Exploring Knowledge Distillation of Deep Neural Networks for Efficient Hardware Solutions

Haitong Li
Department of Electrical Engineering, Stanford, CA 94305
Email: haitongl@stanford.edu

Abstract

Knowledge distillation is a methodology of transferring dark knowledge from a teacher DNN to a student DNN along with regular training, which can help deploying a less complex DNN on resource-constrained edge computing hardware for on-the-fly training scenarios. In this work, we study the knowledge distillation on MNIST and CIFAR-10 datasets, with a variety of student-teacher DNN pairs explored. Results show that knowledge distillation provides regularization benefits for both shallow and deep networks, and special use cases such as having unlabeled or partial dataset can benefit from dark knowledge of teacher models as well.

1 Introduction

Deep neural networks (DNNs) have been widely deployed on the cloud for a wide spectrum of applications from computer vision to natural language processing. For real-time inference, deploying DNNs efficiently on edge devices (e.g., mobile devices and IoT nodes) are typically constrained by low-power requirement and limited memory/storage resources. Hence, the simplicity of a DNN model becomes crucial for those scenarios. Knowledge distillation (KD), formulated by Hinton *et al.* [1], is a promising methodology to distill teacher models and partially transfer the dark knowledge to simpler student models. These student models, trained with the combination of original training set and distilled knowledge from teacher models, can provide a viable solution for resource-constrained hardware implementations of DNNs. They may contain richer knowledge than vanilla student models yet possess less complexity (in terms of either total number of parameters, complexity of hidden layers, network depth, or a combination of above) than original teacher models. The role of dark knowledge is not well understood and interpreted so far, therefore, it is worth exploring knowledge distillation with various experiments to further probe into its impact, which may provide useful insights and guidelines for future hardware-software co-implementations. In this project[*], we explore knowledge distillation for image classification on MNIST and CIFAR-10 datasets, using various training set schemes (full-size, data-less, unlabeled). Shallow, deep, and very-deep neural networks are used during the KD experiments (MLP, CNN-5, ResNet-18, WideResNet, ResNext-29, PreResNet-110, DenseNet). Visualization and analysis are further performed to provide better understanding of knowledge distillation and dark knowledge.

2 Related Work

The knowledge distillation for the purpose of model compression is first proposed in [2]. The following related work [3] explores the relationship between depth and width of a neural network in terms of capacity and representation power, which involves training a wide but shallow student network with deep teacher network. Finally, Hinton *et al.* generalizes and re-formulates knowledge distillation in [1], where a student is set to utilize the information contained in the "soft targets" from teacher's softmax probability distribution (softened by a temperature hyperparameter) to aid the training of student models for the same task. Those soft targets may have dark knowledge helpful for student models to also learn from what is hidden in the incorrect classes that a trained teacher produce given the same sample. Recent works have also been reported to utilize such dark knowledge from intermediate features/statistics or other metadata of the teachers [4], [5]. Additionally, model quantization and distillation can be combined to enable model compression [6].

[*]Github repo: https://github.com/peterliht/knowledge-distillation-pytorch

3 Datasets

We use MNIST handwritten digits dataset [7] and CIFAR-10 dataset [8] for the image classification experiments with knowledge distillation. For MNIST with 10-class grey images, 60,000 samples are used for training, and 10,000 samples are used for validation. Normalization is used for preprocessing ([-1, 1] based on mean/std [9]). For CIFAR-10 with 10-class RGB images, 50,000 samples are used for training, and 10,000 samples for validation. Normalization is also used for preprocessing [10]. Data augmentation is used, including random cropping and random horizontal flip. For MNIST, unlabeled training is explored during experiments. For CIFAR-10, training using full-size and 5% training set is explored. MNIST and CIFAR-10 are standard datasets for experimentation, and the goal is more about exploration and less about finding a final single best model on "test" or real-world images. Therefore, we use all the originally available training samples in the training set, and the official test samples all in the validation set for experimentation and exploration.

4 Methods

Training with KD for students first requires training of a teacher model. This can be done as usual using cross entropy loss on the ground truth labels. Then, the first step of knowledge distillation from a regularly-trained teacher model is to convert the pre-softmax logits, z_i, computed for each class into a probability, q_i, by the following equation with the *temperature T (T >= 1)* [1]:

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

With higher temperature, we can get "softer" probability distribution over classes. Since the softmax scores are now softened, the hidden information from the incorrect classes may become more evident to be distilled. The knowledge learned from training a teacher model on the training set with normal softmax (i.e., $T = 1$) can be distilled and partially transfer to a student network, by minimizing the new knowledge distillation (KD) loss (L_{KD}) [1]:

$$L_{KD}(W_{student}) = \alpha T^2 * CrossEntropy(Q_S^\tau, Q_T^\tau) + (1 - \alpha) * CrossEntropy(Q_S, y_{true}). \quad (2)$$

Q_S^τ and Q_T^τ are the softened "targets" of the student and the teacher using the same temperature T (> 1), and α as another hyperparameter tunes the weighted average between two components of the loss. The first component of the KD loss forces the optimization towards a similarly softened softmax distributions for the student, whereas the second component of the KD loss forces the optimization towards approximating the ground truth labels as usual. α = 1 corresponds to using distilled knowledge only with "unlabeled" data for student training.

We have implemented the KD loss and the training pipeline using PyTorch, in the following manner:

(i) We implement the semi-customized KD loss by combining the built-in KL-Divergence loss (for the first component of KD loss) and the CrossEntropy loss (for the second component). This semi-customization approach can better utilize the underlying C-backend for efficiency.

(ii) A teacher network is built and trained with regular softmax scores (log-softmax in PyTorch for numerical stability) and CrossEntropy loss. The model state/parameters are saved.

(iii) Computation graph is created for the student network along with the new KD loss for training. For training mini-batches, Q_T^τ values are fetched from the static teacher model under evaluation mode. We optimize the code to perform pre-fetching before mini-batches, which provides 50X speedup for training the student (due to relatively large size of the teacher models described shortly afterwards). Experiments are done using one Tesla K80 GPU.

5 MNIST Experiments: Unlabeled KD Training

We first explore the knowledge distillation using 3-layer MLPs on MNIST dataset as introduced in [1]. SGD optimize with momentum as 0.9 is used. Training batch size is 128, and the training pipeline follows what is described in the

previous section. Results are summarized in Table 1 (reporting validation accuracy on 10,000 samples). The teacher network MLP-784-1200-1200-10 has 2 hidden layers with ReLU activation units and dropout to reduce variance, and learning rate is set to be 0.01. The trained teacher model is then distilled for training a smaller MLP model of the same depth. It shows that the knowledge from larger teacher model indeed helps the student model generalize better (98.18% validation accuracy). The marginal increase in accuracy is consistent with the results reported in [1] on MNIST. The interesting observation is that, with a larger learning rate, the vanilla student model has decreased performance and also worse performance than the teacher model, yet the student model with distilled knowledge is able to leverage the higher learning rate yielding a better performance (98.50%) than the teacher model. Besides, for the scenario where on-the-fly training on edge devices (e.g., for privacy and security concerns) is needed without access to training labels, we conduct experiments training the student MLP with knowledge distillation but without training labels, as shown in the last row of Table 1. With distilled knowledge from the teacher MLP, the student can still learn a great deal of information. This is probably because the teacher MLP can achieve low bias (> 99% training accuracy), which makes the distilled softmax distributions close to the "true" distributions. One could intentionally modulate teacher's bias and see how bias-variance tradeoff would further impact the learning capability of the student with only distilled knowledge for training, which was not explored due to time constraints.

NN architecture & distillation details	Learning rate: 0.01	Learning rate: 0.1
MLP-784-1200-1200-10 (dropout = 0.8)	98.30% (as the teacher model)	Learning rate too high
MLP-784-800-800-10	98.10%	97.75%
MLP-784-800-800-10 w/ KD	98.18%	98.50%
MLP-784-800-800-10 w/ KD **(unlabeled training data)**	97.69%	**98.16%**

Table 1: MNIST experiments with knowledge distillation

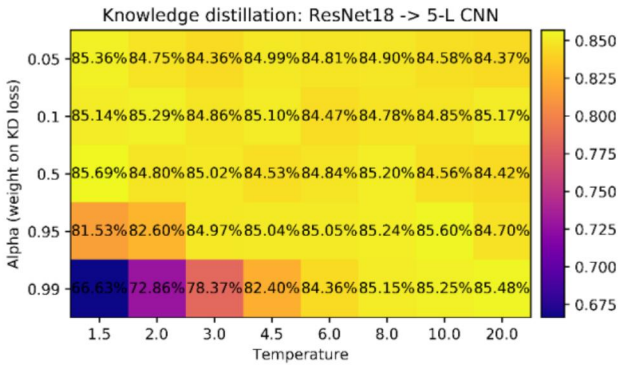

Figure 1: evaluation of distilling *temperature* and *alpha* as KD hyperparameters

6 CIFAR-10 Experiments: Shallow and Deep Distillation

In this section, we describe the KD experiments and analysis on CIFAR-10 dataset. We first define two baseline networks and train the corresponding models. Training batch size is 128 for both. The first one is a 5-layer CNN. We use three convolution (3 by 3 CONV kernels, with stride and padding both equal to 1) layers, each followed by batch normalization, max pooling, and ReLU activation. The first CONV layer has 32 output channels, which get doubled for the following two CONV layers. After that, two consecutive fully-connected (FC) layers are added with Dropout added. It is trained using Adam with 0.001 learning rate. Another baseline model is a 18-layer ResNet [11], which is trained from scratch with 200 epochs on one GPU. Training uses SGD optimizer with 5E-4 weight decay and 0.1 learning rate, which is scheduled to decrease to 0.01 after 150 epochs. The ResNet-18 achieves 94.175% validation accuracy.

	Dropout = 0.5	No dropout
5-layer CNN 3 CONV (w/ BN) + 2 FC	83.51%	84.74%
5-layer CNN **w/ ResNet18-KD**	84.49%	**85.69%**
5-layer CNN 5% training data	/	65.86%
5-layer CNN w/ ResNet18-KD 5% training data	/	66.71%

Table 2: shallow distillation experiments

	Evaluation accuracy (10k samples)
Baseline ResNet-18	94.175%
+ KD WideResNet-28-10	94.333%
+ KD PreResNet-110	94.531%
+ KD DenseNet-100	94.729%
+ KD ResNext-29-8	**94.788%**

Table 3: deep distillation experiments

First, we explore "shallow" KD training of the student 5-layer CNN, using the trained ResNet-18 as the teacher. We evaluate the impact of distilling *temperature* and *alpha*, as the two KD hyperparameters, on the distillation performance (validation accuracy). Since the purpose here is not to find the optimal hyperparameter combination (due to computational and time resource constraints), we use grid search of certain empirical values instead of random search. Results of 40 training experiments are summarized in Figure 1. Intuitively, *alpha* being close to 1.0 means that we put more "trust" on teacher's dark knowledge (even if it may have biases). From the results, it seem to indicate the we should not put too much faith in teacher (e.g., setting *alpha* to be 0.99) unless we use high distilling temperature (e.g, 8.0). Then, we compare adding or removing Dropout for regularization. Table 2 shows that knowledge distillation itself serves as some degree of regularization, to help the student generalize better even without Dropout. Here, it is then interesting to explore such regularization to deal with low-bias, high-variance overfitting situation. We then conduct another set of experiments, using only 5% percent of the original 50,000 training set (2500 samples). Even "shallow" CNN can easily overfit, yielding much worse validation accuracy. Here, it is shown that although the benefit is limited, but indeed the distilled knowledge from ResNet-18 trained on full training set can improve the validation accuracy with 5% training samples.

We then perform "deep" distillation experiments using the baseline ResNet-18 as a student, instead of a teacher. Here, the idea is to explore whether deeper, and more complex state-of-the-art models can be used to help training a student that is already deep enough (in terms of representation power). We choose several state-of-the-art CNNs that excel at image classification, including Residual network variants such as WideResNet [12], PreResNet [13], ResNext [14], as well as DenseNet [15]. We use pre-trained models provided by [17] on CIFAR-10 for these teachers, which all yield better accuracy than the student ResNet-18. For each selected teacher, the student ResNet-18 is trained from scratch for 170 epochs, following the aforementioned optimizer and learning rate schedule. Distilling *temperature* and *alpha* are set to 0.95 and 6.0, respectively. After four different experiments, the results are summarized in Table 3. For the four teachers, the depth increases from 28 (WideResNet-28-10) to 100 (PreResNet-110). The best improvement is obtained from ResNext-29-8 distillation. This shows that even for the state-of-the-art DNNs, knowledge distillation can be leveraged to further provide benefits of improving generalization ability. Further hyperparameter tuning may yield even better results, which was not performed due to limited computational resources.

7 Visualization and Analysis

Figure 2 shows the confusion matrix obtained on the validation set, using KD-trained ResNet-18 with ResNext-29 as the teacher. This only shows the "outside world", and we probe into the "inside world" of knowledge distillation by visualizing the softmax distributions of the ResNext-29 teacher model. As shown in Figure 3(a), on a random subset of 100 training samples, the *temperature* used for distilling ResNext-29 has a significant impact on the "softmax" outputs. When *temperature* rises to 2.0 from 1.0 towards 6.0, the distributions become softer and softer, corresponding to the so-called "dark knowledge" used throughout literature. Such dark knowledge is then partially transferred to the training process of the student ResNet-18. Further error analysis, part of which is shown in Figure 3(b), reveals how knowledge distillation helps the ResNet-18 to improve generalization performance. Given 4 random test samples which baseline ResNet-18 predicts wrong, KD-trained ResNet-18 leverages the softened targets of ResNext-29 and shifts the softmax score distributions to better match the "true" data distributions. Such statistics may be further "recycled" to improve on the knowledge distillation.

8 Conclusion and Future Work

In this work, we experiment with MNIST and CIFAR-10 datasets, unlabeled and data-less schemes, and both shallow and deep neural networks to explore knowledge distillation. It is shown that KD can improve a model's generalization performance, under both shallow and deep distillation cases. Unlabeled and data-less scenarios should leverage the knowledge distilled from large models for resource-constrained hardware. Regularization benefits of KD need to be further explored on ImageNet dataset for large-scale tasks in the future work. Since the teacher models trained on ImageNet may have high biases, it is expected that the distillation need to be tuned to balance the knowledge from true labels with soft targets (not only in the sense of hyperparameters), where a modified form of KD loss may be required. Our open-source code can be accessed via: https://github.com/peterliht/knowledge-distillation-pytorch

9 Team Contribution

Haitong Li performed all the work presented in this final report.

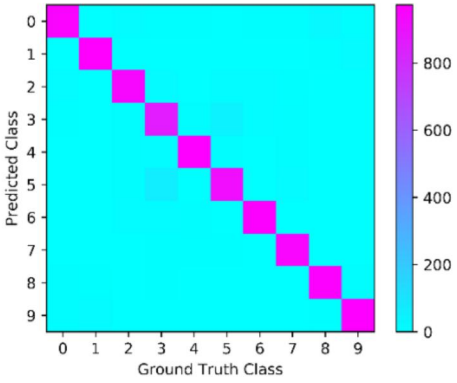

Figure 2: confusion matrix of ResNet-18 with ResNext-29 as the teacher

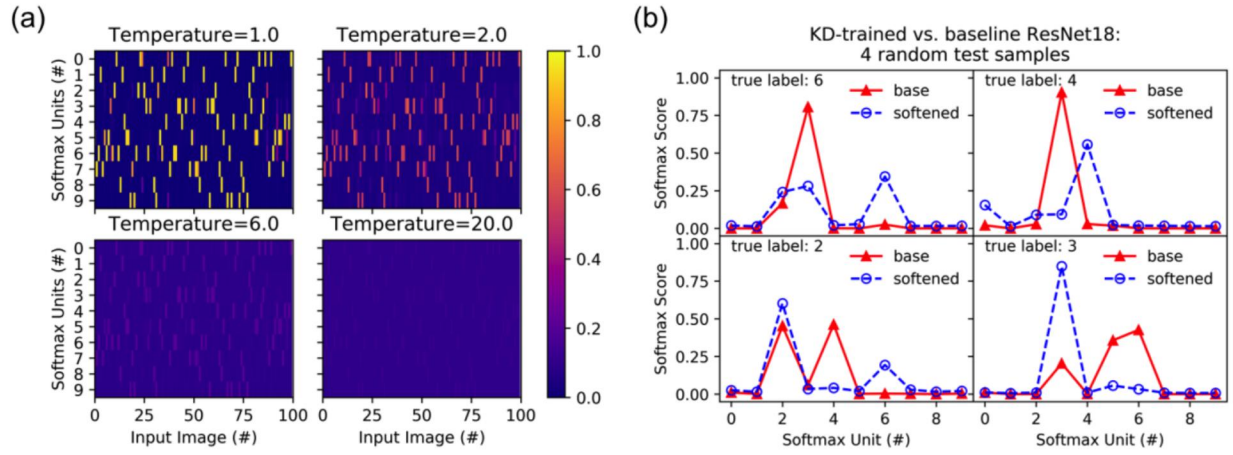

Figure 3: (a) visualization of distilling *temperature*'s impact on softmax distributions. (b) error analysis on 4 random test samples which the baseline ResNet-18 got wrong whereas the KD-trained ResNet-18 got right

References

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

[2] Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006.

[3] Ba, Jimmy, and Rich Caruana. "Do deep nets really need to be deep?." Advances in neural information processing systems. 2014.

[4] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550.

[5] Lopes, R. G., Fenu, S., & Starner, T. (2017). Data-Free Knowledge Distillation for Deep Neural Networks. *arXiv preprint arXiv:1710.07535*.

[6] Polino, A., Pascanu, R., & Alistarh, D. (2018). Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.

[7] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[8] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[9] PyTorch examples, https://github.com/pytorch/examples/tree/master/mnist

[10] PyTorch Tutorial, http://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[11] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[12] Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." arXiv preprint arXiv:1605.07146 (2016).

[13] He, Kaiming, et al. "Identity mappings in deep residual networks." European Conference on Computer Vision. Springer, Cham, 2016.

[14] Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on. IEEE, 2017.

[15] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. Vol. 1. No. 2. 2017.

[16] https://github.com/bearpaw/pytorch-classification