Reimplementation of Distilling the Knowledge in a Neural Network

Junzhe Wu **Lihao Lu** **Haoxuan Sun** **Xiaoyang Wang**
Zhanghao Chen **Zhihan Chen**
University of Illinois at Urbana–Champaign
{junzhew3,lihaolu2,haoxuan8,xw28,zc32,zhihanc3}@illinois.edu

1 Introduction

Deep neural networks have recently brought about breakthroughs in a wide range of applications from computer vision to natural language processing (LeCun et al., 2015). However, existing deep neural network models are often too computationally expensive and memory intensive to allow deployment to a large number of edge devices (e.g. smartphones and IoT devices). This calls for the need of model compression without significant loss in performance. Hinton et. al. proposed Knowledge Distillation (KD) (2015) as an effective way to compress models, where the knowledge in a cumbersome teacher model or an ensemble of models is distilled and transferred to a more compact student model. The student model is trained to learn the class distributions output by the softmax function of the teacher model, whose temperature is raised to produce a suitably soft set of targets. They achieved promising results on the MNIST dataset and improved an commercial acoustic model by applying KD to an ensemble of models.

In this paper, we present an reimplementaion of Knowledge Distillation (Hinton et al., 2015) with a focus on distilling the knowledge of a single large model into a smaller model for image classification tasks. To explore the full capacity of KD, we perform extensive experiments with various type of deep neural networks on the MNIST, CIFAR-10 and ImageNet datasets. In addition, we also included defensive distillation (Papernot et al., 2016) in this project.

2 Related Work

The idea of exploiting knowledge transfer for model compression was first proposed by Caruana et. al. (2006). They viewed a classification model as a function mapping from input data to sets of predicted class probabilities. An compressed model was trained on unlabeled data or generated pseudo-data labeled by a large target ensemble model to mimic the function learned by the target model. But the work is limited to shallow models. The idea was then adopted by (Ba and Caruana, 2014) to compress deep networks into shallow ones following a student-teacher paradigm. Hinton et. al. (2015) generalized the idea by setting the student model to learn ”soft targets” from the teacher’s softmax function (softened by the temperature hyper-parameter) so as to utilize the richer information encoded in the softer class probability distribution. Recent works have reported to utilize information from intermediate activation records to avoid access to the original training data (Lopes et al., 2017). Aside from knowledge transfer, a number of other model compression techniques have also been developed, including quantization (Gong et al., 2014; Wu et al., 2016), parameter pruning and sharing (Srinivas and Babu, 2015), and matrix factorization (Denil et al., 2013; Sainath et al., 2013).

3 Knowledge Distillation

Knowledge of a teacher model is transferred to a distilled student model by setting the student model to learn the class probabilities output by the softmax function of the teacher model:

$$q_i = \frac{exp(z_i/T)}{\sum exp(z_j/T)} \tag{1}$$

where T is a temperature that is normally set to 1. To produce a softer probability distribution over classes as ”soft targets” for the distilled model to learn, a higher value of T will be used, but in testing phase, the normal temperature of 1 will be used again. The objective function for KD is simply the cross entropy over the transfer dataset between the soft targets and the class probabilities output by the softmax function of the distilled model with the same raised temperature used for producing soft targets:

$$L_{KD} = H(Q_S^{soft}, Q_T^{soft}) \quad (2)$$

where H is the cross entropy, Q_S^{soft} and Q_T^{soft} are class probabilities output by the softmax function of the student and teacher model respectively with the same raised temperature T.

When the correct labels are known for all or some of the transfer set, the method can be improved by also training the distilled model to predict the correct label. In this case, a weighted average of two different objective functions is used as the KD objective function. The first objective function is the same as the simplest KD objective function discussed above. The second objective function is the cross entropy with the correct labels computed with the normal temperature of 1. However, since the magnitude of the gradients produced by the soft targets are scaled by $1/T^2$ due to the raised temperature, the first objective function is multiplied by T^2 to counteract the scaling:

$$\begin{aligned} L_{KD,y} =& \alpha T^2 \cdot H(Q_S^{soft}, Q_T^{soft}) + \\ & (1-\alpha) \cdot H(Q_S, y) \end{aligned} \quad (3)$$

where y is the correct label.

3.1 Defensive Distillation

Defensive distillation is proposed to deal with adversarial examples. Adversarial examples are conceptually same to the clean sample but will be classified incorrectly by the neural network due to adversary noise. One way to generate adversarial examples is to compute the gradient of the model with respect to the data and add the gradients to the clean sample. The adversarial noise computed by the gradient is often come with a constraint which is commonly bounded by the L_2 norm or L_∞.

The defensive distillation tries to give the adversary a misleading gradient so the adversary can not find a desirable local maximum. Unlike the knowledge distillation which aims to compress the model, defensive distillation uses the same architecture for teacher and student model, only change the inputs for student model whose inputs include the soft target produced by teacher model. By doing this, the adversary can't compute the correct gradient of the teacher model when attacking it, the student model will give the adversary a incorrect gradient.

4 Experiments

4.1 On MNIST

In this section, we perform experiments on MNIST to see how well distillation work. A large neural network with two hidden layers of 1200 units was trained as our teacher model. This net using dropout after each hidden layer which is described in another paper(Hinton et al., 2012). We have two different target nets, one target model is a net with two hidden layers of 30 units and another target net has two hidden layers of 300 units. The MNIST data are applied random crop and normalization as data augmentation.

The best teacher model finally achieve 99.37% test accuracy. Adam optimizer is used with 0.001 learning rate. We trained the teacher model for 65 epochs and multiply 0.95 to learning rate after each epoch.

As for the target model with 30 units in each layer, this net can reach 97.50% test accuracy without distillation. Then we trained the model with knowledge distillation in different temperatures and α. The results are shown in table 1. The best test accuracy from table 1 shows that using the knowledge from the teacher model has an improvement in test accuracy of around 0.57%. We can see that soft targets between teacher and student can transfer a great deal of knowledge to the distilled model.

We perform same experiments on the another target net which has 300 units in each hidden layer. This net can achieve 99.22% test accuracy without distillation. Knowledge distillation can not have a significant improvement in test accuracy for this net where the best test accuracy is 99.12% among multiple experiments. Since the student model and teacher model have almost same test accuracy when trained without distillation. However, we find that the convergence speed is obviously improved using knowledge distillation.

Finally, we tried omitting all examples of the digit 3 from the training dataset. The target model gets 89.10% test accuracy without distillation, since the net can not learn anything about digit 3 and we have 1010 three in the test set. After knowledge distillation, the distilled model only makes 92 test errors of which 27 errors are on digit 3. The test errors on digit 3 as shown in the figure.

		temperature		
		1	5	10
α	0.1	96.44%	97.67%	97.75%
	0.5	97.49%	98.07%	97.54%
	0.9	97.47%	97.52%	97.16%

Table 1: MNIST test accuracy of distilled MLP with 2 hidden layers of 30 units

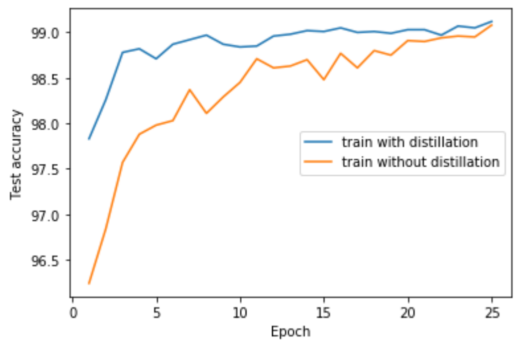

Figure 1: MNIST test accuracy of distilled MLP with 2 hidden layers of 300 units

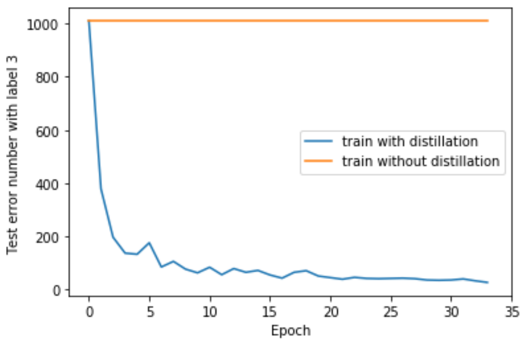

Figure 2: MNIST test errors on digit 3

4.2 On CIFAR-10

Apart from MNIST, experiments have been conducted on CIFAR-10 dataset with ResNet architectures. ResNet-18 is used as the student model and ResNet-101 as the teacher model, with both model trained from scratch. To benchmark the performance, experiments were carried out without knowledge distillation, after which 9 different experiments were conducted with different α and T hyperparameters shown in (3). The ResNet implementation for CIFAR-10 and the PyTorch implementation of equation (3) are available online(Li, 2018) with the rest of the code implemented from scratch.

The CIFAR-10 training set is augmented with random crop and random horizontal flip, and is normalized after converted to tensor. The same normalization is applied to test set. SGD is used as the optimizer with initial learning rate at 0.1.

We first trained a ResNet-18 network and a ResNet-101 network for 101 epochs, with evaluation on test set every 5 epochs. The best test accuracy for ResNet-18 is 92.59% and the best test accuracy for ResNet-101 is 92.12%. As ResNet-101 is around 4 times larger than ResNet-18 in terms of number of parameters, we expect it will take a longer time for ResNet-101 to achieve a higher accuracy and ResNet-101 still has a higher capacity for it to be a teacher model in the knowledge distillation.

With the best ResNet-101 model saved, we trained another ResNet-18 model on the same dataset with knowledge distillation. The ResNet-101 model was loaded and put in evaluation mode when training the ResNet-18 student model. The result predicted from the teacher model was fed into (3) together with the result from the student model. 3 different α and 3 different T were chosen separately and in total 9 experiments were conducted, with their best accuracies listed in the table below:

		temperature		
		1	5	10
α	0.1	92.22%	93.38%	93.03%
	0.5	92.52%	93.21%	92.78%
	0.9	92.41%	92.81%	92.52%

Table 2: CIFAR-10 test accuracy of distilled ResNet-18 model

It can be observed that some experiments shown better results than the original ResNet-18 experiment, which may be attributed to knowledge distillation. Overall, the performance didn't improve a lot, which may be caused by the fact that both ResNet-18 and ResNet-101 already perform well on a small dataset like CIFAR-10 such that the effect of knowledge distillation becomes insignificant even though ResNet-101 has a significantly higher capacity than ResNet-18. Specially, when the temperature is 1, equation (1) decays to socftmax function and the performance was more close to a combination of the two original models.

11 experiments were conducted in total and the total GPU hours consumed was around 290 hours.

4.3 On ImageNet

ImageNet dataset is not publicly available recently, so it took some effort to get this dataset and set it up for our model. After we downloaded the dataset and unzip it, we found that the training data are correctly organized where all pictures for a category sit in one folder. However, all the validation datasets are in one directory. We were able to find a script that creates folders for each category and move pictures into those folders. In this way, we can use the built-in image dataset loader in PyTorch and created loaders for training data and validation data.

We tested our distillation algorithm with a standard ImageNet dataset. The teacher model is a ResNet-50 pretrained on ImageNet, the student model is a ResNet-18 without pretraining. We test our code with temperature equals to 0.5 and 10. The optimizer is Adam with learning rate = 0.01, betas = (0.9, 0.999), epsilon = 1e-8, weight decay = 0. The batch size is 32.

(Dataset processing)The dataset is preprocessed with random crop, ramdon flip and normalization.

Due to the time limited, we report the result we have collected so far. Figures 3, 4, 5, 6 show the accuracy using different temperatures. The red line with the highest accuracy is obtained by the model distilled with temperature = 10, the orange line in the middle stands for the model trained with temperature = 0.5, the magenta line at the bottom is a reference, which is the accuracy of a ResNet-18 trained without a teacher.

As we can see from the figures, the model trained with soft target from teacher model has a better performance. The higher temperature gives us better performance because the ResNet-50 is not a extremely large model compared to ResNet-18. So the soft-target with a higher temperature can give the student model more information. If the capacity of the teacher and student model differs significantly, we would expect the accuracy with a lower temperature would be better.

4.4 Defensive Distillation

We perform experiments on MNIST and CIFAR-10 dataset with FGSM and PGD attacks. The epsilon is set to 0.1 for each attack.

The neural network architecture is the same as the paper(Papernot et al., 2016) described, which is also listed in3 . We use SGD optimizer with lr = 0.01, momentum = 0.9, weight decay=1e-6 and nesterov acceleration. The batch size here is 128 for both MNIST and CIFAR dataset. We applied normalization for CIFAR-10 dataset, no proprocessing for MNIST dataset.

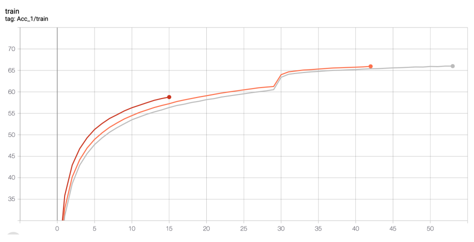

Figure 3: Top 1 Error in Training Set

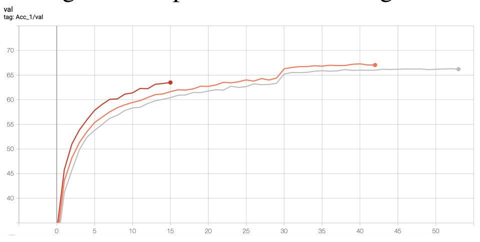

Figure 4: Top 1 Error in Validation Set

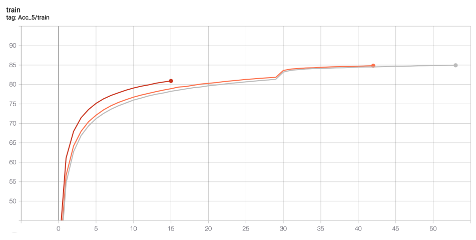

Figure 5: Top 5 Error in Training Set

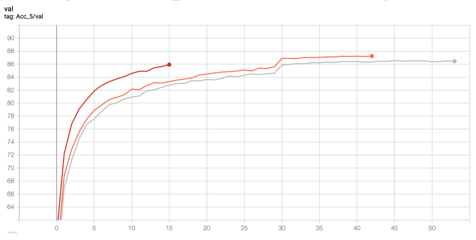

Figure 6: Top 5 Error in Validation Set

Layer Type	MNIST	CIFAR-10
ReLU Conv	32 filters (3x3)	64 filters (3x3)
ReLU Conv	32 filters (3x3)	64 filters (3x3)
Max Pool	2x2	2x2
ReLU Conv	64 filters (3x3)	128 filters (3x3)
ReLU Conv	64 filters (3x3)	128 filters (3x3)
Max Pool	2x2	2x2
ReLU FC	200 units	256 units
ReLU FC	200 units	256 units
Softmax	10 units	10 units

Table 3: Defensive Distillation Network Architecture

We train the teacher model from scratch, then use it to teach another model trained also from scratch but with both hard target and soft target. The soft-max temperature is set to 1 for teacher model and 1/10/100 for student model. The student model with temperature = 1 is a baseline in our experiment.

To test the robustness of our distilled model, we perform attack on distilled model and test whether the adversarial samples can cause a misclassification on teacher model. In the experiment We generate 1000 adversarial for each type of attack. The results showed our distillation can defend 65%-70% adversarial examples. This result is not perfectly aligned with the results in the paper, we suspect the problem is related to the distillation input and temperature.

5 Conclusion

In this project, we re-implemented knowledge distillation paper and primarily demonstrated its effectiveness using different network architectures on different dataset.

Collaboration Statement

Lihao Lu and Zhnaghao Chen implemented the initial code base for the distillition model on ImageNet and MNIST.

Haoxuan Sun and JunZhe Wu implemented and performed knowledge distillation on MNIST.

Zhihan Chen was responsible for the implementation and experiments for CIFAR-10 dataset.

Xiaoyang Wang participated in implementing and testing knowledge distillation code, performed knowledge distillation experiment on ImageNet, as well as re-implemented Defensive Distillation paper.

All authors jointly contributed to the report.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, 2654–2662 (2014).*

Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Miziln. 2006. Model compression. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 535-541 (2006).*

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. 2013. Predicting parameters in deep learning. *Advances in Neural Information Processing Systems 26: Annual Conference on Neural Information Processing Systems 2013, 2148–2156 (2013).*

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115 (2014).*

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

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580.*

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature 521, 436–444 (2015).*

Haitong Li. 2018. knowledge-distillation-pytorch. https://github.com/peterliht/knowledge-distillation-pytorch.

Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. 2017. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535 (2017).*

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE.

Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. *2013 IEEE international conference on acoustics, speech and signal processing, 6655-6659 (2013).*

Suraj Srinivas and R. Venkatesh Babu. 2015. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149 (2015).*

Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized convolutional neural networks for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4820-4828 (2016).*