature on the soft targets produced by the model.

of the two models [4, 6], which holds under the assumption of zero-mean. In the original work by Hinton et al. they investigated how temperature, denonted by $\tau$, can be used to create 'soft targets' from the logits of the large model.

$$q_i = \frac{exp(z_i/\tau)}{\sum_j exp(z_j/\tau)} \tag{1}$$

In the field of ML, a classification task is one that, given a set of classes $k_1 \dots k_n$, requires taking some input $\mathbf{x}$ and producing a probability for each class. A common way to achieve this goal is to use a 'softmax' function which computes a probability $q_i$ by comparing a logit, $z_i$, to the logits of the other classes. By setting $\tau = 1$ in (1) we produce the 'softmax' function. When the value of $\tau$ increases, it results in a softer probability distribution of the classes.

Prior to training the student, we define a loss function, (4), that can be used by autograd package in the DL framework of choice to perform backpropagation through the student model. The loss function is comprised of two distinct function. Firstly, the hard targets, $\mathcal{L}_{\text{CE}}$, of the student model are calculated using the logits of the student ($z_s$) and using $\tau = 1$, which is used in the cross entropy loss function with the true labels from the dataset. This is shown in Equation (2).

$$\mathcal{L}_{\text{CE}} = \mathcal{H}(\text{softmax}(z_s), y) \tag{2}$$

$$\mathcal{L}_{\text{KD}} = \tau^2 \text{KL}\left(\text{softmax}\left(\frac{z_s}{\tau}\right), \text{softmax}\left(\frac{z_t}{\tau}\right)\right) \tag{3}$$

$$\mathcal{L}_{\text{student}} = \alpha\mathcal{L}_{\text{CE}} + (1-\alpha)\mathcal{L}_{\text{KD}} \tag{4}$$

Secondly, we calculate the KL-Divergence loss, (3), using the soft targets from the teacher and student, represented by $z_t$ and $z_s$ respectively. Additionally, since Hinton et al. show the magnitudes of gradients produced by soft targets scale with $1/\tau^2$, hence, we multiply the KL-loss by $\tau^2$ to account for the softened gradients. Finally, we combine (2) and (3) by a weighted sum of the two quantities by $\alpha$ and $1-\alpha$ respectively. Furthermore, by using a weighting of $\tau = 0.1$, it will guide the optimisation in the direction towards the softened targets while still taking into account the hard targets produced by the student and the cross entropy loss with the true targets. The aim here is to choose a value of $\tau$ that suitably balances the two criteria. As the student fails to perfectly match the logits of the teacher, we use the hard targets from the student to perform supervised learning using the true labels.

# 4 Implementation

The following section describes the technologies, frameworks, datasets and implementation details for the project.

## 4.1 Datasets

There were two datasets that were chosen for this project, those being MNIST [7] and CIFAR-10 [14]. The MNIST dataset has become known as the "Hello, world" of Machine Learning, it is often used to validate that a machine learning model can learn the generalise before testing the model on much larger, expensive datasets such as COCO or ImageNet. The MNIST dataset contains 70,000 28x28 greyscale images which split into training and test datasets, with 60,000 and 10,000 images respectively. Secondly, the CIFAR-10 dataset contains 60,000 32x32 colour images which are split into 10 unique classes, each of which contain 6000 images. PyTorch (and most major deep learning frameworks) consist of 50,000 training images as well as 10,000 test images.

For the experiments in this project, the both datasets were used in two different experiments. Firstly, the whole dataset was used to perform knowledge distillation from the teacher to the student, and secondly, 3% of the dataset was extracted, with the samples split equally amongst the 10 classes. In terms of preprocessing the dataset, we perform transforms, randomly, to images in the training dataset. The key transform is performing a horizontal flip of the image, as well as normalizing the images using an average mean and standard deviation. Both these techniques are standard techniques to stabilize the training of the models.

## 4.2 Model Definition

Prior to training the student using the Knowledge Distillation methods described in the section above, the teacher model must be trained. For implementing the models, we chose to use the PyTorch [15] deep learning framework. This framework was chosen for its simple API for writing models as well as a simple, readable, structure for training loops. Furthermore, the supplementary package, `torchvision`, provides out-of-the-box support for both the MNIST and CIFAR-10 datasets which support image prefetch and asynchronous transfers of mini-batches to the GPU memory for efficient CUDA operations.

Both the MNIST and CIFAR-10 datasets were used to evaluate the knowledge distillation methodology. Since the MNIST dataset is widely known to be easier to train and achieve high accuracy with shallow networks, the MNIST dataset was used to train a relatively small teacher model (compared to the teacher model for CIFAR-10) that had a structure as follows. Firstly, we use 32 convolution filters with a kernel of size 3 using ReLU activation. This is followed by 64 convolution filters, again using a kernel of size 3 and ReLU activation. Prior to the fully-connected layers, we apply max pooling using a kernel of size 2, followed by a dropout layer with $p = 0.25$. Finally, we apply two linear layers, with 128 nodes, then 10 nodes, each with ReLU activation and a dropout of $p = 0.5$ prior to the final layer. This network results in a network that has 1.2 million parameters that occupy 4.58MB, assuming the weights are stored as a 32-bit floating point number.

The structure of the network for the teacher is based on the ResNet-18 architecture, a detailed overview can be found in the paper by He et al., [16]. The student network structure was chosen to remain as shallow as possible while still maintaining high accuracy of the chosen datasets. Due to the limited computation available, the student network had limited training time (around 30-50 epochs) and using too high a learning rate led to unstable learning. Additionally, to aid in stabilizing learning, both batch normalisation and dropout layers were included in the network. The structure of the student is as follows. Firstly, a convolution layer with 32 filters, followed by batch normalisation and max pooling (with a kernel of size 2). This is followed by another convolution layer, of the same structure, with 64 convolution filters. The second convolution layer connects to a fully-connected network, with 128 nodes and 10 nodes, again, both using ReLU activation and dropout layers preceding the activation layer. In total, this network contains 300,000 parameters, which totals 1.23MB of parameters.

## 5 Experiments

The following section describes and analyses the various experiments performed over the course of the project. We start with providing results using the MNIST dataset, followed by the CIFAR-10 dataset.

Initially, we used the MNIST dataset to experiment with the KD technique. We performed various experiments which have been summarised in Table 1. It shows the test accuracy for each type of model that was trained; the teacher model, described in Section 4.2, shows that we achieved 98.74% accuracy on the 10,000 test images. The parameters had a total size of 1.20MB which represents a 85.8% reduction in model size. Despite the large reduction in size, the student trained with KD (shown as Student + KD in the table) achieves 92.40% accuracy, a ∼6% decrease. On the other hand, the baseline student, trained without KD, achieves 93% accuracy. We found that on MNIST, it was difficult to prevent the student with KD from overfitting even with Batch normalisation and dropout layers providing strong regularisation. However, we observe that when the training was restricted to only use 3% of the training data, around 1800 images, the student with KD outperforms the baseline student.

Table 1: MNIST Test Dataset Accuracy

| Model | Test Accuracy | Model Size (MB) |
|---|---|---|
| Teacher | 98.74% | 1.20MB |
| Student Baseline | 93.00% | 0.17MB |
| Student + KD | 92.40% | |
| Student Baseline (3%) | 83.51% | |
| Student + KD (3%) | 85.30% | |

Table 2: CIFAR-10 Test Dataset Accuracy

| Model | Test Accuracy | Model Size (MB) |
|---|---|---|
| Teacher (ResNet-18) | 83.21% | 42.63MB |
| Student Baseline | 73.85% | 1.23MB |
| Student + KD | 78.14% | |
| Student Baseline (3%) | 41.68% | |
| Student + KD (3%) | 54.36% | |

Table 2 shows the results for experiments using the CIFAR-10 dataset. We performed the same tests as those with MNIST. However, CIFAR-10 leads to more intriguing results. Firstly, of note, the teacher required a much more complex architecture to achieve a reasonable accuracy. The model?s parameters required ∼42MB to store, far greater than that of the MNIST teacher. Despite this, the student network had a size of 1.23MB, a 97% reduction in the size for the parameters. TODO: talk about accuracy results!

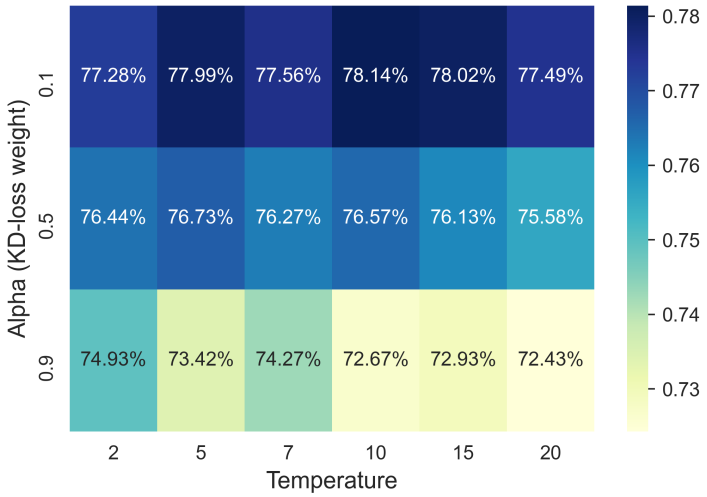



Figure 2: Heatmap showing the effect of temperature and $\alpha$ on test accuracy

Figure 2 shows a heatmap of the results from an experiment we performed that aimed to represent the relationship between temperature and $\alpha$. We can see that the weighting factor $\alpha$ has a strong effect on the resulting test accuracy of the model. In Equation 4 we see that with a higher value of $\alpha$ the model will tend towards optimising for the hard targets from the student output, with less weight placed

on the KL-Divergence loss. During training, for higher values of $\alpha$, the model would often overfit on the training data, possibly due to the small network size of the student. On the other hand, for lower values of $\alpha$, we placed more weight on the soft targets produced by the teacher, leading for the student to try and match the teachers logits.

Another observation we can make is that the temperature has a strong effect on the overall accuracy of the network. While at the extremes of the range tested ($\tau = 2$ and $\tau = 20$), the accuracy of the network suffers but in the middle of the range, around $\tau = 10$, we see the highest test accuracy overall. It is important to note that during training, we observed that while using higher temperatures, the network was more lenient on the learning rate, by increasing the learning rate at higher temperatures, it was possible to achieve better accuracy (often around 2% over the results reported).

## 6 Conclusion

In conclusion, this project has investigated knowledge distillation and its application on Computer Vision tasks. We performed experiments using both the MNIST and CIFAR-10 datasets to investigate properties of the KD approach. In addition, we found that one needs to carefully choose both the temperature and $\alpha$ to allow the student to accurately learn from the dark knowledge of the teacher; the optimum value in this case was using $\tau = 10$ and $\alpha = 0.1$.

Interestingly, we found that we can reduce the size of the CNN models by over 90% without losing a large proportion of the accuracy. Even more promising, when training the student with only an extremely small proportion of the original dataset, 3% in our experiments, the student trained with KD outperformed the student model trained without KD. It is important to note that the major disadvantage to using KD is one needs to have a pre trained teacher model from which we can distill knowledge. The main applications for this technology could be for the distribution of large, deep models to edge devices with a limited computation budget.

Finally, by combining multiple model compression approaches, for example, by performing Qantisation on a trained student model, it would decrease the model size further, however, it may come at the cost of performance. One possible way to approach this is to use methods such as Quantisation-aware training, it may aid to alleviate the loss of accuracy due to using only 8-bits to represent the weights rather than the full 32-bit floating point weights.

## References


[1] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[2] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference, 2017.

[3] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[5] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant, 2019.

[6] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 535–541, New York, NY, USA, 2006. Association for Computing Machinery.

[7] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[8] Revisiting the unreasonable effectiveness of data. `https://ai.googleblog.com/2017/07/revisiting-unreasonable-effectiveness.html`, Jul 2017.

[9] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015.

[10] Bharat Bhusan Sau and Vineeth N. Balasubramanian. Deep model compression: Distilling knowledge from noisy teachers, 2016.

[11] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, 2018.

[12] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge distillation with adversarial samples supporting decision boundary, 2018.

[13] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Kdgan: Knowledge distillation with generative adversarial networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 775–786. Curran Associates, Inc., 2018.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.