Knowledge Distillation in Neural Networks

Chun Hung Lin
chlin3@kth.se

Run Yan Tan
rytan@kth.se

Vivek Chalumuri
vivekc@kth.se

Abstract

We explored the main project objective of distilling knowledge from a well-trained teacher network into a simpler student network, by using soft targets from the former to train the latter. Furthermore, we extended our work by examining a more recent paper on knowledge distillation through model compression, by using the block-wise outputs from the teacher network to train the student network. Finally, as both methods are different knowledge distillation methods, we experimented and made comparisons between them.

1 Introduction

Deep and complex neural networks are often trained to produce accurate results, however these models are often too bulky to deploy in terms of efficient execution time and compact memory requirements. Therefore, there is increased research in the area of knowledge compression while retaining a large part of the original network performance. The underlying objective is to use knowledge from the trained bulky models (teacher models) to train simpler deployable models (student models). This knowledge transfer is known as 'knowledge distillation'. After performing the experiments in the main recommended paper of this project [1], we further explored a more recent and different method [2], and made comparisons between them.

The underlying insight is that when a complex network learns to generalize different classes, the relative probabilities it assigns to each class can be very informative in terms of knowledge of the model. For instance, a well-trained model will be able to learn a better probability distribution of the classes instead of just predicting the correct class. This would mean a model that is able to inform a closer probability between the 'cat' and 'dog' classes and a more distant probability between the 'cat' and 'ship' classes, for example. Using such relative information from the teacher model to train the student model could be more advantageous as opposed to directly using the hard ground truth labels.

2 Methods

The last layer of a network usually applies a softmax function to derive a normalized list of hard probabilities; this means the larger outputs from the previous layer are exponentially stretched to become even larger and the smaller outputs are exponentially depressed to become even smaller. As a result, the softmax values do not capture the subtle relationships between the softer outputs in the previous layer (the logits). Therefore, we believe that the logits, when combined with a higher temperature in the softmax, can produce softer probability distributions that can tell the relative difference between the different classes in addition to just being able to tell the most probable class, as shown in figures 1 and 2.

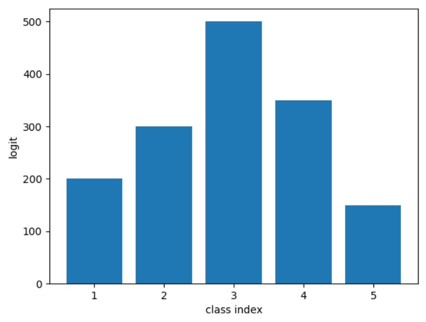

Figure 1: Logits

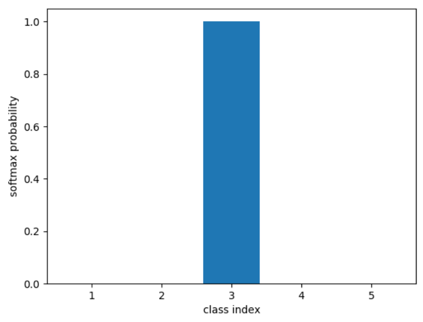

Figure 2: Softmax applied to Figure 1

It is with the above insight, and according to [1], that we make use of logits from the well-trained teacher model to teach the simpler student model. Specifically, we made use of a hybrid loss function that is composed of a weighted sum of two losses - (i) Kullback–Leibler Divergence between the soft targets from the teacher and the student generated with same temperature, and (ii) Cross Entropy between the student logits and the ground truth labels. In addition, there are two parameters in the loss function, the logits transfer temperature T and the weight α given to loss (i), which implies that the weight given to loss (ii) is $1 - \alpha$. The overall loss equation is:

$$\begin{aligned} \text{HybridLoss} = \alpha \cdot T^2 \cdot \text{KLDiv}\left(\text{SoftMax}(\frac{logits_{student}}{T}),\ \text{SoftMax}(\frac{logits_{teacher}}{T}) \right) \\ + (1-\alpha) \cdot \text{CrossEntropy}(y_{student},\ y_{label}) \end{aligned} \tag{1}$$

As recommended in [1], in determining a good value of the temperature T, T it is to be slowly increased from a small value to an optimum value that allows the most transfer of knowledge from the teacher network to the student network. Mathematically, as seen in (1), increasing T would reduce the difference between all the logits because we divide all of them by a larger denominator. This results in generating softer targets and a greater transfer of implicit knowledge, i.e. an understanding of how close or how apart the probabilities of the different classes are in addition to just being able to tell the most probable class, as shown in figures 3 and 4.

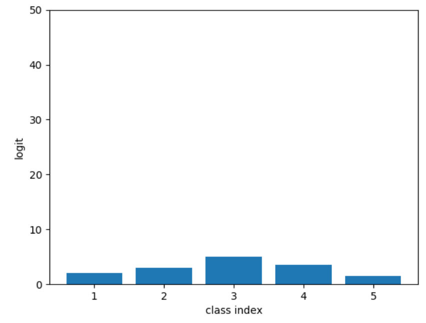

Figure 3: Logits

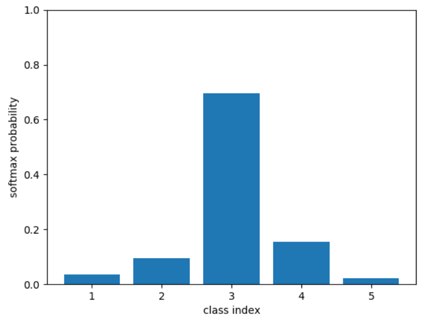

Figure 4: Softmax applied to Figure 3

The training scheme for this knowledge distillation is as follows: The teacher network is first trained to give a high accuracy. Then the student model is trained with the hybrid loss (1). From the Loss function (1) one can notice that while training the student with knowledge distillation we also require the corresponding logits from the teacher network. Calculating the teacher logits every batch while training significantly affects the training time. In order to overcome this problem, we first fix all

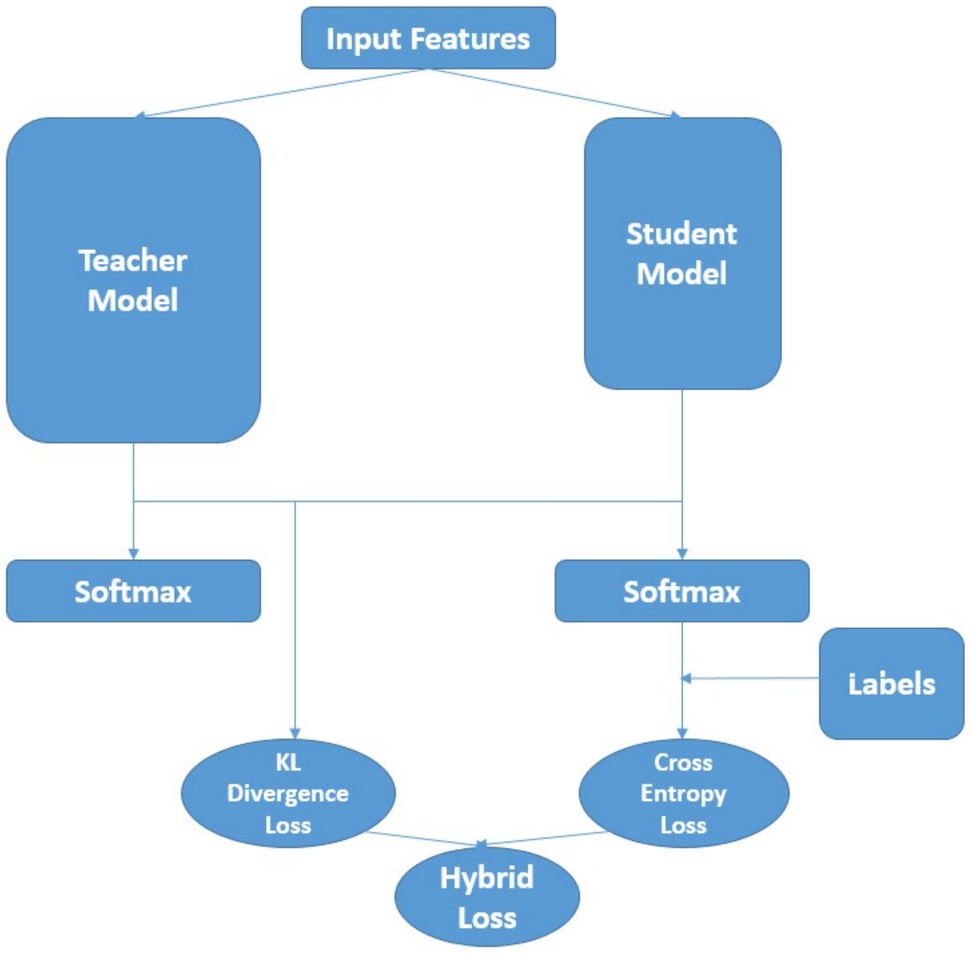

Figure 5: Student Training scheme with Knowledge Distillations

the necessary random seeds and then calculate the logits for entire training data and save in a list. And then we train student by turning off the shuffle, thus ensuring the same sequence of batches and looking up for the corresponding teacher logits from the list. This speeds up the training time by upto 70%.

The graphical illustration of this knowledge distillation training scheme is shown in figure 5.

After we are satisfied with our experiments on knowledge distillation as described above, we extended our research to the more recent blockwise progressive knowledge distillation, which is quite a different approach. Instead of learning from the soft targets produced by the teacher, the progressive blockwise learning aims to train the student to learn from the teacher intermediate output. More specifically, this approach is to minimize the element-wise differences between the blockwise output of the teacher and the student.

The motivation of this approach is that the effectiveness of model distillation depends on the teacher-student network optimization strategy and the student network design. [2]. Also, it is argued that in this scheme, training a limited number of student weights in a blockwise fashion is more efficient than training all the student weights altogether, especially in a a non-convex joint function in [2] where there are numerous local optimas.

The training scheme of the progressive blockwise knowledge distillation is as follows: the teacher network is first considered as a composition of N sub-network teacher blocks. In training stage k, we

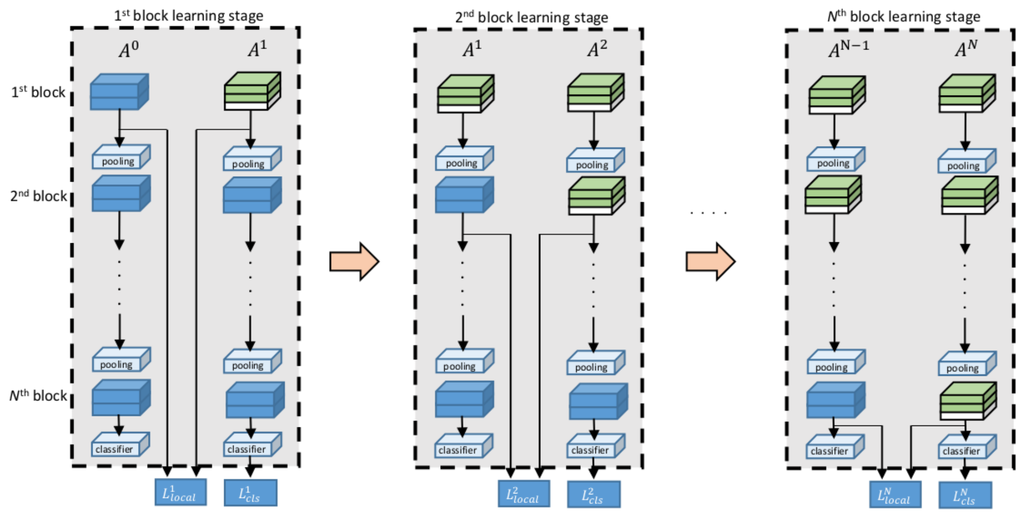

Figure 6: Graphical illustration of the training scheme in progressive blockwise knowledge distillation. This figure is from Wang, Hui, et al. [2]

replace the kth teacher block with a compressed student block and retain the other $N - 1$ teacher blocks. The compressed kth student block must have the same input size and output size as the corresponding kth teacher block that was replaced, however its number internal nodes should be much smaller (50% reduction). We then repeat this N times until the entire network made up of N teacher blocks has been replaced by N smaller student networks, therefore the name *progressive blockwise training*. There are many different possible sequences to replace N blocks one by one; the authors tested top-down, bottom-up and also starting from the middle; and concluded empirically that bottom-up gives the best accuracy.

The total loss function for each stage k is:

$$L^k = \lambda \cdot L^k_{local} + \text{CrossEntropy}(y_{student},\ y_{label}) \tag{2}$$
$$L^k_{local} = \frac{1}{2}||I_{block_k, A^k} - I_{block_k, A^{k-1}}||^2_F$$

where L^k_{local} is for minimizing the element-wise differences between two sub-blocks output and $|| \cdot ||_F$ is the Frobenius norm operator.

The graphical illustration of the progressive blockwise training scheme is shown in fig. 6.

3 Experiments and Results

3.1 Knowledge Distillation by Soft Targets

Data

We used the full CIFAR10 train data of 50,000 images for training and test set of 10,000 images for validation.

Models

For the teacher model, we chose a predefined Resnet-18 model and its pre-trained weights on ImageNet data. We changed the dimension of the last layer to 10 so that it outputs 10 labels to be consistent with CIFAR10. We then fine-tuned the model weights by training it without freezing any layer. The training parameters were SGD with a batch size of 100, learning rate of 0.001 and momentum of 0.9.

For the student model, we built a simple CNN with 3 Convolutional layers (Stride = 1 and Padding = 1) each of which is followed by batchnorm and maxpooling. Following them are two Fully Connected layers. Dropout of 0.5 was applied to the penultimate layer to achieve some regularization. We initially tried SGD with triangular learning rates from 10e-5 to 10e-2 but later we experimented that using Adam with a learning rate of 0.001, beta of 0.9 and no weight decay resulted in faster convergence and higher validation accuracies most of the time.

Training Parameters and Results

We performed a grid search on the student network using 25 different sets of parameters, permuting 5 values of T = [1, 2, 4, 8, 10] with 5 values of α = [0.1, 0.25, 0.5, 0.8, 0.95]. For each parameter setting, we took snapshots of the validation accuracy up to the 50th epoch and also up to the 100th epoch, as shown in figures 7 and 8 respectively. Both figures show a similar trend, that is, better accuracies produced from larger values of T (>4.0) and smaller values of α (<0.5). Also, convergence of the student network for each setting is achieved between 50 and 100 epochs (the average improvement in validation accuracy per epoch from epoch 50 to epoch 100 is very small, at around 0.02%), but we intentionally allowed the models to continue to run so that we could visualize the long term trend.

Notably, when the teacher is removed by setting α=0, the student network achieved only 84.01% after 100 epochs, which is lower than the entire heatmap in figure 8. This is strong evidence that having the teacher's influence on the student's hybrid loss function is more beneficial than using only the ground truth labels.

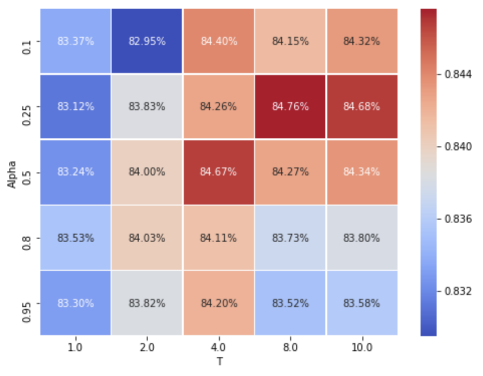

Figure 7: Heatmap 50 epochs

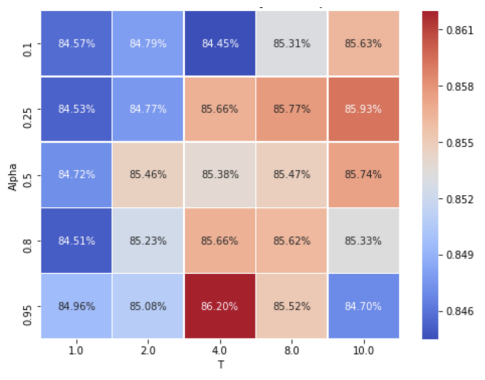

Figure 8: Heatmap 100 epochs

In figure 9, we averaged the validation accuracy for each T across its 5 values of α. The trend is similar to figures 7 and 8. The curves for T = [4, 8, 10] (green, red and purple lines) have the highest validation accuracies.

As an added experiment, we used α = 0.25 and T = [1, 2, 8] to train an ANN student with a feed forward network of 3 layers, so that we can explore if an ANN could also learn the knowledge as well as the CNN trained with the same α and T. However, this was not the case as the ANN has a much lower validation accuracy, as shown in figure 10,

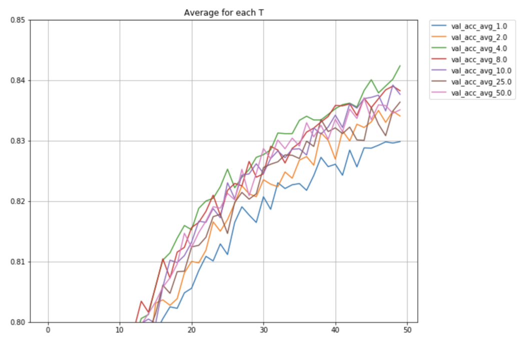

Figure 9: Validation accuracies for various T

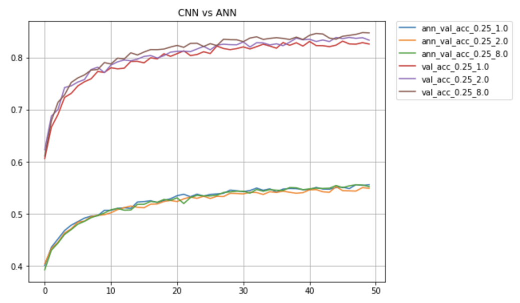

Figure 10: Validation accuracies for CNN and ANN

3.2 Knowledge Distillation by Progressive Blockwise Training

Data

We used the CIFAR10 dataset and normalized the RGB values with means and variances as suggested in [2]. We used 49500 of the data as training set, 500 as the validation set and 10000 as the test set.

Models

For the teacher network, we used a pre-trained VGG-16 with batch normalization, as recommended in [2]. We changed the fully connected classifier by reducing the number of nodes in the fully connected layers from 4096 to 512. We also used a dropout rate of 0.5 on each fully connected layer. The reasons for these modifications are to reduce the computational demands while still achieving a high accuracy in the teacher network. Subsequently, we fine-tuned it without freezing any layers.

Using the pooling layer as the boundaries for the sub-network blocks, we identified the teacher VGG-16 network as being composed of 5 such blocks.

For the student network, we halved the number of channels of all convolution layers from the teacher model. To ensure the output size of the student sub-network block is the same as the corresponding teacher sub-network block, each student's sub-network block has to add N 1x1 filters, where N is the original number of filters (refer to [2]). For convenience, we call this student network **VGG-16*** in the following content.

Training methods

There are three methods to train the student network: (i) knowledge distillation by soft targets, (ii) knowledge distillation by progressive blockwise training, and (iii) training the student from scratch.

Training the student from scratch and training the student with soft targets are quite similar in terms of the training framework, except the change of the loss function from a cross entropy loss to the hybrid loss in equation 1. However, training the student with the progressive blockwise scheme is different. There are three methods purposed in [2], top-down, bottom-up and starting from the middle. We employed the **bottom-up** optimization order because it is the best method among the three methods, according to the empirical results from [2].

Training Parameters

We used SGD with a mini-batch size of 100 to train the student networks. We also applied a momentum of 0.9 and a learning rate weight decay of 0.0005. Instead of the step-wise learning rate used by Wang, Hui, et al. [2], we experimented with the cyclic triangular learning rate and then formally adopted it, as it always results in a faster convergence in every stage.

We set the step size of the cyclic learning rate to 2 times the number of iterations in an epoch and the maximum and the minimum learning rate are 10^{-2} and 10^{-5} respectively.

In the progressive blockwise knowledge distillation, we trained each stage of the network (auxillary network) with 10 epochs and passed the auxiliary network with the highest validation accuracy to the next stage.

As we have five stages in the progressive blockwise knowledge distillation and a total of 50 epochs, to compare with knowledge distillation using soft targets, we also trained the student only with logit cross entropy loss over 50 epochs.

Results

With the help of the cyclic learning rate scheme, all the student models converged within 50 epochs.

For training VGG-16* with knowledge distillation by progressive blockwise training, we performed a grid search on the hyperparameter λ and found that λ=0.005 gave the best result.

For training VGG-16* with knowledge distillation by soft targets, we tested four sets of hyperparameters with the heuristics from the first part of our project. We found that α=0.9 and T=4 gave the best result.

The validation accuracy and the test accuracy are listed in table 1. We see that the student models created using knowledge distillation by soft targets and progressive blockwise training have a significant improvement compared to the student trained from scratch without any knowledge distillation.

Our VGG-16* trained from scratch has a similar performance to the VGG-16* trained only with cross entropy loss of true labels in Wang, Hui, et al [2]. We trained the VGG-16* from scratch without freezing any layer while Wang, Hui, et al. trained the VGG-16* only on the convolution layers and used the fully connected classifier weights from the teacher networks, effectively freezing the fully connected classifier in their experiment.

The accuracy of our VGG-16* trained from scratch is 77.9% whereas the accuracy of the VGG-16* from Wang, Hui, et al. is 72.41%. As we employed the cyclic triangular learning rate scheme and our teacher model has better performance, our model performance is slightly higher than their model.

The accuracy of our VGG-16* trained progressively is 85.6 % and the model from Wang, Hui, et al. is 83.6%. We believe that the same reasons from the above paragraph contributes to our slightly higher performance.

In fig. 12, we can see that there is a sharper drop in model performance from stage 2 to stage 3. We suggest two possible explanations: (i) teacher block 3 is likely to be much more informative than other blocks and therefore it is much more challenging for the student block to capture all information from the teacher block; (ii) it is highly unlikely that we have identical initialization scheme as Wang, Hui, et al, and if the L^k_{local} term is very large, the cost function is too large and the gradient explodes. Potential Improvements to this problem could be to have a flexible number of training epochs at each stage and a better re-scaling multiplier (e.g. the reciprocal of the teacher network output) to balance two terms in eq. (2).

Table 1: The performance of different models in progressive blockwise training

Model	Description	Validation Acc.	Test Acc.
Teacher	VGG-16 fine-tuned from pretrained weights	91.4%	90.4%
Student-ST	VGG-16* with KD by soft targets	89.6%	85.6%
Student-PBT	VGG-16* with progressive blockwise KD	85.2%	82.9%
Student-scratch	VGG-16* from scratch	81.4%	77.9%

4 Discussion and Conclusions

In our first part on knowledge distillation with soft targets, our best student model when trained without knowledge distillation only reached a maximum of 84.01% validation accuracy. Comparing this inferior accuracy to the better results in figure 8, we clearly see that the teacher imparts valuable knowledge to the student. This allows us to conclude that knowledge distillation does improve the learning of a smaller student network from the larger well-trained teacher network.

As mentioned in section 2, the temperature T can tuned to transfer the most knowledge to the student model. From figure 9, our empirical results demonstrated that this is indeed the case. Specifically, at the optimal T=4 (green graph), we almost always have higher accuracies compared to the other T values.

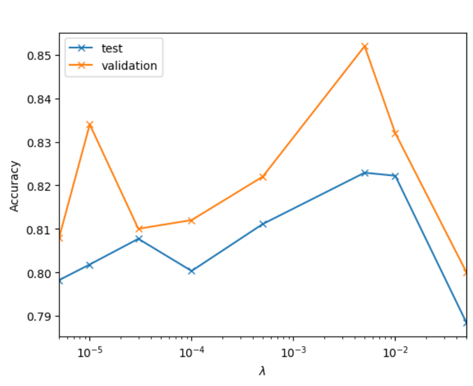

Figure 11: The validation and test accuracy against the hyperparameter λ

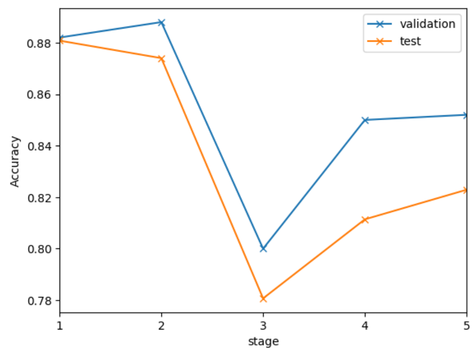

Figure 12: Validation and test accuracies in different stages in progressive blockwise distillation after training. λ=0.005

Even though we already gathered strong evidence in favour of knowledge distillation described in [1], we wanted to go further to explore better accuracies we could potentially get from knowledge distillation by changing other parameters. For example, we removed dropout from our architecture. Interestingly, we obtained a significant improvement in accuracy. When averaged over 10 experiments with T=[1, 2, 4, 8, 10] and α=0.5, we consistently achieved accuracies above 86.5%, with the best accuracy of 88.12% at T=4.0. This gave us even more insight into knowledge transfer. We hypothesized that by using extra knowledge from teacher model in addition to using the truth labels, we are already regularizing the hard truth labels with the soft teacher targets, and therefore adding dropout might lead to excessive regularization.

In our second part on progressive blockwise knowledge distillation, the overall performance of progressive blockwise training performed by us is consistent with the result done by Wang, Hui, et al [2]. Unlike the method in [1], this method restricts the structure of student at very sub-network block when distilling the knowledge. In order to draw a comparison between (i) knowledge distillation using soft targets and (ii) knowledge distillation using progressive blockwise training, we performed another experiment - we trained the same student with only the soft targets form the same teacher and our results from table 1 showed that the student trained with knowledge distillation using soft targets performed better than the student trained by the progressive blockwise scheme.

One possible explanation for our results could be that the CIFAR10 dataset is relatively easy to learn and the extent of knowledge transfer can be effected very well using the soft targets. On the other hand, in a scenario where we have 1000 classes, simply transferring knowledge from soft targets may not be sufficient and in such cases the progressive blockwise training may be more effective. Another possible explanation is that progressive blockwise training inherently uses a contrained stage-by-stage optimization scheme with constraints on model structure and also constrains on weights learned in prior stages, while on the other hand, knowledge distillation using soft targets is a free optimization method with no constraint on the model parameters. In this view, it follows that the former would produce a result at best as good as the latter.

In the absence of extensive experimentation with larger and more varied datasets due to time constraints, we do not conclude the superiority of first method over the second. Hence, we would recommend further experimentation with larger and more varied datasets to understand the effectiveness between knowledge distillation using soft targets and knowledge distillation using progressive blockwise training.

References

[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[2] Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *IJCAI*, pages 2769–2775, 2018.

5 Learning Outcome

5.1 Chun Hung Lin

Fast implementation and testing with open-source packages

In this project, we used pytorch to implement and test the models and our ideas within a reasonable time. It is practically important when we would like to build a prototype at the stage of proof-of-concept.

Able to understand the frontier scientific research in deep learning

With the course content, I can quickly learn the technical context from those frontier research papers. For example, concept of knowledge distillation as well as the typical computer vision network architecture like VGG-16.

Fine-tuning a pretrained network

By making use of a pretrained network with modifications, a well-performed network can be obtained for a specific task. For example, in order to build a teacher network and to reduce the computational complexity, I simplified the VGG-16 classification layers for a simple task like classifying CIFAR-10.

5.2 Run Yan Tan

Approaching research as an open-ended question

Through the implementation of the project's core research publication *Distilling the Knowledge in a Neural Network* and a further extended implementation of a newer publication *Progressive Blockwise Knowledge Distillation for Neural Network Acceleration*, I was reminded that knowledge creation through research really depends on how much we are willing imagine and try. There are no absolute right or wrong methods in research; we simply make use of what we already know, and keep on trying to increase the effectiveness of scientific methods.

Practising the fundamentals and not just use functions blindly

It was a happy moment for me to mathematically and graphically demonstrate and share my insight with the team on how the softmax function exponentially skews the relative difference between the outputs, such that the outputs usually result in only one very dominant class. As a result, the network outputs just before the softmax would retain more information on their relative differences than the actual softmax outputs would. My realization is that it is important to practise the fundamentals and not just use functions (softmax) blindly.

Learning technical skills in PyTorch

Technically, I learned several core features of the PyTorch framework, such as creating cyclical learning rates, encapsulating datasets in an object-oriented manner, applying image variation methods through transforms such as jitter, resizing, cropping and flipping. Overall in this project, I also completed the entire train-validate-test neural network lifecycle in PyTorch.

5.3 Vivek Chalumuri

Familarity with PyTorch

This is the first time I am using it. Using torchvision models and pretrained weights for teacher training and then writing code from scratch for student model familiarized me well with PyTorch. Both in terms of using its existing features for quick applications like transfer learning or writing things form scratch.

Comfort in building and training deep learning models

Tweaking the Resnet18 for CIFAR10 and the raining by freezing the layers and training as a whole without frozen layers. This gave me good insight into what we learned in the class. How freezing some layers makes training faster, less gradients are being calculated etc. Moreover, the dropout experiment with student models in the first method implementation gave me little more insight into dropout. Like how adding it blindly is not a good thing, if it functions as a regularizer then we could also potentially overregularize it by adding it.

Confidence in reading and implementing deep learning research papers

I think this is my main learning outcome. With the extent of current deep learning research, it was quite overwhelming for me to get into the full-depth reading and implementing research papers. In this project, implementing a famous paper gave me a lot of confidence. This I believe would greatly help me during my thesis when I will be researching alone. To be more specific, getting the intuition about what softmax really does and how are soft targets different and what we mean by transferring knowledge....answering these questions one by one made it easy for me to implement the main idea of the paper as a whole.