Model Compression: Going in Depth on Hint-Based Knowledge Distillation

Abstract

Fitnets: Hints for thin Deep Nets by Romero et. al. was one of the first works that introduced Hint-Based Knowledge distillation, a form of model compression, where inner hidden representations of larger teacher networks are used in the training of smaller student networks. In this work, we recreate hint training experiments in the same vein as in Romero et al. utilizing various hint training setups and exploring the importance of depth in student networks. These experiments explore comparing performance when using the teacher model layers or "hints" from different locations in the teacher. It includes experimenting with varying depths of the student network to confirm that deeper student models are useful and improve performance as shown in Romero et. al. This work also includes experimentation with multiple hint layers in the training process to see whether performance is improved or worsened. Some of these ideas were not tested out in Romero et al. and our contributions can provide more insight into the utility of Hint-Based Knowledge Distillation and how it can be used in different ways to improve performance of student networks in knowledge distillation setups.

Code:
1: `https://github.com/vgupta-1/HintTrainingPytorch.git`

1 Introduction

Many pretrained machine learning models and deep neural networks are able to achieve state of the art performance on many challenging tasks of which include image classification, Natural Language Processing, Object Detection, etc. However, these models tend to take up lots of computational power and resources and deploying them in a widespread setting is very complicated and sometimes not possible. In light of this complication, model compression has become an extensive research field trying to reduce the size and cost of large machine learning models while maintaining state of the art performance. Knowledge Distillation has become a very useful method of model compression and can be done in many different ways. Knowledge distillation was introduced by Hinton et al. where the softened output of the Teacher and Student logits are minimized as part of the loss function. In the paper by Romero et al., the use of Feature-Based knowledge was introduced in this process with the idea that the hidden layers in the teacher provide more important information to improve performance.

The knowledge learned by the student model can be categorized into three types. The first is Response Based Knowledge, in which the output of the Teacher layer is used in calculating a distillation loss between the student and teacher outputs so the student outputs more similar results to the teacher. Second is Feature Based Knowledge, where information is learned from the intermediate layers of the teacher model and the distillation loss is then minimized between these intermediate layers between the teacher and student. Finally, there is Relation Based Knowledge where the loss is calculated between feature maps of the two models. In this project report, the experiments primarily focus on

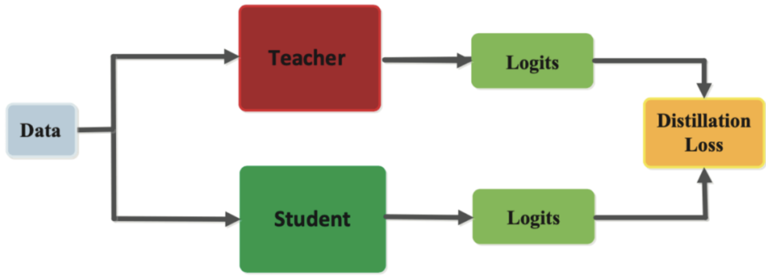


Figure 1: Response-Based Knowledge using the Logits of both of the teacher and student models.
Figure from Gou et. al.

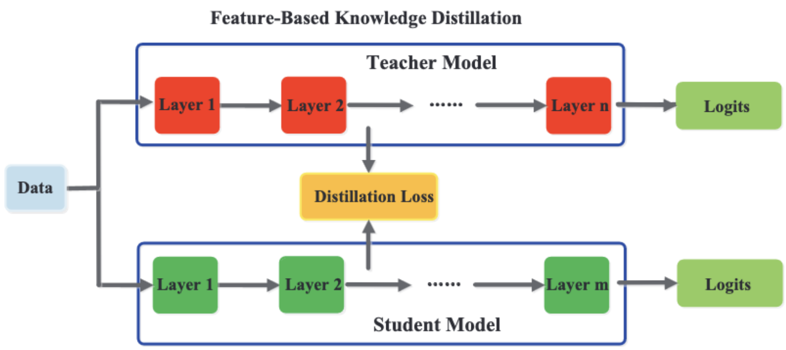


Figure 2: Feature-Based Knowledge using the hidden layers of teacher and student.
Figure from Gou et. al.

training with feature-based knowledge. Response-Based knowledge is used as part of the process as well. Figures 1 and 2 provide visual representations of this type of learning.

Romero et. al. introduced their method of hint training with thin deep student models they called "Fitnets". The paper emphasized the importance of depth when it comes to student networks. Up until the time the paper was published (note this was the pre-ResNet era), most knowledge distillation frameworks would train student networks that had the same number of layers as the teacher networks. The increased depth of the student network was considered to be a major contributing factor in being able to create high performing student networks. They used a 2-stage feature-based knowledge distillation setup where the middle layers of the teacher and student were selected as "hint" and "guided" layers. The student used the teacher's "hint" and minimized the loss between the hint and it's guided layer. Once this loss was minimized the second stage was to use the basic knowledge distillation setup from Hinton et. al. This basic setup can be visualized by Figure 1.

In this work, we use the training method from Romero et. al. to test multiple ideas not explored in the paper and confirm the importance of depth in the student network. First, we see what what happens if the hint and guided layers being in the beginning or end of the networks affect performance since the paper only showed results if they are in the middle. Second, we created our own depth experiments to confirm whether the performance increases with student depth. Finally, we test the performance if the loss is calculated using multiple hint and guided layers instead of just 1. Since the paper implementation [4] uses an old library THEANO, I create my own experiments with a similar setup using PyTorch on the datasets CIFAR-10 and CIFAR-100.

2 Training Methodology

The training process for the experiments we ran was a 2-stage setup where first stage involves the hint training minimizing the loss between the hidden representations of the student and teacher models. The second stage involves the normal knowledge distillation setup from Hinton et al. First, the basic knowledge distillation setup from Hinton et al. will be explained.

2.1 Prior Art

As proposed in Hinton et. al., we use the output of the last layer of the teacher and student models to create the distillation loss. The softmax is used on the logits of both the models, and the soft-target loss between these two values is calculated like in the paper. This would be considered the distillation loss, which is then used within a weighted sum of this value and the loss between the student model predictions and the ground truth labels. This other loss is computed using a Cross-Entropy loss function and the final loss is computed with this weighted sum. The weight alpha can be changed whether we want to increase or decrease the importance of the ground truth labels loss or the Distillation loss in the calculation. Once this final loss is computed, we proceed with back-propagation on the student model and proceed training in this same way. The loss function is described in Figures 3 and 4. This would be the stage 2 vanilla knowledge distillation process after the stage 1 Feature-Based training discussed next.

$$P_t^\tau = softmax(\frac{a_T}{\tau}), \quad P_s^\tau = softmax(\frac{a_S}{\tau})$$

Figure 3: Softmax probabilities (P_t^τ and P_s^τ) of teacher and student scaled by temperature parameter τ are calculated using logits from the teacher and student networks (a_T and a_S). The soft target loss between them is computed like in Hinton et el. and used in the knowledge distillation loss function shown in Figure 4.

$$L_{KD}(T,S) = \lambda H(y_{true}, P_s) + (1-\lambda)STL(P_T^\tau, P_S^\tau)$$

Figure 4: In this function above we use the softmax probabilities above to compute the soft target loss (STL) between the teacher and student logits and we compute the H (cross-entropy) loss of the true labels y_{true} and the student predictions P_s. There are two parts to this loss function, both of these losses are weighted by a parameter λ. In our experiments, $\lambda = 0.75$.

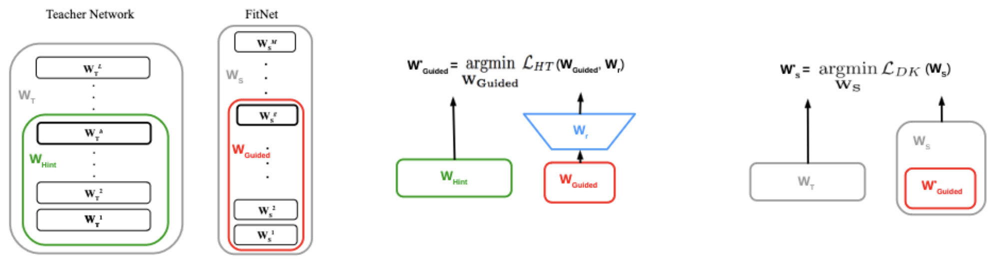
2.2 Hint Training

The focus of the training process in this work is the Feature-Based distillation approach adapted from Romero et al. The idea is choosing a hidden layer from the teacher, the "hint", and one from the student, the "guided" layer, find the loss between those two layers and incorporate that in the training of the student. These two layers will be different sizes so we add a convolutional regressor on top of the guided layer in the student like in Romero et. al so the layers have a matching non-linearity. The hint layer parameters and the regressor feature map parameters are returned to the training portion, and the Mean Squared Error (MSE) loss is computed and the convolutional layers of the network are updated based on this loss. Some differences in our setup vs. Romero et al. is we to use this training process to initialize all the convolutional layer parameters of the student neural networks instead of just up to the guided layer, we use an MSE loss, we change the position of the hint and guided layers, and we experiment with multiple guided and hint layers.

Below are 2 figures and an Algorithm showing this 2-stage featured-based training process we adapted from Romero et al. Figure 5 shows the loss function used to minimize the hint and guided layer loss. Figure 6 from Romero et al. shows the parameters from the hint and guided layers used in the training process and how the hint training loss function and the knowledge distillation loss function use these parameters. Finally, Algorithm 1 shows the two step stage-wise training process we implemented which was adapted from the paper. Romero et. al describes this stage as an important pre-initialization of the student network which makes sense given this stage is updating the network parameters based on the teacher knowledge but not testing against any labels.

$$L_{HT}(Hint_p, Guided_p) = MSE(Hint_p, Regressor(Guided_p))$$

Figure 5: In this loss function above we use the mean squared error (MSE) loss function on the parameters of the hint layer and guided layer ($Hint_p$ and $Guided_p$). A convolutional regressor is put on top of the Guided layer, and the output of this regressor function $Regressor(Guided_p)$ is used as input to the MSE function.



(a) Teacher and Student Networks (b) Hints Training (c) Knowledge Distillation

Figure 6: These graphics show the parameters used and the stage-wise training process. First, we have the hint and guided layer parameters in (a). Then we minimize the loss between these two layers with the loss function from Figure 5 and initialize the parameters up to the last convolutional layer of the student network in (b). Finally, we conduct the normal Knowledge Distillation training from Hinton et. al. in (c). Figure is from Romero et. al. and "Fitnet" is the student.

Algorithm 1 Here, we describe the stage-wise training process for the student networks:
Stage 1: We start with the student and teacher parameters (W_s and W_t) and with the Hint parameters and the Guided parameters (W_h and W_g). Then, we put the guided layer parameters through a convolutional regressor W_r to create a 1-to-1 mapping between the guided and hint layer. Then we minimize this hint loss function for a number of epochs initializing the parameters of the convolutional layers in the student network. This is an initialization process of the student network parameters up until the classification portion similar to Romero et. al.
Stage 2: This involves the basic vanilla knowledge distillation with the new initialized student and the teacher. The knowledge distillation loss is minimized and the final student is returned (W_s^*).

Input: W_s, W_t, W_h, W_g
Output: W_s^*
Student Parameters: $W_s \leftarrow \{W_s^1 ... W_s^n\}$
Teacher Parameters: $W_t \leftarrow \{W_t^1 ... W_t^n\}$
Hint Parameters: $W_h \leftarrow \{W_t^1 ... W_t^h\}$
Guided Parameters: $W_g \leftarrow \{W_s^1 ... W_s^h\}$
Stage 1:
$W_r \leftarrow Regressor(W_g)$
$W_s^* \leftarrow argminL_{HT}(W_h, W_r)$
Stage 2:
$W_s^* \leftarrow argminL_{KD}(W_t, W_s^*)$
$return\ W_s^*$

---

3 Experiments

The Experimental setups and Architectures can be viewed on our GitHub linked in the abstract, here we have brief overviews of the experiments and why we conducted them. The datasets used were image classification datasets CIFAR-10 and CIFAR-100. They are datasets of 32 x 32 coloured images where CIFAR-100 has 100 classes and CIFAR-10 has 10 classes.

3.1 Experiment 1: Hint and Guided Layer Location

In Romero et. al., the authors made the hint and guided layers the middle layers of the teacher and students. They justified this by saying it was to prevent over-regularization of the student network from the teacher network. This was likely a result of their experiments as well. Since they did not show or discuss much on the chosen location of the hint and guided layer, we decided to conduct experiments where the locations were different. In the teacher networks used in this project, we use a shallow wide convolutional neural network of 3 layers. So we conduct three tests where the hint layer is in the beginning, middle and end. The student models are sectioned into 3 portions of convolutional layers so the guided layer will be at the end of the first section if the hint layer is the first teacher layer, end of the second section if the hint is the middle, etc. The goal of this experiment is to see whether there are changes or improvements with performance depending on the hint and guided layers being in the middle, beginning, or end. Using this information we decide the setup of the hint and guided layers for experiment 2.

3.2 Experiment 2: Depth Variation

This experiment was inspired by the depth experiments in Romero et. al. We use the same wide teacher neural network as experiment 1. The student networks are trained using the same two-stage training process outlined in Algorithm 1. We train 4 student networks that have 3, 6, 9, and 12 convolutional layers respectively. The depth of the student network was a major focus of the training process because as mentioned in the Fitnets paper, student networks of larger depth were not considered in knowledge distillation training setups at this time. In their experiments, it seemed that the increase in depth of the student led to better overall performance of the student. This experiment is to confirm this idea. We also decide the hint and guided layer location based off of the results from experiment 1. We want to create the setup that gives the student the best performance and see whether this concept of depth increasing performance holds up.

3.3 Experiment 3: Multiple "Hints"

This experiment was a slight twist on the method inspired from Romero et. al. The paper never considers performance when we use multiple hint and guided layers. So we created a setup where we consider multiple hint and guided layers in the training process and modify the loss function. Since our big teacher network has 3 convolutional layers, we considered three instances of using multiple hint layers. We checked the performance when the hint layers were 1 and 2, 2 and 3, and 1 and 3 from the teacher. The guided layers follow suit like in Experiment 1. The focus of this experiment is to see if multiple hidden representations boost student performance.

4 Results

The next three subsections are tables of the results from our runs of each experiment. We give information about the model size, architecture, training method, and the hint/guided layers. The compression rate gives an idea of how much smaller the student model is compared to the teacher model, putting into perspective why the use of these student models is so important.

In the Experiment 2 section we also provide graphs to show visuals of whether increasing the student depth leads to higher performance. Also note that all of these experiments are separate runs of these models, so while baseline teacher and student model architectures are the same, we will see very minor differences in baseline accuracy too.

4.1 Experiment 1

Table 1: CIFAR-10 Results

Model/Training	Parameters	Accuracy	Compression Rate	# of Convolutional Layers	Hint → Guided (Convolutional Layer #)
Teacher(CE)	3,201,122	77.22%	1	3	NA
Student(CE)	151,988	74.06%	21	9	NA
Student(Basic KD)	151,988	74.64%	21	9	NA
Student(Hint=1)	158,938	74.64%	20	9	1→3
Student(Hint=2)	243,262	72.13%	13	9	2→6
Student(Hint=3)	294,526	74.79%	11	9	3→9

Table 2: CIFAR-100 Results

Model/Training	Parameters	Accuracy	Compression Rate	# of Convolutional Layers	Hint → Guided (Convolutional Layer #)
Teacher(CE)	3,247,292	44.75%	1	3	NA
Student(CE)	151,988	33.79%	21	9	NA
Student(Basic KD)	151,988	35.74%	21	9	NA
Student(Hint=1)	170,548	36.61%	19	9	1→3
Student(Hint=2)	254,872	37.23%	13	9	2→6
Student(Hint=3)	306,136	36.02%	10	9	3→9

4.2 Experiment 2

Table 3: CIFAR-10 Results

Model/Training	Parameters	Accuracy	Compression Rate	# of Convolutional Layers	Hint → Guided (Convolutional Layer #)
Teacher(CE)	3,201,122	77.22%	1	3	NA
Student 1 (Hint=1)	94,234	71.28%	34	3	1→1
Student 2 (Hint=1)	126,586	74.45%	25	6	1→2
Student 3 (Hint=1)	158,938	73.68%	20	9	1→3
Student 4 (Hint=1)	191,290	74.54%	17	12	1→4

Table 4: CIFAR-100 Results

Model/Training	Parameters	Accuracy	Compression Rate	# of Convolutional Layers	Hint → Guided (Convolutional Layer #)
Teacher(CE)	3,247,292	45.78%	1	3	NA
Student 1 (Hint=1)	105,844	37.91%	31	3	1→1
Student 2 (Hint=1)	138,196	37.84%	23	6	1→2
Student 3 (Hint=1)	170,548	36.02%	19	9	1→3
Student 4 (Hint=1)	202,900	35.11%	16	12	1→4

4.3 Experiment 3

Table 5: CIFAR-10 Results

Model/Training	Parameters	Accuracy	Compression Rate	# of Convolutional Layers	Hint → Guided (Convolutional Layer #)
Teacher(CE)	3,201,122	77.22%	1	3	NA
Student(CE)	151,988	72.32%	21	9	NA
Student(Basic KD)	151,988	74.85%	21	9	NA
Student(Hint=1,2)	261,822	75.07%	20	9	1→3 2→6
Student(Hint=2,3)	397,410	72.99%	13	9	2→6 3→9
Student(Hint=1,3)	313,086	74.08%	11	9	1→3 3→9

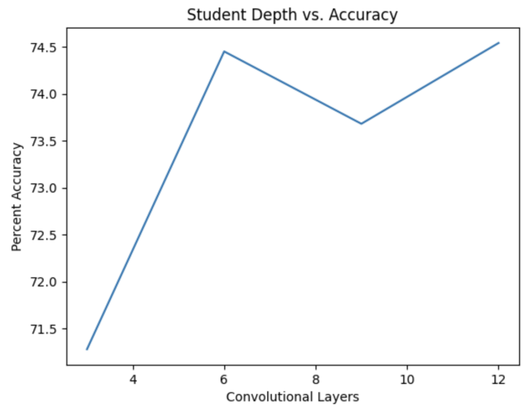


Figure 7: CIFAR10 Depth Experiment Results.

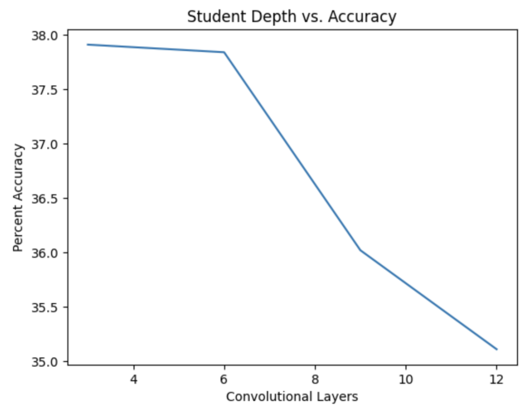


Figure 8: CIFAR100 Depth Experiment Results.

Table 6: CIFAR-100 Results

Model/Training	Parameters	Accuracy	Compression Rate	# of Convolutional Layers	Hint → Guided (Convolutional Layer #)
Teacher(CE)	3,247,292	45.20%	1	3	NA
Student(CE)	151,988	33.84%	21	9	NA
Student(Basic KD)	151,988	36.83%	21	9	NA
Student(Hint=1,2)	170,548	33.48%	19	9	1→3 2→6
Student(Hint=2,3)	409,020	36.09%	8	9	2→6 3→9
Student(Hint=1,3)	324,696	35.52%	10	9	1→3 3→9

4.4 Analysis of Results

4.4.1 Experiment 1

As we can see from Table 1 and 2, we train the baseline Teacher and Student models with Cross Entropy loss (CE). With this training, we get a teacher accuracy of 77.22% and 74.06% for CIFAR-10, and 44.75% and 33.79% for CIFAR-100. The number of parameters and convolutional layers can be seen for each model in the tables as well. Also, with hint training the parameters in the student are more than in the baseline student model because of the added convolutional regressor location in the network. As we can see normal knowledge distillation yields improved accuracies for the student on both datasets of 74.64% and 35.74%. Now moving on to the Hint Training results for each dataset, we see that for CIFAR-10, when the hint layer is the first convolutional layer of the teacher, we get the same accuracy as the knowledge distillation run. With hint being the second convolutional layer of the teacher, we get a lower accuracy of 72.13% and when its the third we get our highest accuracy of 74.79%. Thus, from this experiment it seems that the middle layer is the least accurate and the last convolutional layers being our guided and hint layers yield our highest accuracy. For CIFAR-100 its different and the hint being the 2nd convolutional layer yields our highest student accuracy.

These results show that hint training does yield improved performance on student models as can be seen by the results our highest student accuracies are through hint training. However, this experiment was to test if we could see a pattern in hint and guided layer location affecting performance. However, for CIFAR-10 the middle layers being used yields our worst result and for CIFAR-100 it yields our best result. It is hard to pinpoint why this happened for the CIFAR-10 dataset. It is possible it has to do with the weight we put on hint training versus cross-entropy in the loss function. This was something we could have tested out more, but we kept our λ value to 0.75 throughout our experiments. It is also possible that in this particular experiment for CIFAR-10 learning from the second hidden layer of the teacher is not particularly useful and might provide more bad than useful information. There seems to be no clear winner when it comes to putting the hint and guided layer in the beginning, middle, or end. However, when the hint and guided layer are put in the beginning, it is our second best result for both datasets. Therefore, for experiment two we will set up the experiment such that the hint and guided layers are in the beginning of the teacher and student networks as this seems to be our best bet to get a decent result.

4.4.2 Experiment 2

The depth of the student was a major factor of the Fitnets paper was shown to be very effective in making student networks perform better in knowledge distillation setups like these. So we tried testing this theory out in practice with 4 student networks each with 3, 6, 9, and 12 convolutional layers respectively for both datasets. Based on our results from experiment 1 we decided to put the hint and guided layers in the beginning to maximize overall performance. Tables 3 and 4 show our accuracies for each student of varying depth. Figures 7 and 8 graph these accuracies to show whether performance increases with depth.

Interestingly enough we get conflicting results from the Fitnets paper when comparing the CIFAR-100 results. We can see a clear decline in performance as we increase student depth which is quite surprising. In fact, our student network with three convolutional layers reaches our highest student accuracy 37.91% with a compression rate of 31 for CIFAR-100! This was pretty shocking, as these results conflicted what we saw in the Fitnets paper. However, the graph for CIFAR-10 supports what we see from Romero et. al. as we can see a general positive trend with depth and student accuracy, although the student with 6 convolutional layers performs better than the student with 9. However, it is still a generally positive trend like we were expecting. We can conclude from these results that depth is only useful to maybe a certain point, and that increasing the depth might also increase the leaning of bad information, thus yielding the results we saw from CIFAR-100. However, in practice it may be better to try both methods to see whether there is a trend as we saw a positive and negative trend here. Depth is definitely an influencing factor, for better of for worse based on these results as we see a positive and negative trend.

4.4.3 Experiment 3

In this experiment, we tried to see whether using multiple hints in the training would yield even better results than we saw in experiment 1. However, it appears that it clearly did not help and may have even hurt our results. For both datasets, our vanilla knowledge distillation run on the student outperformed all of the hint training runs here, so there was not any improvement, but actually a slight step back in improvement in this test. This may be because the student is receiving too much varying information from the teacher and not making enough sense of it over the training process. Whereas, if it just receives 1 hidden representation, the hint training process will be really good at using that information and it will help the performance which may be averaged down by training with two hidden representations. It is also possible that there was not enough weight in the loss function on the hidden representation portion of the loss function and playing around with this weight λ we could yield more interesting results. However, in this training setup, using 1 hidden representation instead of 2 seems to be the better option. There may be other setups or loss functions where training with more hidden representations from a teacher model better results, however this setup proved not to be the ideal scenario.

5 Future and Related Work

In this project we explored model compression through hint-training based knowledge distillation. The knowledge type was feature-based and focused on what happened if student models used hidden representations from a large teacher model in their training. Another topic we would like to explore are different Teacher and Student learning architectures. For example, a system where there are multiple teachers or an ensemble of teacher models the student learns from. Also different forms of training would be interesting, for example there are instances where student and teacher train simultaneously. Training with these different strategies and architectures would be interesting areas for future work. There are also many applications of knowledge distillation we can see from Gou et. al. Large Language models have become easier to use with this practice as has been shown with the creation of Distil-BERT. This form of model compression will be useful with any machine learning task like Natural Language Processing, Image Recognition, etc.

6 Conclusion

Overall, in this project we explored the effectiveness of the training method brought forth in the Fitnets paper by Romero et. al. and implemented our own experiments and adaptation in PyTorch. We learned about using feature-based knowledge from the Fitnets paper and the process of how it can be done. We tried different things from the paper such as changing the location of the hint and guided layers as well as using multiple hidden representations in the training process. Not all results were as expected but some seemed to support the results found in Romero et. al. The depth experiments for CIFAR-10 for example. Some were conflicting, for example the middle location for hint and guided layers seemed not to always be the best and depth was not an obvious positive factor for the student. Overall, we learned that hint training can be a powerful tool to improve student networks' performance, depth of the student is an important factor in performance, and more hidden information from the teacher does not always mean better performance.

References

[1] Gou, Jianping, et al. "Knowledge Distillation: A Survey." International Journal of Computer Vision, vol. 129, no. 6, 2021, pp. 1789–1819, https://doi.org/10.1007/s11263-021-01453-z.

[2] Hinton, Geoffrey E. et al. "Distilling the Knowledge in a Neural Network." ArXiv abs/1503.02531 (2015): n. pag.

[3] "Knowledge Distillation Tutorial." Knowledge Distillation Tutorial - PyTorch Tutorials 2.1.0+cu121 Documentation, `pytorch.org/tutorials/beginner/knowledge_distillation_tutorial.html`. Accessed 29 Oct. 2023.

[4] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In: ICLR.

[5] Sachdeva, Kapil. "[Knowledge Distillation] Fitnets : Hints for Thin Deep Nets." Medium, Towards Data Science, 30 June 2020, towardsdatascience.com/knowledge-distillation-fitnets-hints-for-thin-deep-nets-c64840aa2baa.