When does Knowledge Distillation Help?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Knowledge distillation (KD) has enabled a simple paradigm of designing efficient neural architectures by transferring knowledge from a cumbersome teacher model to a small student model. Recently, several studies have also been devoted to reducing generalization errors when utilizing both the teacher and student models identically. Despite these consistent improvements, it is still challenging to understand how KD facilitates generalization.

In this paper, we aim to understand the effects of KD. We start by introducing two intriguing findings. First, we observe that the original KD can outperform any other recently modified KD methods by optimizing the balancing hyperparameter α and temperature scaling hyperparameter τ. Second, we demonstrate that the small teacher model is able to train any student model with high accuracy using a significantly small amount of training data. We explain these observations using the concept of *data uncertainty* and show that KD is an adaptive label smoothing method that enables the student model to predict only significantly confident data correctly. Additionally, we observe that regularizations such as weight decay and batch normalization do not adversely affect the performance of KD. Under both image classification and machine translation tasks, we validate our belief with extensive experiments.

1 Introduction

Despite the considerable success of the deep neural networks in various tasks such as image classification and natural language processing, there have been increasing demands of building resource-efficient deep neural networks (e.g., fewer parameters) without sacrificing accuracy. Recent progress in this direction has involved in designing efficient neural architecture families [18, 35, 17], sparsely training a network [6, 27], quantizing the weight parameters [3, 45], and distilling knowledge from a well-learned network into another network [15, 46].

The last of these, knowledge distillation (KD), is one of the most potent model compression techniques by transferring knowledge from a cumbersome model to a single small model [15]. Expressed precisely, it utilizes the "soft" probabilities of a large "teacher" network instead of the "hard" targets (i.e., one-hot vectors) to train a smaller "student" network. Some studies have attempted to distill the hidden feature vector of a teacher network in addition to the soft probabilities so that the teacher can transfer rich information [31, 42, 33, 19, 14, 13]. KD can be leveraged to reduce the generalization errors in teacher models (i.e., self-distillation) [44, 30] as well as model compression. However, there is a lack of studies explaining how KD helps generalization.

In this paper, we aim to shed light upon the behavior of neural networks trained with KD via the concept of *data uncertainty*, i.e., the data that a network predicts confidently or not. We systematically optimize the performance of KD into two folds: (1) hyperparameters α and τ of KD, and (2) the amount of training data. Our findings are counter-intuitive ideas to existing advances in KD as follows:

Finding 1. We observe that the original KD without the ground-truth label can outperform various other KD methods that distill the hidden feature vector (Table 1).

Student	Baseline	SKD	FitNets [31]	AT [42]	Jacobian [33]	FT [19]	AB [14]	Overhaul [13]	FKD
WRN-16-2	72.68	73.53	73.70	73.44	73.29	74.09	73.98	75.59	**75.76**
WRN-16-4	77.28	78.31	78.15	77.93	77.82	78.28	78.64	78.20	**78.84**
WRN-28-2	75.12	76.57	76.06	76.20	76.30	76.59	76.81	76.71	**77.28**

Table 1: **Top1 accuracy of various KD methods** on CIFAR100. 'WRN' indicates a family of Wide-ResNet. All student models share the same teacher model as WRN-28-4. SKD (Standard KD) and FKD (Full KD) represent the original KD methods with different hyperparameter values (α, τ) used in Eq. (1) - (0.1, 5) and (1.0, 20), respectively. Others are results reported in [13].

Finding 2. We show that fewer training data leads to better performance of KD when the teacher model is more straightforward (i.e., shallower and thinner network) than the student model (Figure 1). This observation contradicts the long-term belief not only in machine learning that more data certainly gives more accuracy but also in KD that the teacher should be cumbersome.

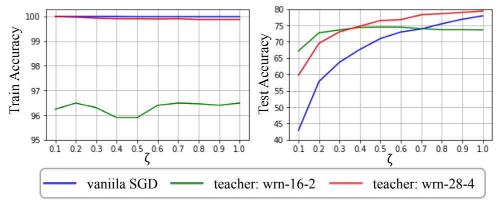


Figure 1: CIFAR-100 accuracy v.s. the ratio of data used to total training data (ζ). For instance, 20,000 data are utilized for training among the entire 50,000 data if ζ is equal to 0.4. All lines retain the same architecture as a WRN-28-4 network, but different training recipes. Both green and red lines utilize the KD method, and each has a teacher model as WRN-16-2 and WRN-28-4, respectively.

To uncover these intriguing phenomena, we contribute the followings:

- We demonstrate that KD is an adaptive label smoothing method based on data uncertainty. Precisely, a student network trained with KD correctly predicts the confident data whose predicted softmax score of the teacher network is significantly high (e.g., nearby 0.99), whereas it does not accurately predict the others. We provide both theoretical and empirical analyses on this idea in section 2.
- We show that KD enhances the performance of a student network as a teacher network becomes shallower and thinner when a significantly small amount of training data is used. Additionally, this effect is negatively correlated with the number of data samples in the training dataset (section 3). We empirically confirm our findings on the effects of KD on image classification and machine translation tasks.
- We observe that the performance of KD without ground-truth labels does not deteriorate when compared with the presence of regularizers such as shortcuts and batch normalization (section 3).

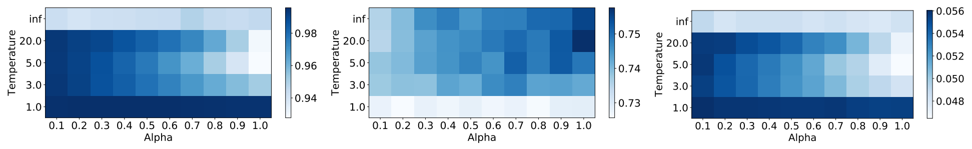
1.1 Preliminary: Knowledge Distillation

We provide a mathematical description of KD before introducing our study. Let us denote the softened probability vector in a network f as $p^f(\tau)$ where τ is a hyperparameter of temperature scaling. The k-th element in $p^f(\tau)$ is defined as $p^f(\tau)_k = \frac{e^{z_k^f/\tau}}{\sum_j e^{z_j^f/\tau}}$ where z^f is a logit vector which is an input to the softmax function and z_k^f is the k-th element in z^f. Then, the typical loss for a student network is a linear combination of the cross entropy (CE) loss $\mathcal{L}_{CE}$ and the KD loss $\mathcal{L}_{KD}$:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{CE}(p^s(1), q) + \alpha\mathcal{L}_{KD}(p^s(\tau), p^t(\tau)),$$
$$\text{where} \quad \mathcal{L}_{KD}(p^s(\tau), p^t(\tau)) = \tau^2 \sum_j p^t(\tau)_j \log \frac{p^t(\tau)_j}{p^s(\tau)_j} \tag{1}$$

where s is a student network, t is a teacher network, q is an one-hot ground truth vector, and α is a hyperparameter of the linear combination. Standard choices are $\alpha = 0.1$ and $\tau \in \{3, 4, 5\}$.

Figure 2: **Regularization effects of α and** τ on CIFAR-100 when (teacher, student) = (WRN-28-4, WRN-16-2). It presents the grid maps of (a) **training top1 accuracies**, (b) **test top1 accuracies**, and (c) **teacher's average entropy** of training samples, which a student predicts correctly. KD with $\tau = \infty$ is implemented with the hand-crafted gradient (Eq. (3)). Detailed values are in Appendix.



(a) Training accuracy. (b) Test accuracy. (c) Entropy of correct samples.

2 Analyzing the regularization effects of KD

In this section, we demonstrate that KD is an adaptive label smoothing method that addresses the over-fitting issue by learning only data that the teacher predicts with the correct answer, confidently. We start by optimizing the hyperparameters α and τ and reveal the advantages of KD based on data uncertainty. We evaluate the performance of KD by varying the combinations of teachers and students in a family of Wide-ResNet (WRN), maintaining the same settings as described in [13, 4].

2.1 Regularization effects from the temperature scaling τ

We observe that α and τ are positively correlated with the test accuracy, whereas they are in contrast to the training accuracy. As Figure 2 depicts, there are consistent tendencies that the higher the α, the less over-fitting problem (i.e., low training accuracy, but high test accuracy) (Figure 2a) under the condition that τ is greater than 1. We find this consistency in various pairs of teachers and students (refer to the Appendix).

An intriguing feature of the hyperparameter τ is that the student model attempts to imitate the logit distribution of the teacher model as τ grows up, while the learning depends more on the classification outcomes of the teacher and the student as τ goes to 0. Here, we extend the gradient analysis of logit in [15] a little further. Consider the gradient of $\mathcal{L}$ in Eq. (1) *w.r.t.* logit z_k^s on each training instance:

$$\frac{\partial \mathcal{L}}{\partial z_k^s} = (1-\alpha)\frac{\partial \mathcal{L}_{CE}}{\partial z_k^s} + \alpha \frac{\partial \mathcal{L}_{KD}}{\partial z_k^s} = (1-\alpha)(p^s(1)_k - q_k) + \alpha\tau(p^s(\tau)_k - p^t(\tau)_k) \qquad (2)$$

The following theorem characterizes the generalization of the student models as τ changes.

Theorem 1 *Let K be the number of classes in the dataset, and $\mathbb{1}[\cdot]$ be the indicator function which is 1 when the statement inside the bracket is true and 0 otherwise. Then,*

$$\lim_{\tau\to\infty} \frac{\partial \mathcal{L}_{KD}}{\partial z_k^s} = \frac{1}{K^2}\sum_{j=1}^{K}\left((z_k^s - z_j^s) - (z_k^t - z_j^t)\right) \qquad (3)$$

$$\lim_{\tau\to 0} \frac{1}{\tau}\frac{\partial \mathcal{L}_{KD}}{\partial z_k^s} = \mathbb{1}[\arg\max_j z_j^s = k] - \mathbb{1}[\arg\max_j z_j^t = k] \qquad (4)$$

Theorem 1 can explain the consistent tendency depicted in Figure 2 as follows: in the course of regularizing the $\mathcal{L}_{KD}$ with sufficiently large τ, a student model attempts to imitate the logit distribution of a teacher model. Specifically, the larger the τ, the more the KD makes the element-wise difference of the student's logit vector similar to that of the teacher. Thus, the logit distribution learning can convey much richer information to the student. We manually train the models with the gradient as Eq. (3) instead of Eq. (2) and confirm its comparable performance to Eq. (2) with $\tau = 20$ (top rows of Figure 2). On the other hand, when τ is close to 0, the gradient of $\mathcal{L}_{KD}$ does not consider the logit distributions, but just check if the student and the teacher share the same output, which transfers limited information. Besides, there is a scaling issue when τ goes to 0. As decreasing τ, KD increasingly loses its qualities and eventually becomes less involved in the learning. One can easily fix the scaling issue by multiplying $1/\tau$ to $\mathcal{L}_{KD}$ for $\tau \leq 1$.

From Theorem 1, we hypothesize that *KD with large α and τ forms the decision boundary through the data which the teacher confidently predicts not through uncertain one.* Consider the early stage of training. Since the student's logit vector is a uniformly random vector, only the data which has significant differences among the elements of the teacher's logit z^t affects the gradient of $\mathcal{L}_{KD}$ while others do not. On the other hand, CE trains all data equally regardless of the degree of uncertainty.

2.2 Mechanism of KD based on data uncertainty

To support our hypothesis, we focus on the uncertainty of each training instance. We study the data uncertainty using the following two measurements:

- **Entropy [32, 8]**: It is defined as $\sum_{j=1}^{K} -p_j^t(x) \log p_j^t(x)$, where x is an input and p_j^t is a corresponding probability of a teacher model for class j. The less entropy, the more confident.
- **Top logit difference (TLD)**: It is defined as $z_{k_*}^s - z_{k_t}^s$, where k_* and k_t indicates the index of the true label and the largest elements in logit z^s of a model s except k_*, respectively. The greater TLD, the more confident.

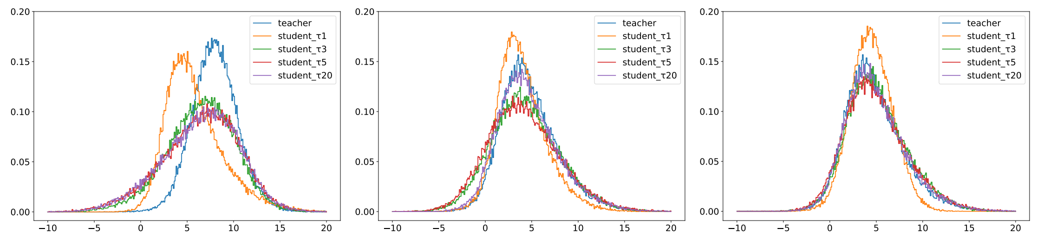
Entropy. We find two consistent trends: (1) increasing both τ and α decreases the average entropy of the training samples that the student model predicts correctly (Figure 2(c)), (2) increasing both hyperparameters also decreases training accuracy (Figure 2(a)). Uncertain data are not well-trained in terms of prediction accuracy when both τ and α are large. However, this training behavior is good for generalization (Figure 2(b)). For the more thorough analysis of these observations, we investigate the average entropy of the entire training data by dividing it into four cases, considering whether the teacher predicts correctly and whether the distilled student predicts correctly. For instance, if a teacher trained with CE predicts correctly for a sample while the student trained with KD does not correctly predict that, this case belongs to the T/F case among four cases in Table 2. We observe that the number of samples in T/T and the average entropy of T/T decrease as τ and α increase, which means that the more influential the distillation is, the more confident the student tries to learn data. It is because, with a high temperature τ, the student tries to learn the teacher's logit distribution rather than the output label, as explained in Theorem 1.

Table 2: Teacher's average entropy and the number of samples corresponding to each case. A teacher and a student are WRN-28-4 and WRN-16-2, respectively.

Case	$\alpha = 1.0$				$\tau = 20$			
	τ=1.0	τ=3.0	τ=5.0	τ=20.0	α=0.1	α=0.4	α=0.7	α=1.0
T/T	0.0541(49696)	0.0468(47540)	0.0451(46361)	0.0459(46348)	0.0543(49621)	0.0532(49290)	0.0512(48618)	0.0459(46348)
T/F	0.2985(253)	0.2252(2409)	0.1881(3588)	0.1778(3601)	0.2213(328)	0.2197(659)	0.2071(1331)	0.1778(3601)
F/T	0.9863(37)	0.8794(20)	0.9725(15)	1.0125(19)	1.0593(37)	1.0041(37)	0.9022(31)	1.0125(19)
F/F	1.1732(14)	1.1397(31)	1.0647(36)	1.0525(32)	0.9804(14)	1.1261(14)	1.2475(20)	1.0525(32)

Top logit difference (TLD). We further interpret the above hypothesis via the perspective of logit difference. Figure 3 depicts the probability density function (pdf) from the histogram of the TLD over the entire training data, where negative values are the TLD of the misclassified data.

Figure 3: Pdf of TLD. All students are trained with $\alpha = 1.0$, and all teachers are trained with CE. Both models share other training recipes such as learning rate, batch size, and weight decay.



(a) WRN-28-4 to WRN-16-2. (b) WRN-16-2 to WRN-16-2. (c) WRN-16-2 to WRN-28-4.

We empirically support Theorem 1 by analyzing the changes of TLD as τ increases. As τ increases, the mean of the student's pdf converges to the mean of the teacher's pdf, which is expected from Theorem 1. When a teacher is the same with (Figure 3(b)) or smaller than a student (Figure 3(c)), a student learns the teacher's logit distribution almost wholly under the sufficiently large τ. However, the student's logit distribution has no longer changed after τ is above a specific value when a teacher is more cumbersome than a student (Figure 3(a)). We believe that because of the bottleneck coming from the student's simplicity, the student can not follow the exact logit distribution of the teacher.

In summary, we think that a teacher supervises the logit distribution of students more strictly as α and τ grow up. Some recent works [36, 7] argued that the benefits of KD come from the class relationship

Table 3: Pearson correlation coefficients (PCC) between training accuracy (Figure 2a) and the average entropy (Figure 2c). The bold indicates p-value<0.05. For each τ, we measure the PCC between two row vectors corresponding to the τ for each grid map, and for each α, between two column vectors corresponding to the α.

Pearson Correlation	τ				α									
	1.0	3.0	5.0	20.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
PCC	**0.6331**	**0.9971**	**0.9975**	**0.9917**	0.6083	**0.9591**	0.9247	0.9408	**0.9719**	**0.9777**	**0.9684**	**0.9735**	**0.9775**	**0.9769**

knowledge given by the teacher. We find that KD indeed transfers data uncertainty, especially the logit differences. We can expect a more substantial effect as both α and τ increase (Table 3). Based on this discovery, we set the hyperparameters α and τ to 1.0 and 20.0, respectively, in the following experiments because it performs similarly compared to that of $\tau = \infty$ while training with Eq. (3) needs more time costs in training.

3 Benefits from the amount of training data

By using the prior knowledge given by the teacher model, KD resolves the over-fitting problem. However, is it necessary to utilize the entire training dataset for KD? In this section, we seek to answer this question. To this aim, we perform an extensive study over a variety of models for image classification and machine translation tasks. We start by investigating the performance of the student when the training data in the process of KD is sub-sampled. For the experiments, we consider the following sampling cases:

Case 1. Equally reduce the number of data for each class in the entire training set.

Case 2. Exclude some classes from the entire training set.

The first case is to check if KD is robust to the amount of training data, and the second case is to answer the following question: can a model trained with KD predict the out-of-distribution (OOD) data (i.e., whose class is not in training dataset) correctly? To this aim, we handle the amount of training data with hyperparameters δ and ζ, where δ is the ratio of the number of classes sampled to the total number of classes and ζ is the ratio of the number of data sampled for actual training to the total number of the training dataset. For instance, when a δ is 0.1 on CIFAR100, the training dataset is reconstructed to have only ten classes, and when a ζ is 0.1 on CIFAR100, the training dataset is reconstructed to have only 50 samples for each class out of 500 samples. If both δ and ζ are equal to 1.0, the entire training dataset is used to train a model.

For the experiments, we consider three types of teacher and student pairs: (1) a teacher has more parameters than a student (OD: over-distillation), (2) a teacher and a student share the same architecture (SD: self-distillation), and (3) a teacher has fewer parameters than a student (UD: under-distillation). We test all three types of teacher and student pairs in image classification tasks and machine translation tasks. For the machine translation, we consider a family of a teacher and a student model as AT-AT and AT-NAT, where AT and NAT stand for the autoregressive transformer and the non-autoregressive transformer, respectively.

We also investigate the benefits from various regularizations such as weight decay, batch normalization, and shortcut when training with FKD.

3.1 Empirical improvements with simple teacher

Efficiency on the amount of training data. A simpler teacher model could facilitate a student's generalization when there is a small amount of training data. As Table 16 reports, in the case of $(\delta, \zeta) = (1.0, 0.1)$, all models of which the teacher model is WRN-16-2 show better generalization results than those with other teacher models. Figure 4 supports that students can mimic the logit distribution of the entire training data of the teacher only with a fraction of the data. We observe the same trend in a family of ResNets (see Appendix).

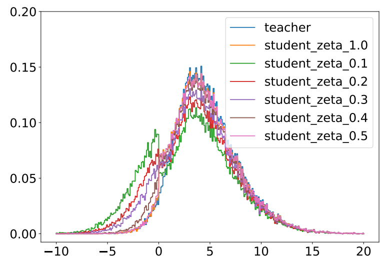


Figure 4: Pdf of TLD for (student, teacher) = (WRN-16-6, WRN-16-2) as ζ varies.

We also test the English-to-German machine translation task using the Transformer [37] architecture. Recently, for training efficiency, there have been increasing demands of building NAT models with

Table 4: Comparison of top1 accuracies on CIFAR-100 with different pairs of teacher and student models. 'Vanilla' indicates the standard SGD training with CE. All teacher models are trained with the dataset whose (δ, ζ) is equal to $(1.0, 1.0)$. We report the best result over 3 individual runs with different initializations.

	teacher	student	$\zeta, \delta = 1.0$	$\delta = 1.0$					$\zeta = 1.0$				
				ζ=0.1	ζ=0.2	ζ=0.3	ζ=0.4	ζ=0.5	δ=0.1	δ=0.2	δ=0.3	δ=0.4	δ=0.5
Vanilla	None	WRN-16-2	72.56	42.51	54.76	60.15	63.74	66.65	9.13	17.06	24.99	32.33	39.55
		WRN-16-4	76.89	43.29	57.56	63.94	66.83	69.81	9.05	17.39	25.50	32.69	40.85
		WRN-16-6	77.62	45.07	58.63	64.54	68.81	69.14	17.32	25.62	33.37	41.24	48.92
		WRN-28-2	74.83	40.58	55.76	61.37	65.45	67.74	8.92	17.35	25.28	32.72	40.45
		WRN-40-2	75.80	40.76	55.23	61.73	66.05	68.62	9.08	17.34	25.24	32.93	40.81
		WRN-28-4	77.93	42.88	58.39	63.73	67.66	70.98	9.00	17.40	25.61	33.17	41.18
OD	WRN-28-4	WRN-16-2	75.76	54.09	62.70	66.84	69.38	71.11	10.16	19.32	26.95	34.35	42.4
		WRN-16-4	78.84	60.15	68.52	71.68	73.63	75.54	10.96	24.02	32.17	40.76	49.02
		WRN-28-2	77.36	54.93	65.14	68.84	71.14	72.63	9.69	19.44	27.61	35.91	44.13
		WRN-40-2	77.80	55.83	65.52	70.13	72.18	74.03	9.80	19.02	27.80	36.03	44.11
SD		WRN-16-2	73.05	63.39	68.79	70.62	71.72	72.35	**32.07**	50.11	57.75	62.51	66.25
		WRN-16-4	77.15	64.72	72.81	74.31	75.38	76.39	15.30	30.51	39.61	46.33	52.07
		WRN-16-6	79.18	66.66	73.63	75.79	76.94	78.23	19.39	45.03	54.72	61.76	65.90
		WRN-28-2	76.13	59.69	68.38	71.48	73.20	74.41	14.37	34.65	45.66	52.73	58.96
		WRN-40-2	76.97	58.61	68.34	71.74	73.42	74.46	11.20	25.44	36.99	47.05	54.01
		WRN-28-4	79.41	59.76	69.54	73.05	74.79	76.49	10.43	22.48	31.25	41.54	50.13
UD	WRN-16-2	WRN-16-4	73.61	**67.22**	72.30	73.23	**74.00**	73.79	**43.74**	61.67	67.14	71.9	73.05
		WRN-16-6	73.38	**68.52**	73.03	74.49	**74.64**	74.11	**48.91**	65.21	69.39	73.05	73.57
		WRN-28-2	73.97	**63.38**	70.49	**72.01**	72.91	73.55	**32.36**	53.48	61.05	66.07	68.25
		WRN-40-2	74.65	**64.27**	70.6	**72.39**	73.08	73.75	**32.23**	55.63	62.27	67.00	69.52
		WRN-28-4	73.61	**67.45**	72.74	73.62	74.40	**74.48**	**44.32**	62.83	67.58	70.90	72.94

Table 5: Comparison of BLEU scores on WMT14 En-De with different pairs of teacher and student models. In this task, since the data has no clear categories like image data, we only conduct an experiment with hyperparameter ζ. We use the same settings in [46].

teacher	student	ζ									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
AT-base	NAT-base	9.74	11.82	12.78	13.89	15.34	15.69	16.38	16.58	16.85	18.53
	AT-base	19.62	21.53	22.66	22.93	25.11	25.99	26.67	26.74	27.00	27.10
	AT-big	19.97	22.43	23.18	23.76	25.86	26.86	27.35	27.26	27.83	27.91

KD [46]. Unlike previous approaches, we find that distilling the knowledge from an AT model to another AT model is much more efficient than to NAT model. Table 5 shows that the AT-base student model with $\zeta = 0.1$ has a 1.12 higher BLEU score than NAT-base with $\zeta = 1.0$ while they have the same number of weight parameters.

Effectiveness in predicting the OOD data. We observe that a student model accurately predicts a class of data that has never been seen before (i.e., OOD data) when a student learns from a simple teacher. Specifically, the knowledge of cumbersome teachers is too marginal to instruct a student to know a class of OOD data that a teacher trains. In Table 16, when (δ, ζ) is equal to $(0.1, 1.0)$, most of them have accuracy as nearby 10% since they are only trained with 10 classes of data. However, on the contrary to this result, WRN-16-2 improves the efficacy regardless of the kinds of the student model.

Regularization effect to release the over-fitting issue. Interestingly, UD goes to a better optima when training a student with fewer data. As shown in Table 16, there exists certain $\zeta \leq 0.5$ under $\delta = 1.0$ which satisfies with one of the two conditions depending on the combination of pair: (1) a deep student (e.g., WRN-28-2 and WRN-40-2) trained with UD of a shallow teacher (e.g., WRN-16-2) outperforms the student trained with any other distillation methods including OD and SD, or (2) a wide student (e.g., WRN-16-4 and WRN-16-6) which learns the knowledge from a thin teacher (e.g., WRN-16-2) outperforms the student trained with the same training recipe when $\zeta, \delta = 1.0$.

The last of these results is somewhat surprising in light of the long-term belief in the machine learning that more data generalize the model better. This observation is also found in a family of ResNet (see

Table 6: Ablation performance on various regularizations on CIFAR-100 dataset. 'RN' indicates the a family of ResNet. All teacher models are trained with such regularizations on the dataset whose (δ, ζ) is equal to $(1.0, 1.0)$.

	teacher	student	$\zeta, \delta = 1.0$	$\delta = 1.0$					$\zeta = 1.0$				
				ζ=0.1	ζ=0.2	ζ=0.3	ζ=0.4	ζ=0.5	δ=0.1	δ=0.2	δ=0.3	δ=0.4	δ=0.5
W/O WD	None	RN-20-4	72.67	36.05	50.14	58.11	63.28	66.17	8.74	16.98	24.46	32.12	39.6
	RN-20-1	RN-20-1	68.71	57.31	63.62	65.41	66.55	67.21	31.13	47.50	53.14	58.54	61.17
		RN-20-4	69.49	66.99	70.02	71.39	70.57	70.58	53.25	65.38	68.73	69.73	69.98
W/O BN	None	RN-20-4	76.15	41.21	56.73	63.52	67.07	70.07	8.97	17.49	25.63	33.2	41.17
	RN-20-1	RN-20-1	69.04	57.55	63.72	65.35	67.07	66.96	30.42	47.86	54.12	57.68	61.70
		RN-20-4	69.72	66.27	69.96	71.31	70.55	70.5	52.99	66.01	68.82	70.02	69.82
W/O SC	None	RN-20-4	74.61	37.18	53.23	61.67	64.96	67.27	8.97	17.1	25.08	32.61	40.65
	RN-20-1	RN-20-1	67.45	51.06	60.57	63.77	65.38	66.45	19.67	40.7	48.18	53.93	56.78
		RN-20-4	70.41	63.72	69.1	70.56	70.82	71.02	40.88	62.94	66.68	68.6	69.93

Appendix). The simple teacher is over-fitted with the distribution of training data, but by using fewer data, it solves the problem of over-fitting. Therefore, *a teacher with lower accuracy is sometimes the one that distills better.*

3.2 The effects of SD iterations

We demonstrate that SD iterations act as a regularizer and thus consistently improves generalization until certain steps. However, if the number of iterations exceeds certain steps, SD iterations lose the information on soft probabilities. Precisely, SD iterations make not only the smoothing part more uniform but also the confidence part smaller. Therefore, after certain steps, the performance starts to decrease. Table 7 reports the result. In the first step of SD, each student model consistently has a better optima than vanilla. After a specific step, each of them has a similar performance to vanilla.

Table 7: Top1 accuracy of SD iterations on the entire CIFAR-100 whose (δ, ζ) is equal to $(1.0, 1.0)$.

model	Vanilla	The step of SD							
		1	2	3	4	5	6	7	8
WRN-16-2	72.56	73.02	72.79	72.86	72.86	72.9	72.52	72.33	72.53
RN-20-1	68.08	68.51	68.67	68.55	68.42	68.53	68.31	68.42	68.12

3.3 Ablation study on other regularizations

We conduct an extensive ablation study by systematically eliminating three regularizations to analyze the robustness of KD towards such regularizations: (1) weight decay (WD), (2) batch normalization (BN), and (3) shortcut (SC). In the setting for BN, we fix the values of both β and γ (i.e., affine parameters) in BN layers of a student model instead of elimination since the network can not be trained without BN layer. For the investigation of SC, we eliminate the shortcut of a student model, including identity mapping. The results in Table 6 show that, interestingly, even without such regularizations, a student model trained with KD has a similar characteristic, which fewer data leads to better generalization. Additionally, other consistent tendencies described in subsection 3.1 also happen.

4 Other factors of KD

Class similarity. We study how the prior in the form of class similarities [29] affects the benefit from the number of training data compared to FKD. Specifically, we experiment with the use of a hand-crafted label (HCL) that utilizes the prior knowledge of class similarities from the teacher [29] (Table 8). Artificially creating a soft target with hand-craft by importing class-wise information from the teacher is not valid when there are few data, while the performance slightly increases when $\delta, \zeta = 1.0$.

Table 8: Top1 accuracy of HCL on CIFAR-100 dataset. We keep the student the same as WRN-16-2 and use the same settings described in [29].

teacher	$\zeta, \delta = 1.0$	$\delta = 1.0$					$\zeta = 1.0$				
		ζ=0.1	ζ=0.2	ζ=0.3	ζ=0.4	ζ=0.5	δ=0.1	δ=0.2	δ=0.3	δ=0.4	δ=0.5
WRN-16-2	72.78	40.52	52.86	58.86	62.59	68.08	9.13	17.34	25.09	31.92	39.73
WRN-16-6	73.04	42.05	53.27	60.05	63.92	66.92	9.16	17.48	24.99	32.51	40.20

Regularizaion of large learning rate. Recently, [24] demonstrates that a large learning rate model with annealing generalizes better on hard-to-generalize and easier-to-fit patterns than its small learning rate. Here, we aim to study whether KD facilitates the regularization effect of this concept. For the experiment, we train models for 200 epochs with an initial learning rate of 0.1, and the learning rate is annealed when the epoch is 100 and 150.

Table 9 shows that the larger the τ, the lower the teacher's average entropy of training samples while they have the same training accuracy in the phase of the initial learning rate. In other words, KD with larger τ attempts to learn more from the data that is hard-to-generalize and easy-to-fit patterns before the learning rate is annealed. Furthermore, this effect still works in annealed learning rates.

Table 9: Top1 training accuracy and the teacher's average entropy of training samples when the learning rate is annealed. 'CE' and 'KD' indicate the models trained with CE and KD, respectively. We use the pair of a teacher and a student as (WRN-28-4, WRN-16-2). All 'KD' have $\alpha = 1.0$.

learning rate	CE	KD (τ=1)	KD (τ=3)	KD (τ=5)	KD (τ=20)
0.1	66.02 (0.0432)	63.44 (0.0409)	65.66 (0.0381)	65.67 (0.0373)	66.61 (0.0388)
0.01	92.69 (0.0524)	91.66 (0.0515)	87.52 (0.0442)	85.50 (0.0429)	86.06 (0.0432)
0.001	99.47 (0.0558)	99.03 (0.0545)	94.69 (0.0478)	92.19 (0.0467)	92.36 (0.0469)

5 Related Work

There have been debates on explaining the dark knowledge of KD. In general, the mysteries of dark knowledge have been mainly explained into two folds: (1) class similarity and (2) sample re-weighting. Hinton *et al.* [15] first suggested that the wrong answers of a teacher rather strengthen KD via the concept of similarity information between classes. They showed that the distilled model could distinguish the data whose label does not exist in the training set. Recently, [36] claimed that KD benefits from class similarity information by comparing each template (i.e., a row vector of the fully connected layer corresponding to the class). As evidence, in the distilled student, they observed high cosine similarity values between templates that share common super-class.

On the other side, Furlanello *et al.* [7] asserted that a maximum value of teacher's softmax probability is similar to importance weighting by showing that permuting all of the non-argmax elements can also improve performance. Yuan *et al.* [41] argued that the dark knowledge serves as a label smoothing regularizer rather than as a transfer of class similarity by showing a poorly-trained or smaller teacher can boost performance. Recently, Tang *et al.* [36] modified the conjecture in [7] and showed that the sample is positively re-weighted by the prediction of the teacher's logit vector.

Some studies demonstrated that the dark knowledge also releases the challenge of training a network with a fraction of the entire dataset. Kimura *et al.* [21] distilled the pseudo training data generated by the teacher in an adversarial manner. Xu *et al.* [38] gained more generalization through adding unlabeled samples into original dataset. They judged the validity of unlabeled samples by checking whether it is in the original data distribution or not. Lopes *et al.* [25] showed that metadata including the information of a pretrained model deployment even enables the training of a student. Recent progress [23, 29, 40] has been evolving into the generation of pseudo examples to get a high accuracy under no access to the original dataset. Hinton *et al.* [15] observed the robustness of KD against classifying the classes of data, which is omitted from the training dataset. However, no studies still have investigated the effects of the dark knowledge with respect to the number of data samples.

6 Conclusion and future research

We provide a novel approach to understand the mechanism of knowledge distillation (KD) into two folds. Firstly, we optimize the hyperparameters α and τ theoretically and analyze the behavior of KD based on the concept of data uncertainty. We empirically support this idea with a visualization from experiments. Secondly, we uncover that using only a fraction of training data is sufficient to distill the knowledge, and the simpler the teacher is, the more effective it is. Additionally, we present an exhaustive study of other factors influencing KD: applying various regularizations such as weight decay, batch normalization, hand-crafted labels, and an initial learning rate.

In the future, we believe that our findings suggest new directions, focusing on optimizing both the amount of training data and data uncertainty. The design of better algorithms considering the pair of a teacher and a student is also an engaging question for future work.

Broader Impact

We expect that this work can be a silver bullet in solving many challenges of deep learning. Firstly, our findings can promote the studies of deep learning in a lack of training data. Specifically, our findings will be of great help in tasks that do not allow easy access to personal information due to privacy issues, making it difficult to secure training data (e.g., federated learning). Besides, it will make it possible to have a lightweight model with better performance than before. Therefore, many applications of deep learning can release the issues of the limited resource budget, e.g., storage and computational costs. Lastly, it will be of use to understand the course of training in deep learning, so further improvements in the applications of deep learning can be expected based on our findings.

References

[1] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9163–9171, 2019.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. *CoRR*, abs/1805.11046, 2018.

[4] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4794–4802, 2019.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[6] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[7] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

[8] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org, 2017.

[9] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[11] 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.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[13] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1921–1930, 2019.

[14] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3779–3787, 2019.

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

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019.

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[19] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *Advances in Neural Information Processing Systems*, pages 2760–2769, 2018.

[20] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

[21] Akisato Kimura, Zoubin Ghahramani, Koh Takeuchi, Tomoharu Iwata, and Naonori Ueda. Few-shot learning of neural networks from scratch by pseudo example optimization. *arXiv preprint arXiv:1802.03039*, 2018.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few sample knowledge distillation for efficient network compression. *arXiv preprint arXiv:1812.01839*, 2018.

[24] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pages 11669–11680, 2019.

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

[26] Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*, 2020.

[27] Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. *CoRR*, abs/1902.05967, 2019.

[28] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[29] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R Venkatesh Babu, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. *arXiv preprint arXiv:1905.08114*, 2019.

[30] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.

[31] 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.

[32] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[33] Suraj Srinivas and François Fleuret. Knowledge transfer with jacobian matching. *arXiv preprint arXiv:1803.00443*, 2018.

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[35] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[36] Jiaxi Tang, Rakesh Shivanna, Zhe Zhao, Dong Lin, Anima Singh, Ed H Chi, and Sagar Jain. Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*, 2020.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[38] Yixing Xu, Yunhe Wang, Hanting Chen, Kai Han, XU Chunjing, Dacheng Tao, and Chang Xu. Positive-unlabeled compression on the cloud. In *Advances in Neural Information Processing Systems*, pages 2561–2570, 2019.

[39] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017.

[40] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. In *Advances in Neural Information Processing Systems*, pages 2701–2710, 2019.

[41] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisit knowledge distillation: a teacher-free framework. *arXiv preprint arXiv:1909.11723*, 2019.

[42] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[43] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[44] 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. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[45] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International Conference on Machine Learning*, pages 7543–7552, 2019.

[46] Chunting Zhou, Graham Neubig, and Jiatao Gu. Understanding knowledge distillation in non-autoregressive machine translation. *arXiv preprint arXiv:1911.02727*, 2019.

A Related works

A.1 Self-distillation (SD)

There have been increasing demands for improving the performance of SD with variety.

Distilling the information in the latent vector of teacher. In Yim *et al.* [39], two identical deep neural networks are set as a teacher and a student, respectively, to facilitate the teacher's generalization. They penalized the L2-loss from each paired layer of a student and a teacher. Zhang *et al.* [44] utilized auxiliary classifiers which are attached to additional bottlenecks and fully connected layers. They expected to encourage discrimination in lower stages. Experimental results depicted that this modified SD outperforms any other distillation methods. Ahn *et al.* [1] proposed another approach that maximizes mutual information between teacher and student networks.

Understanding the effects of SD iterations. Furlanello *et al.* [7] improved the teacher's generalization in iterative manner. Empirically, they denoted remarkable results into four folds:

1. Only using KD without ground-truth label outperforms using both the teacher's prediction and ground-truth labels.
2. KD has a similar effect to importance weighting.
3. A student distilled with teacher's output permuted except the argmax value still performs fairly well. It implies that the success of KD owes to factors other than information contained in the non-argmax output of the teacher.
4. Ensemble of iterative models performs well with low test error.

Mobahi *et al.* [26] provided theoretical analysis of SD in the circumstance where models are in Hilbert space and fitting these models is subject to l_2 regularization in those specific function space. Specifically, the effect of SD acts like a regularizer by restricting the number of basis functions which are used to represent the desired solution. In this setting, they asserted that SD iterations improve the model performance until a certain step and there exists a lower bound on the number of distillation iterations.

A.2 KD on neural machine translation

Neural machine translation (NMT) has shown remarkable performance in machine translation tasks. Promising NMT models consists of encoder-decoder architecture [2], built up with a module which automatically search the most similar representation of the target word among source words [37]. Certain model family translates the next target word by using the words before as inputs in an autoregressive (AT) manner. Due to this characteristic, the bottleneck always exists in the decoding step when inferencing.

There were approaches to decode in with non-autoregressive (NAT) methods by predicting the target word just by looking at each source word. This type of methods helps parallelism in inference because it doesn't need the former predicted word as a input reducing the inference time. However, NAT in machine translation is a very difficult task because there our diverse candidate for the true target word (i.e., multi modality problem). To mitigate this issue, the true target labels of the corpus have been replaced with labels from a pre-trained AT model [20]. With this concept, NAT has shown a significant improvements [9] . Zhou *et al.* [46] showed that knowledge distillation reduce the complexity in a data sets which helps NAT to deal with multi modality problem.

B Experimental settings

B.1 Data

Image classification. We use two popular datasets: CIFAR-10 [22] (10 classes) and CIFAR-100 [22] (100 classes) for our benchmark.

Machine translation. We use the data set commonly used by prior work as our evaluation benchmark: WMT14 English-German (En-De) [1]. We use the same settings illustrated in [46].

B.2 Models

ResNet (RN). We use the benchmark network as ResNet [11], which is a champion of the ILSVRC2015. This network uses the concept of residual learning which makes layers to learn the residual between underlying mapping and input of layer. In our experiment, we compose the ResNet with a Basickblock that consists of two consecutive 3×3 convolutional layers and for each convolutional layer batch normalizaion and ReLU activation follows sequentially.

We keep almost the same settings as [11], but one different thing from [11] is that we also control the width of networks (i.e., the number of features in each convolutional layer) to systematically dissect the effects coming from the structural factors. To control the width of network, we introduce additional widening factor k. Widening factor k multiplies the number of channels in each convolutional layer thus leading to wider network.

In our experiment, we use notation RN-n-k with n total number of layers and widening factor k. Several blocks compose a group and convolutional layers in each group share the same number of channels. More specifically, RN-20-1 is network with 20 layers which is exactly the same as [11] and RN-20-4 is wider network with widening factor $k = 4$ that extracts more features in each convolutional layer. In the process of distilling knowledge, other network hyperparmeters are fixed except α and τ.

Wide-ResNet (WRN). For the variety, we also use the Wide-ResNet [43]. The authors suggested to increase the width of convolutional layer not the depth of network. They added additional dropout for regularization effect for each residual block. In WRN, only basicblock from RN is used since, bottleneck block makes network thinner and WRN doesn't have interest in deepening the network. The sequence of convolution (Conv), batch normalization (BN) and ReLU activation follows BN-ReLU-Conv as [12]. n and k denote the total number of convolutional layers n and widening factor k in WRN, respectively.

In our experiment, we explore various Wide-ResNet structures. For all WRN, each residual block contains two 3×3 convolutional layers which is a baseline in orginal paper [43] with the best test accuracy in their experiments. We do not utilize dropout in our experiments. In all settings of over distillation, self distillation and under distillation, teacher-student pair is derived with different hyperparameter values of number of convolutional layers and widening factor, n, k respectively. In the process of distillation, we only control the hyperparameters α and τ to maintain consistency.

Autoregressive transformer (AT). In neural machine translation, input and output are sequence of words. Nearby words in a sequence affect each other. Autoregressive transformer (AT) models capture this phenomena by using the previous predicted word as input to predict the current word [37]. It is widely known that AT models can mitigate the issue of high computational burdens in training through parallelism while recurrent models such as recurrent neural networks, long short-term memory [16] and gated recurrent [5] are not.

The auto regressive model used in our experiments are applied based on transformer model. We keep the same settings of building AT models in [46] (Table 10).

Non-autoregressive transformer (NAT). For non-autoregressive model (NAT), we have used slightly shifted version of vanilla NAT model [9] whose official implementation is in Fairseq [2]. The overall architecture of original vanilla NAT is nearly identical as the Transformer except it additionally predicts fertility in the encoding step and it generates the output in non-autoregressive manner in the decoding step. However, in our paper, we don't utilize the fertility value, but simply copy the encode embedding to the decoder. We keep the same settings of building NAT models in [46](Table 10)

[1]http://www.statmt.org/wmt14/translation-task.html

[2]https://github.com/pytorch/fairseq

Models	NAT-base	AT-base	AT-big
d_{model}	512	512	1024
d_{hidden}	2048	2048	4096
n_{layer}	6	6	6
n_{heads}	8	8	16
$p_{dropout}$	0.3	0.3	0.3

Table 10: Hyperparameters of AT, NAT models. We utilize the same notation from Vaswani *et al.* [37]. d_{model} indicates the dimension of key, value and query dimension. d_{hidden} stands for the hidden dimension of feed forward network inside the sub-layer. n_{layer}, n_{heads}, $p_{dropout}$ indicate number of encoder/decoder layers and multi-head attention module and probability of dropout respectively.

B.3 Training setups

ResNet. We run 200 epochs for each model with optimizer SGD with momentum 0.9, initial learning rate $\gamma = 0.1$ and weight decay 1×10^{-4}. Also, we applied step decay for learning rate in 100 and 150 epoch by 0.1. On the other side, scale and shift parameters β and γ in BN were trained with momentum 0.99 and without weight decay. We use the same data augmentation policies in [34].

Wide-ResNet. We explored various types of WRN-n-k in experiment with different hyperparameter values of n and k. Data augmentation method follows from [34] and the same training setup was used in WRN as RN. We run 200 epochs using SGD with momentum 0.9, initial learning rate $\gamma = 0.1$, dropping 0.1 in 100 and 150 epoch, weigh decay 1×10^{-4} and parameters for batch normalization momentum with 0.99 without weight decay. After exploring various values of hyperparameters α and τ, we set those values as 1.0 and 20 respectively to reveal the benefit from the number of training data in knowledge distillation.

Autoregressive transformer (AT). We have almost kept identical configuration with [36] to train auto regressive transformer [37]. During Training, we run 70 epochs for each model and used Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1e - 8$. We adopt inverse square root scheduler with 4000 warm up updates and maximum learning rate 0.0005. We utilize the label smoothing as 0.1 and treated the last model as our best model. We decode AT model using beam-search with size 5.

Non-autoregressive transformer (NAT). We train vanilla non auto regressive transformer [9] 70 epochs with same optimizer setting with AT models. Also adopt inverse square scheduler, but with 10000 warm up updates with label smoothing 0.1. We treat the last model trained 90 epochs as our best model. We have not use any advanced decoding technique such as beam search, but use greedy decoding.

C Details of the section 2

In this section, we discuss the things that we do not show in section 2.

528 **C.1 The details of Equation 2**

$$
\begin{aligned}
\mathcal{L} &= (1-\alpha)\mathcal{L}_{CE}(p^s(1), q) + \alpha\mathcal{L}_{KD}(p^s(\tau), p^t(\tau)) \\
&= (1-\alpha)\left[-\sum_j q_j log p^s(1)_j\right] + \alpha\tau^2\left[\sum_j (p^t(\tau)_j log p^t(\tau)_j - p^t(\tau)_j log p^s(\tau)_j)\right] \\
\frac{\partial\mathcal{L}}{\partial z_k^s} &= (1-\alpha)\left[-\sum_j q_j \frac{\partial log p^s(1)_j}{\partial z_k^s}\right] + \alpha\tau^2\left[-\sum_j p^t(\tau)_j \frac{\partial log p^s(\tau)_j}{\partial z_k^s}\right] \\
&= (1-\alpha)\left[-\sum_j \frac{q_j}{p^s(1)_j}\frac{\partial p^s(1)_j}{\partial z_k^s}\right] + \alpha\tau^2\left[-\sum_j \frac{p^t(\tau)_j}{p^s(\tau)_j}\frac{\partial p^s(\tau)_j}{\partial z_k^s}\right] \text{ where } p^s(\tau)_j = \frac{e^{z_j^s/\tau}}{\sum_i e^{z_i^s/\tau}} \\
&= (1-\alpha)\left[-\frac{q_k}{p^s(1)_k}\frac{\partial p^s(1)_k}{\partial z_k^s}\right] + \alpha\tau^2\left[-\frac{p^t(\tau)_k}{p^s(\tau)_k}\frac{\partial p^s(\tau)_k}{\partial z_k^s} - \sum_{j\neq k}\frac{p^t(\tau)_j}{p^s(\tau)_j}\frac{\partial p^s(\tau)_j}{\partial z_k^s}\right] \\
&\qquad\qquad (\because q_j = 0 \text{ for all } j \neq k) \\
&= (1-\alpha)(p^s(1)_k - q_k) + \alpha\tau^2\left[\frac{1}{\tau}(p^s(\tau)_k - p^t(\tau)_k)\right] \\
&\qquad \left(\because \frac{\partial p^s(\tau)_j}{\partial z_k^s} = \partial\left(\frac{e^{z_j^s/\tau}}{\sum_i e^{z_i^s/\tau}}\right)/\partial z_k^s = \begin{cases} \frac{1}{\tau}p^s(\tau)_k(1-p^s(\tau)_k), & \text{if } j = k \\ -\frac{1}{\tau}p^s(\tau)_j p^s(\tau)_k, & \text{otherwise} \end{cases}\right)
\end{aligned} \tag{5}
$$

529 **C.2 Proof of Theorem 1**

$$
\begin{aligned}
\lim_{\tau\to\infty}\tau(p^s(\tau)_k - p^t(\tau)_k) &= \lim_{\tau\to\infty}\tau\left(\frac{e^{z_k^s/\tau}}{\sum_{j=1}^K e^{z_j^s/\tau}} - \frac{e^{z_k^t/\tau}}{\sum_{j=1}^K e^{z_j^t/\tau}}\right) \\
&= \lim_{\tau\to\infty}\tau\left(\frac{1}{\sum_{j=1}^K e^{(z_j^s - z_k^s)/\tau}} - \frac{1}{\sum_{j=1}^K e^{(z_j^t - z_k^t)/\tau}}\right) \\
&= \lim_{\tau\to\infty}\tau\left(\frac{\sum_{j=1}^K e^{(z_j^t - z_k^t)/\tau} - \sum_{j=1}^K e^{(z_j^s - z_k^s)/\tau}}{\left(\sum_{j=1}^K e^{(z_j^s - z_k^s)/\tau}\right)\left(\sum_{j=1}^K e^{(z_j^t - z_k^t)/\tau}\right)}\right) \\
&= \lim_{\tau\to\infty}\left(\frac{\sum_{j=1}^K \tau\left(e^{(z_j^t - z_k^t)/\tau} - 1\right) - \sum_{j=1}^K \tau\left(e^{(z_j^s - z_k^s)/\tau} - 1\right)}{\left(\sum_{j=1}^K e^{(z_j^s - z_k^s)/\tau}\right)\left(\sum_{j=1}^K e^{(z_j^t - z_k^t)/\tau}\right)}\right) \\
&= \frac{1}{K^2}\sum_{j=1}^K \left((z_k^s - z_j^s) - (z_k^t - z_j^t)\right) \\
&\qquad \left(\because \lim_{\tau\to\infty} e^{(z_j^f - z_k^f)/\tau} = 1, \lim_{\tau\to\infty}\tau\left(e^{(z_j^f - z_k^f)/\tau} - 1\right) = z_j^f - z_k^f\right)
\end{aligned} \tag{6}
$$

$$
\lim_{\tau\to 0} = \tau(p^s(\tau)_k - p^t(\tau)_k) = 0 \;\; (\because -1 \le p^s(\tau)_k - p^t(\tau)_k \le 1) \tag{7}
$$

C.3 Detailed values of Figure 2

Table 11, Table 12, and Table 13 show the detailed values in Figure 2.

alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\tau = 1$	99.53	99.54	99.55	99.56	99.54	99.56	99.53	99.50	99.46	99.34
$\tau = 3$	99.37	99.09	98.69	98.33	97.85	97.43	96.84	96.26	95.76	95.05
$\tau = 5$	99.32	99.07	98.70	98.19	97.66	96.96	96.18	95.11	93.91	92.63
$\tau = 20$	99.33	99.13	98.96	98.63	98.25	97.87	97.29	96.33	95.12	92.76
$\tau = \infty$	94.30	93.98	94.19	94.25	94.39	94.41	94.41	94.52	94.41	94.56

Table 11: Training accuracy on CIFAR100 (Teacher: WRN-28-4 & Student: WRN-16-2).

alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\tau = 1$	72.79	72.56	72.80	72.70	72.84	72.68	72.78	72.60	72.87	72.90
$\tau = 3$	73.76	73.90	73.88	74.30	74.18	74.64	74.78	74.32	74.35	74.24
$\tau = 5$	73.84	74.00	74.36	74.54	74.74	74.54	75.17	74.84	75.24	74.88
$\tau = 20$	73.51	73.94	74.34	74.54	74.64	74.86	75.05	74.86	75.26	75.76
$\tau = \infty$	73.56	73.96	74.60	74.81	74.49	74.78	74.79	75.05	75.08	75.51

Table 12: Testing accuracy on CIFAR100 (Teacher: WRN-28-4 & Student: WRN-16-2).

alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\tau = 1$	0.0545	0.0545	0.0545	0.0547	0.0542	0.0543	0.0542	0.0542	0.0539	0.0541
$\tau = 3$	0.0539	0.0536	0.0529	0.0521	0.0513	0.0507	0.0496	0.0487	0.0480	0.0473
$\tau = 5$	0.0540	0.0536	0.0528	0.0524	0.0515	0.0506	0.0496	0.0482	0.0472	0.0454
$\tau = 20$	0.0544	0.0540	0.0540	0.0536	0.0527	0.0522	0.0515	0.0502	0.0482	0.0464
$\tau = \infty$	0.0474	0.0468	0.0469	0.0473	0.0471	0.0469	0.0470	0.0471	0.0470	0.0473

Table 13: The teacher's average entropy of training samples which a student predict correctly on CIFAR100 (Teacher: WRN-28-4 & Student: WRN-16-2).

C.4 Model calibration

In this subsection, we evaluate the calibration effects of KD as τ increases. To measure calibration, we use the estimated expected calibration error (ECE), which is a quantitative metric of calibration [10, 28], in Table 14. The results demonstrate that a student with large τ attempts to follow the logit distribution of the teacher.

	teacher	CE	KD (τ=1)	KD (τ=3)	KD (τ=5)	KD (τ=20)
ECE	0.86	3.40	4.64	0.61	0.63	0.78

Table 14: ECE of the training samples. Here, (student, teacher) is (WRN-28-4, WRN-16-2), and all student models are trained with $\alpha = 1.0$.

C.5 Various pairs of teachers and students

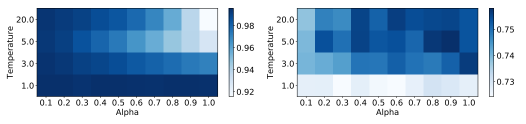
We provide the results that support the Figure 2 in various pairs of teachers and students (Figure 5). All figures depict that higher the value of α, smaller is the over-fitting problem (i.e., low training accuracy, but high test accuracy) under the condition that τ is 20.

C.6 The regularization effects of α

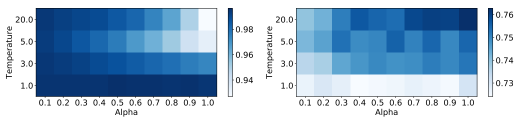
Here, we further investigate the regularization effects of α via the perspective of logit difference (TLD). Figure depict the probability density function (pdf) from the histogram of the TLD over the entire training data, which keeps the same setting in section 2.

Increasing α has similar results compared to that of τ (Figure 6). In the case of UD and SD, a student learns the teacher's logit distribution almost fully when α gets closer to 1.0. On the other side, when the student is more straightforward than the teacher, the student is not able to learn the exact teacher's logit distribution. As we've already mentioned in section 2, it is due to the bottleneck coming from the student's low complexity.

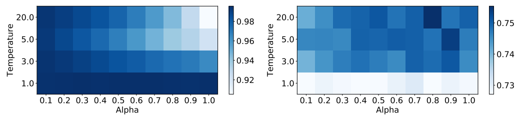
Figure 5: **Regularization effects of α and τ** on CIFAR-100 when (a) (teacher, student) = (WRN-16-4, WRN-16-2), (b) (teacher, student) = (WRN-16-6, WRN-16-2), (c) (teacher, student) = (WRN-28-2, WRN-16-2), and (d) (teacher, student) = (WRN-40-2, WRN-16-2). The left grid maps presents training top1 accuracies, and the right grid maps presents test top1 accuracies.



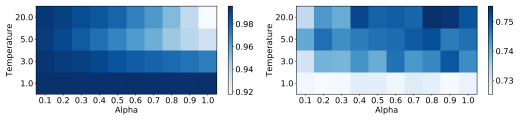
(a) T: WRN-16-4 & S: WRN-16-2



(b) T: WRN-16-6 & S: WRN-16-2

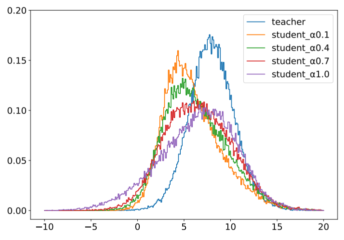


(c) T: WRN-28-2 & S: WRN-16-2

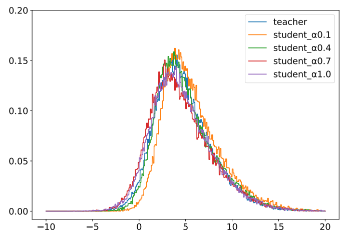


(d) T: WRN-40-2 & S: WRN-16-2

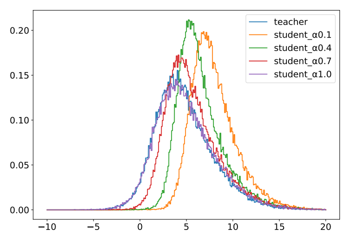
Figure 6: Pdf of TLD. All students are trained with τ=20, and all teachers are trained with CE. Both models share other training recipes such as learning rate, batch size, and weight decay.



(a) WRN-28-4 to WRN-16-2.



(b) WRN-16-2 to WRN-16-2.



(c) WRN-16-2 to WRN-28-4.

C.7 Example re-weighting [36]

In this subsection, we evaluate the example re-weighting of KD based on teacher model's prediction confidence on the ground-truth class [36]. Refer to [36], we raise a question of whether the relationship between a re-weighting value of a sample and the confidence of teacher on the sample keeps positive during in the course of training (Figure 7). Here, x-axis means the softened softmax value of teacher for ground-truth class, i.e., $p^t(\tau)_k$, and y-axis means the log value of re-weight factor when $\alpha = 1$, i.e., $\tau\left(\frac{p^t(\tau)_k - p^s(\tau)_k}{1-q^s(1)_k}\right)$.

We find that there is a positive correlation between the effect or example re-weighting and the teacher's softened top1 prediction. Especially, it gets more stronger in the early stage of training (Figure 7). After 1 epoch, this trend continues and no significant changes happen in the epoch of learning rate decay.

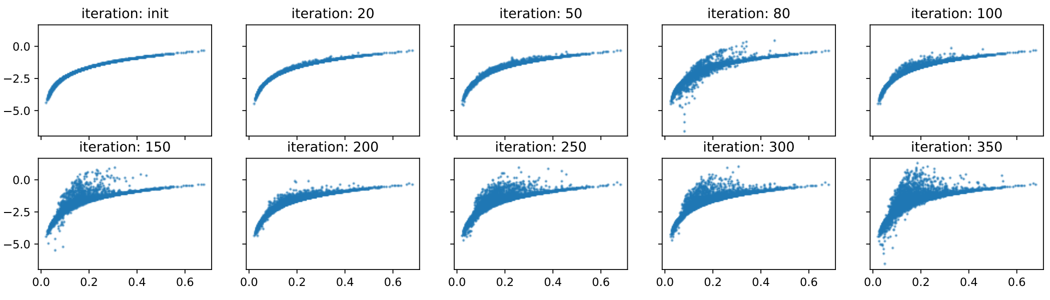
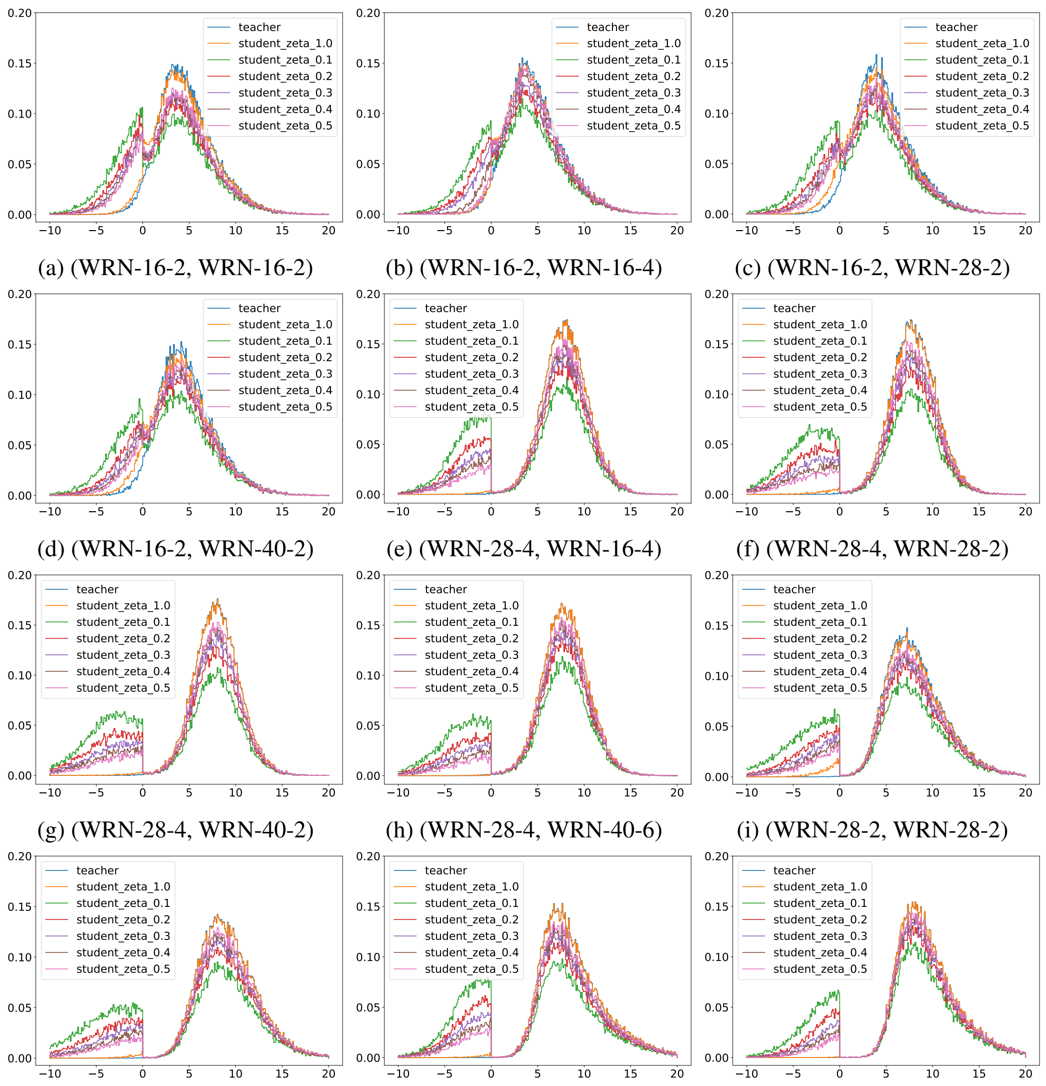


Figure 7: Re-weighting factor scatterplot in the first epoch.

D Details of the section 3

In this section, we discuss the things that we do not show in section 3.

D.1 Pdf of TLD varying the amount of training data



(a) (WRN-16-2, WRN-16-2) (b) (WRN-16-2, WRN-16-4) (c) (WRN-16-2, WRN-28-2)

(d) (WRN-16-2, WRN-40-2) (e) (WRN-28-4, WRN-16-4) (f) (WRN-28-4, WRN-28-2)

(g) (WRN-28-4, WRN-40-2) (h) (WRN-28-4, WRN-40-6) (i) (WRN-28-2, WRN-28-2)

(j) (WRN-40-2, WRN-40-2) (k) (WRN-16-4, WRN-16-4) (l) (WRN-16-6, WRN-16-6)

Figure 8: All teacher models are trained with CE, and dataset has $\delta = 1.0$. In this setting, we train all student models with FKD.

D.2 Comparison of top1 accuracies on CIFAR-100

E Details of the section 4

In this section, we provide the details of HCL described in section 4.

Table 15: Comparison of top1 accuracies on CIFAR-100 with a family of ResNets. 'Vanilla' indicates the standard SGD training with CE. All teacher models are trained with the dataset whose (δ, ζ) is equal to $(1.0, 1.0)$. We report the best result over 3 individual runs with different initializations.

	teacher	student	$\zeta, \delta = 1.0$	$\delta = 1.0$					$\zeta = 1.0$				
				ζ=0.1	ζ=0.2	ζ=0.3	ζ=0.4	ζ=0.5	δ=0.1	δ=0.2	δ=0.3	δ=0.4	δ=0.5
Vanilla	None	RN-20-1	72.56	42.51	54.76	60.15	63.74	66.65	9.13	17.06	24.99	32.33	39.55
		RN-20-4	76.89	41.10	56.54	63.69	67.25	69.93	8.99	17.35	25.51	33.15	41.03
UD	RN-20-1	RN-20-2	69.57	63.85	68.51	69.91	70.25	69.96	44.15	61.21	65.46	68.93	69.4
		RN-20-3	69.48	65.46	69.12	70.49	70.64	70.35	49.98	64.73	67.99	69.92	69.60
		RN-20-4	69.98	66.57	69.93	71.10	70.74	70.66	52.77	65.86	68.86	69.81	69.93
		RN-50-1	69.63	59.66	66.87	68.82	69.51	70.34	36.18	55.23	61.84	65.15	67.44
		RN-110-1	70.98	60.09	67.78	69.61	70.13	70.73	32.83	53.08	64.18	67.25	68.21
		RN-152-1	71.12	58.49	68.39	69.81	70.53	70.53	30.07	57.96	64.72	67.30	68.52

Table 16: Comparison of top1 accuracies on CIFAR-10 with different pairs of teacher and student models. 'Vanilla' indicates the standard SGD training with CE. All teacher models are trained with the dataset whose (δ, ζ) is equal to $(1.0, 1.0)$. We report the best result over 3 individual runs with different initializations.

	teacher	student	$\zeta, \delta = 1.0$	$\delta = 1.0$					$\zeta = 1.0$				
				ζ=0.1	ζ=0.2	ζ=0.3	ζ=0.4	ζ=0.5	δ=0.1	δ=0.2	δ=0.3	δ=0.4	δ=0.5
Vanilla	None	WRN-16-2	93.83	80.05	85.46	88.93	90.12	91.21	10.00	19.89	29.58	38.85	47.06
		WRN-16-4	94.70	80.41	87.16	89.53	91.03	92.30	10.00	19.88	29.61	38.88	47.40
		WRN-16-6	94.99	81.09	87.17	89.99	91.62	92.83	10.00	19.86	29.64	38.89	47.52
		WRN-28-2	94.47	79.86	86.91	89.51	90.93	91.87	10.00	19.89	29.55	38.93	47.34
		WRN-40-2	94.80	79.43	86.63	89.69	90.99	92.12	10.00	19.85	29.58	38.89	47.31
OD	WRN-28-4	WRN-16-2	94.54	83.14	89.31	91.31	92.23	93.09	10.00	20.72	32.34	42.10	52.90
		WRN-16-4	95.55	86.30	90.95	92.51	93.71	94.17	10.00	24.72	37.05	49.92	61.50
		WRN-28-2	95.13	84.28	89.67	91.91	93.05	93.61	10.00	20.40	31.34	42.76	53.86
		WRN-40-2	95.05	83.76	89.60	91.98	93.21	93.67	10.00	20.27	30.74	41.86	53.81
SD		WRN-16-2	93.98	86.90	91.24	92.04	93.09	93.29	15.55	45.32	63.96	72.99	77.87
		WRN-16-4	94.84	88.15	92.38	93.50	94.16	94.30	13.17	37.25	59.24	71.41	76.49
		WRN-16-6	95.52	88.16	92.86	94.04	94.48	94.81	11.75	38.83	54.96	69.89	75.72
		WRN-28-2	94.70	84.65	89.95	91.85	92.87	93.88	10.00	25.29	38.32	52.76	63.01
		WRN-40-2	95.09	83.92	90.21	91.72	92.64	94.00	10.00	20.92	33.05	45.33	57.07
		WRN-28-4	95.69	85.43	91.47	93.15	94.15	94.56	10.00	22.02	35.00	48.37	61.12
UD	WRN-16-2	WRN-16-4	94.24	88.46	92.31	93.34	93.77	94.05	23.96	57.65	76.53	81.33	88.14
		WRN-16-6	94.44	89.04	92.67	93.70	94.10	94.14	30.04	62.36	78.79	83.77	89.03
		WRN-28-2	94.04	86.90	91.92	92.95	93.44	93.56	12.97	45.46	60.98	70.75	79.75
		WRN-40-2	94.12	86.94	91.78	93.00	93.32	93.52	12.44	42.63	59.74	69.54	77.21
		WRN-28-4	94.26	88.43	92.67	93.50	93.88	94.03	14.96	53.59	70.62	78.90	85.25

Class similarity matrix. [29] To artificially soften the label vector via the teacher's prior, we first obtain the class similarity matrix from the teacher. The class similarity is calculated as follows:

$$C(i,j) = \frac{w_i^T w_j}{||w_i|| ||w_j||}$$

where $C(i,j)$ is the (i,j) elements in class similarity matrix C, and w_i is the i-th row vector of fully-connected layer's weight matrix.

Crafting labels via Dirichlet sampling [29] [29] proposed a method of crafting the labels via Dirichlet distribution whose the concentration parameter α is considered as the row vector α_k corresponding to class k. Here, they handle the amount of distillation with a hyperparameter β:

$$p(s) = Dir(K, \beta \times \alpha)$$

where K is the number of classes, and β is a scaling factor.

In our work, we conduct an experiment with the class similarity matrix of WRN-28-4 and $\beta = 1.0$.