Knowledge Distillation: Current Understanding and Future Directions in an Adversarial Context

[1]**Haoran Zhang,** [1]**Luke Tianyou Zhuo,** [2]**Martin Kuo**
[1]Department of Computer Science, Duke University
[2]Department of Electrical and Computer Engineering, Duke University
{hz271, ltz2, mk585}@duke.edu

Abstract

Convolutional neural networks have been widely employed in the graphic classification tasks. However, these heavily trained networks are too large to run on devices with limited computing resources like wearable devices and mobile phones. Consequently, researchers have sought to boost model accuracy while minimizing model size. Knowledge distillation (KD) achieves this goal by distilling the knowledge from a large model (or an ensemble of models) to a single small model. By reproducing, analyzing, and experimenting with reputed knowledge distillation works, this paper intends to provide additional insights into this field, particularly on student-teacher reverse distillation, self-distillation, and an approach to adversarial distillation training we devised which has shown promise.

1 Introduction

Knowledge distillation typically refers to the process of transferring knowledge from a large model to a smaller model and is widely used in neural network compression. Nevertheless, the necessity for the role of a "teacher" has been debated. Researchers have been developing new ways to reduce or even remove the need for training a larger teacher model. This project reviews prominent research works in this field. In reevaluating and exploring approaches introduced by these works, the project aims to provide new insights into reversed distillation and self-distillation and introduce new approaches to adversarial distillation training.

2 Related Work

The first idea of learning between models is proposed by Romero *et al.*, aimed at reducing the distance between feature maps of student models and teachers models [1]. Hinton *et al.* apply this to neural network compression to distill the knowledge from an ensemble into a single smaller model [2]. Yuan *et al.* revisit distillation and support the notion that knowledge belongs to label smoothing regularization, indicating a poorly-trained teacher model still serves a student model well in knowledge distillation [3]. Yuan *et al.* also propose a Teacher-free framework, where a student model learns from itself or a manually designed "virtual teacher". Zhang *et al.* extend this with Self-Distillation, in which a model learns from the soft outputs of each block of the model during training, where the knowledge in the deeper and shallower portions of the networks is combined [4].

Submitted to Professor Yiran Chen and members of the Chen-Li Lab. Do not distribute.
GitLab Repository: `https://gitlab.oit.duke.edu/ltz2/knowledgedistillation/-/tree/master`
All authors contributed equally to this paper.

Table 1: Accuracy & Architecture comparison between large model and small model on MNIST

Model	**Shape**			
Large Model	(fc1) Linear(in_features=28*28, out_features=1200, bias=True) (fc2): Linear(in_features=1200, out_features=1200, bias=True) (fc3): Linear(in_features=1200, out_features=10, bias=True))			
	FLOPS	**# of Parameters**	**Accuracy**	
	2,392,800	940800+1440000+12000	0.9898	
Small Model	(fc1): Linear(in_features=28*28, out_features=400, bias=True) (fc2): Linear(in_features=400, out_features=10, bias=True)			
	FLOPS	**# of Parameters**	**Accuracy**	
	317,600	313600+4000	0.9806	0.9873

* Small Model accuracy trained w/o distillation (left) and with Distillation (right)

Table 2: Accuracy comparison with Distillation and digit omitted

Digit Omitted	**Accuracy**			
	Without Distillation (T = 1)		**Accuracy with Distillation (T = 10)**	
One	0.9803 (w/o one)	0.00 (only one)	0.9825 (w/o one)	0.9850 (only one)
Two	0.9812 (w/o two)	0.00 (only two)	0.9851 (w/o two)	0.9273 (only two)

3 Methodology

In this paper, we reproduced four works in knowledge distillation, namely Vanilla Distillation, Reversed Distillation, Teacher-free Framework, and Self-Distillation. We compared the prediction accuracy improvement between the student and teacher models (or model accuracy without label smoothing and self-distillation). We mainly use the convolutional neural network (e.g. ResNet) as the experiment model, with MNIST, CIFAR-10, and CIFAR-100 as the test datasets. All neural networks were implemented with PyTorch and accelerated with Google Colab GPU resources. Hyperparameters like epochs, weight regularization, and learning rate schedules were initially taken from relevant works [2, 3, 4, 5], then further experimented on.

4 Experiment

4.1 Vanilla Distillation

The network architecture and the accuracy of the MNIST dataset with and without self-distillation are included in **Table 1**. It is clear that the soft labels generated by the large model do help to improve the accuracy of the smaller model (from 98.06% to 98.73%).

The temperature parameter is used to smooth out the probability predicted from the model. With temperature equal to T, the probability with respect to the input z_i is equal to:

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

Afterwards, we randomly omit one digit from the MNIST datasets and see if the teacher model can transfer knowledge to the student model on digits that it has not seen in the training process.

Higher accuracy in the cases of both the full dataset and omitted digit suggest transference of “dark knowledge” - privileged information without parameter gain - to the student.

Table 3: Re-KD experiment results (accuracy, mean±std over 3 runs in % on CIFAR10)

Teacher: baseline	**Student: baseline**	**Re-KD (S→T)**
95.29±0.09	94.98±0.10	95.46±0.12

Table 4: Tf-KD experiment results (accuracy, mean±std over 3 runs in % on CIFAR10)

ResNet18: baseline	**Tf KD-reg**	**Tf KD-self**
94.98±0.10	95.39±0.15	95.42±0.13

As shown in **Table 2**, the knowledge distillation not only improves the overall prediction accuracy, it transfers the knowledge of digits that the student model has never seen before.

4.2 Reversed Distillation

With $H(q, p)$ represents the cross-entropy loss $\sum_{c=1}^{M} q(k) \log p(k)$, K representing the total number of the classes of labels in the dataset, $p(k)$ represents the hard output prediction, the loss function is defined as:

$$L_{KD} = (1 - \alpha)H(q, p) + \alpha D_{KL}(p^t_\tau, p_\tau) \tag{2}$$

$p^t_\tau(k)$ is the softmax of teacher model S^t output logits. We let the student model S have the hard label q, hard output prediction p, and we designate the output probabilities p_τ. α is a hyperparameter in $[0, 1]$. We seek to minimize the divergence in probabilities between S and S^t.

We use ResNet18 as the student and ResNet50 as the teacher to implement Reversed Distillation [6]. In **Table 3**, we can see that ResNet50 improved by learning from ResNet18 by 0.17%.

In [3], the relationship between Knowledge Distillation (KD) and Label Smoothing Regularization (LSR) is analyzed, whcih can be summarized as follows:

- Knowledge distillation is a learned label smoothing regularization
- With higher temperatures, the distribution of teacher's soft targets in knowledge distillation is more similar to the uniform distribution of label smoothing

Therefore, our experiment results of Re-KD can be explained as the soft targets providing model label smoothing regularization for the teacher.

4.3 Teacher-free Distillation

4.3.1 Teacher-free Self-Training Distillation

As shown in **Table 4**, the self-training approach to knowledge distillation improves accuracy.

Yuan *et al* propose the method of Teacher-free knowledge distillation via self-training, denoted Tf KD-self [3]. We examined the performance of Tf KD-self on CIFAR-10. To implement Tf KD-self, we trained the student model on its own to obtain a pre-trained model the S^p, with output probabilities designated p^t_τ. We then trained the student model S with the hard label q, hard output prediction p, and output probabilities designated p_τ, seeking to minimize the divergence in probabilities between S and S^p. This was done by optimizing for the loss function:

$$L_{KDself} = (1 - \alpha)H(q, p) + \alpha D_{KL}(p^t_\tau, p_\tau) \tag{3}$$

4.3.2 Teacher-free Knowledge Distillation via Manually Designed Regularization

Yuan *et al* also propose the method of teacher-free knowledge distillation via manually designed regularization, denoted Tf KD-reg [3]. We manually designed a virtual teacher, assigning a probability δ to the correct class and probability $(1 - \delta)/(K - 1)$ to all other classes, which was kept in $[0.9, 1]$. We then trained the student model S with hard label q and hard output prediction p, seeking to minimize the divergence in probabilities between the student model S and manually designed virtual

Figure 1: Heatmap of average gradients at each layer of each stage (ResBlock) for Teacher-free Knowledge Distillation and ResNet18 without Knowledge Distillation.

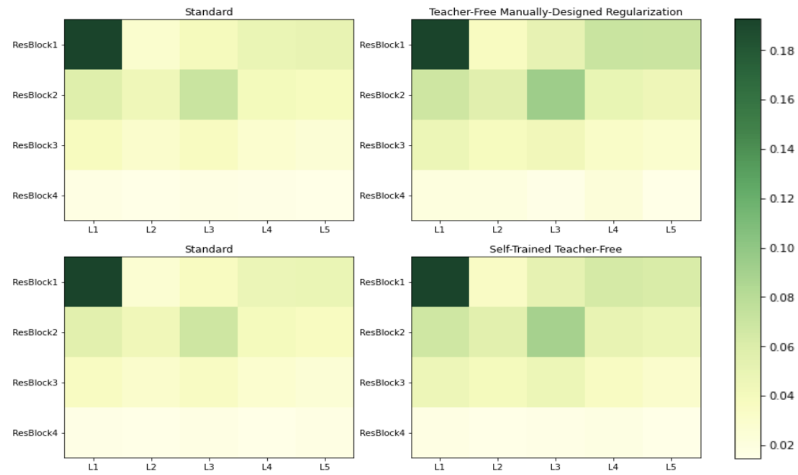

Table 5: Adversarially Trained Tf-KD experiment results on CIFAR10 (accuracy in %).

Method of Knowledge Distillation	**Clean Accuracy**	**AutoAttack Accuracy**
ResNet18 (AT) Student, No Teacher	84.22	46.99
ResNet18 Student, ResNet18 (AT) Teacher	82.70	47.66
ResNet18 (AT) Tf KD-reg	82.02	48.28
ResNet18 (AT) Tf KD-self	81.53	48.83

teacher S^p, with output probabilities designated p_τ and p^d_τ, respectively; this was done by optimizing for the loss function:

$$L_{KDreg} = (1-\alpha)H(q,p) + \alpha D_{KL}(p^d_\tau, p_\tau) \tag{4}$$

To explain the increased accuracy of Tf KD-reg and Tf KD-self, we look to label smoothing and gradients. With either a model itself as a teacher or no teacher entirely, teacher-free methods resemble Re-KD and benefit from the aforementioned label smoothing regularization. Examining gradients for Teacher-free approaches also indicates knowledge distillation mitigates the vanishing gradient problem known to plague deep neural networks [4]. **Figure 1** shows average gradients for layers in the Tf KD-reg and Tf KD-self models are significantly larger than those for a standard ResNet18.

4.4 Adversarial Applications of Knowledge Distillation

Teacher-free models were trained for 50 epochs with 0.1 as the initial learning rate using stochastic gradient descent with exponential decay, as outlined in the work of Maroto *et al* [7].

4.4.1 Adversarial Training with Self-Distillation

We designed an approach inspired by the Tf KD-self method, AT Tf KD-self. We trained the student model on its own against data perturbed by PGD-7 to obtain the pre-trained model S^p. The perturbed data x' generated from clean data x can be represented as:

$$x' = \arg\max_{||x'-x||<\epsilon}(H(f(x'), y)) \tag{5}$$

We designated the output probabilities of S^p for data x to be $p^t_\tau(x)$. We then trained the student model S with hard label q, hard output prediction $p(x')$ and output probabilities $p_\tau(x')$, with x' representing data perturbed by PGD-7, seeking to minimize the divergence in probabilities between

S and S^p. We optimized using the loss function:

$$L_{KDself} = (1-\alpha)H(q, p(x')) + \alpha D_{KL}(p_\tau^t(x), p_\tau(x')) \tag{6}$$

4.4.2 Adversarial Training with Teacher-free (Manually Designed Regularization)

We designed an approach inspired by the Tf KD-reg method (AT Tf KD-reg). We trained the student model S, seeking to minimize the divergence in probabilities between student model S and manually designed virtual teacher S^p. We let q represent the label. $p(x')$ and $p_\tau(x')$ represent the hard output prediction and the output probabilities respectively, of S on x'. $p_\tau^d(x)$ represents the output probabilities of S^p. We optimized for the loss function:

$$L_{KDreg} = (1-\alpha)H(q, p(x')) + \alpha D_{KL}(p_\tau^d, p_\tau(x')) \tag{7}$$

As seen in **Table 5**, AT Tf KD-self and AT Tf KD-reg both drove improvements in AutoAttack accuracy, with degradations in clean accuracy, as to be expected as a result of the tradeoff between accuracy and robustness. Both AT Tf KD-self and AT Tf KD-reg show themselves to be more robust than the baseline approach of AT ResNet18 with no teacher and the ResNet18 Student, ResNet18 AT Teacher approach highlighted by Maroto *et al* [7]. We attribute the robustness of AT Tf KD-reg's to adversarial noise, in large part, to the explicit label-smoothing of AT Tf KD-reg.

Label-smoothing has been known to penalize both over-confidently classified points and badly classified points, enabling greater robustness to attacks seeking to miscategorize predictions [5]. We posit AT Tf KD-self's robustness benefits from the ResNet18 student and its teacher (also a ResNet18) both being trained on adversarial data in conjunction with benefiting from properties of knowledge distillation. We put forth that, like AT Tf KD-reg, AT Tf KD-self exhibits robustness because of label-smoothing. As [3] notes that the self-knowledge distillation is a learned label smoothing regularization, we note that AT Tf KD-self has label smoothing regularization arising from soft targets of the adversarially trained teacher [3].

5 Conclusion

Knowledge distillation enables improved accuracy in a student model. This stems from factors that include not only dark knowledge but also label smoothing regularization and mitigating the vanishing gradient problem. We also saw promising results with the approaches devised to adversarially train Teacher-free knowledge distillation models. Delving into the use of Teacher-free knowledge distillation in an adversarial setting is an area in which we hope to further explore and open up to others to explore.

References

[1] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[2] Geoffrey Hinton, Jeff Dean, and Oriol Vinyals. Distilling the knowledge in a neural network. pages 1–9, 03 2014.

[3] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisit knowledge distillation: a teacher-free framework. 2019.

[4] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. pages 3712–3721, 10 2019.

[5] Morgane Goibert and Elvis Dohmatob. Adversarial robustness via label-smoothing. *arXiv: Learning*, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Javier Maroto, Guillermo Ortiz-Jimenez, and Pascal Frossard. On the benefits of knowledge distillation for adversarial robustness. 03 2022.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No] We were limited by our compute resources. Further exploration would include a large number n of trials, particularly for investigation in adversarial knowledge distillation.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We did not deem it necessary, as we did not extend the scope of our findings to the real-world.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]