

# Teaching What You Should Teach: A Data-Based Distillation Method

Shitong Shao<sup>1</sup>, Huanran Chen<sup>1</sup>, Zhen Huang<sup>2</sup>, Linrui Gong<sup>3</sup>, Shuai Wang<sup>4</sup> and Xinxiao Wu<sup>1\*</sup>

<sup>1</sup>Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science and Technology, Beijing Institute of Technology, China

<sup>2</sup>Univeristy of Science and Technology of China, Hefei, China

<sup>3</sup>Hunan University, Hunan, China

<sup>4</sup>Tsinghua University, Beijing, China

1090784053sst@gmail.com, huanranchen@bit.edu.cn, zhenhuang@mail.ustc.edu.cn,  
linruihong965@gmail.com, s-wang20@mails.tsinghua.edu.cn, wuxinxiao@bit.edu.cn

## Abstract

In real teaching scenarios, an excellent teacher always teaches what he (or she) is good at but the student is not. This gives the student the best assistance in making up for his (or her) weaknesses and becoming a good one overall. Enlightened by this, we introduce the “Teaching what you Should Teach” strategy into a knowledge distillation framework, and propose a data-based distillation method named “TST” that searches for desirable augmented samples to assist in distilling more efficiently and rationally. To be specific, we design a neural network-based data augmentation module with priori bias to find out what meets the teacher’s strengths but the student’s weaknesses, by learning magnitudes and probabilities to generate suitable data samples. By training the data augmentation module and the generalized distillation paradigm alternately, a student model is learned with excellent generalization ability. To verify the effectiveness of our method, we conducted extensive comparative experiments on object recognition, detection, and segmentation tasks. The results on the CIFAR-100, ImageNet-1k, MS-COCO, and Cityscapes datasets demonstrate that our method achieves state-of-the-art performance on almost all teacher-student pairs. Furthermore, we conduct visualization studies to explore what magnitudes and probabilities are needed for the distillation process.

## 1 Introduction

Deep neural networks have been widely applied in many fields, such as computer vision [Zagoruyko and Komodakis, 2016b; Wicczorek *et al.*, 2021; Chen *et al.*, 2023], natural language processing [Zhao *et al.*, 2020; Dai *et al.*, 2019], reinforcement learning [Mnih *et al.*, 2013] and speech signal processing [Purwins *et al.*, 2019]. However, the improving performance of the deep learning model is always achieved by the increasing size of the model, which makes

it impractical to deploy the large-scale model further in real-world scenarios, especially on small devices [He *et al.*, 2022; Devlin *et al.*, 2018]. To alleviate this problem, researchers have developed a series of model compression methods, such as parameter pruning [Frankle and Carbin, 2019], quantization [Wu *et al.*, 2016], and knowledge distillation [Hinton *et al.*, 2015].

Inspired by the teaching process in the human world, knowledge distillation is firstly proposed by [Hinton *et al.*, 2015], which aims to improve the target model or so-called student model, by teaching this model using a better teacher model instead of only ground truth labels. There are two mainstream approaches to distilling knowledge from the teacher to the student. One is the logit-based distillation, where the student is supervised not only by the ground truth labels but also the output of the teacher [Zhao *et al.*, 2022; Huang *et al.*, 2022]. The other is the feature-based distillation, which not only aligns the output of the student and the teacher, but also aligns the activation map in some layers [Tung and Mori, 2019; Ahn *et al.*, 2019]. These methods have achieved remarkable progress on learning efficient and effective models in the past decades, and the theoretical and practical systems of these two research directions are pretty well established. Thus, it is difficult for scholars to come up with new algorithms to surpass the past ones and further improve the student’s performance.

In this work, we rethink the knowledge distillation algorithm from a new perspective called data-based distillation. In a real-world teaching scenario, if logit-based distillation and feature-based distillation can be treated as teaching in different styles and ways, then data-based distillation can be seen as teaching knowledge in different fields. Commonly, a good teacher will decide which subject to teach based on the student’s own deficiencies, and, of course, the teacher should at least be proficient in that subject. This gives the student the knowledge it most urgently needs and drives it to improve its abilities in the corresponding weak subject. Thus, we can introduce this teaching idea to knowledge distillation and let the teacher only teach the knowledge that it should teach to boost the performance and effectiveness of the student.

More specifically, we propose a novel data-based distillation method, named “Teaching what you Should Teach (TST),” which can improve the student’s generalization abil-

\*Corresponding author

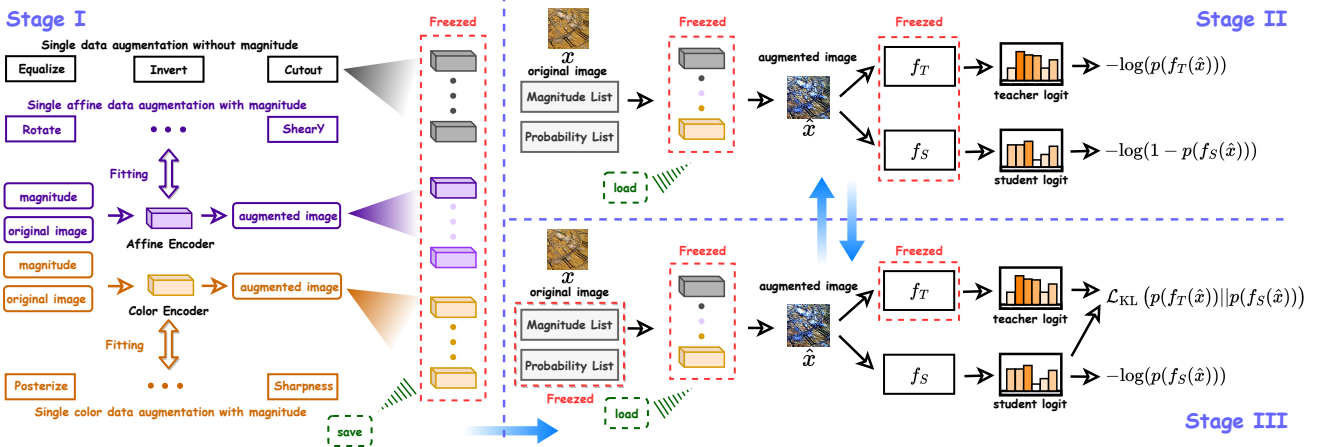


Figure 1: The overall framework of TST. In Stage I, we introduce priori bias about data augmentation into the neural network-based data augmentation module. Then, we switch between updating magnitudes and probabilities in Stage II and  $f_S$  in Stage III to make up for as many of the weaknesses of the student as possible.

ity by generating a series of new samples that the teacher excels at but the student does not. We design a mechanism that contains a data augmentation encoder that aims to generate augmented samples that the teacher is good at while the student is not good at; then, the teacher utilizes these augmented samples to guide the student. Note that this encoder is differentiable and stable, and the safety and diversity of generated samples can be ensured since it introduces manual data augmentations (e.g., Translate, Rotate, and Solarize) and employs the microscopic property of meta-encoders. In addition, we assume that it is easy for TST to generate samples that meet its requirements, which is difficult for the student to learn. Thus, we make the number of training iterations of the encoder much smaller than that of the student to ensure the distilled students have the most robust performance improvement.

We conduct extensive experiments on image classification (CIFAR-100 [Krizhevsky and Hinton, 2009] and ImageNet-1k [Russakovsky *et al.*, 2015]), object detection (MS-COCO [Lin *et al.*, 2014]), and semantic segmentation (Cityscapes [Cordts *et al.*, 2016]) tasks. As a result, TST achieves state-of-the-art (SOTA) performance in all quantitative comparison experiments with fair comparison, indicating that data-based distillation is feasible and needs more scholarly attention.

Our contribution can be summarized as follows:

- We rethink knowledge distillation from a data perspective, which not only inspires us to draw on real-world teaching scenarios but also helps us come up with a new idea, improving knowledge distillation by teaching methods as well as using more desirable data.
- We propose a novel data-based distillation method named TST to help search for augmented samples that are suitable for distillation. In TST, we develop a differentiable and stable data augmentation encoder to generate more data like that and add priori bias into this encoder to ensure its stability and interpretability.

- We have achieved SOTA performance on a range of vision tasks and done plenty of ablation studies to verify the effectiveness of TST. Besides, we visualize the characteristic information of the data augmentation encoder learned by TST and discover some patterns in it.

## 2 Related Work

**Categorize Knowledge Distillation.** We classify knowledge distillation algorithms into three categories: logit-based distillation, feature-based distillation, and data-based distillation. Past researchers have explored knowledge distillation only at the logit or feature levels. Specifically, vanilla KD [Hinton *et al.*, 2015], DKD [Zhao *et al.*, 2022], and DIST [Huang *et al.*, 2022] measure the difference in probability distribution between the student’s logit and the teacher’s logit by designing different forms of functions. And Fit-Net [Romero *et al.*, 2014], ATKD [Zagoruyko and Komodakis, 2016a], and CRD [Tian *et al.*, 2020] apply encoders or novel distance metric functions, or both to perform knowledge transfer at the feature level. However, these works focus only on “how to teach” and ignore “teach what”, which can be analogized to the data level. We categorize knowledge distillation algorithms on the data level as data-based distillation and find it has not been investigated in the generic distillation framework [Yin *et al.*, 2020; Fang *et al.*, 2022]. To fill this gap, we turn to focus on designing a data-based distillation method and use a data augmentation encoder to obtain new samples from the original samples that can be recognized by the teacher but not by the student.

**Learnable Data Augmentation.** Data augmentation utilizes affine and color transformations to expand the training dataset. Traditional manual data augmentation, including RandAugment [Cubuk *et al.*, 2020] and AutoAugment [Cubuk *et al.*, 2019], is non-differentiable and unlearnable. In recent years, differentiable data augmentation [Li *et al.*, 2020] has ensured the learnability of data augmentation through a series of numerical approximations, but its

overly complex framework design has hindered its further application. TeachAugment [Suzuki, 2022] effectively solves the problem of unlearnability of data augmentation through a neural network-based data augmentation module, but the instability of this module leads to the need to impose a series of additional constraints to assist training. Thus, in TST, we add priori bias to the neural network-based data augmentation module and then freeze it in standard training. This ensures that the new data encoder is interpretable and stable.

### 3 Teach What You Should Teach

In this section, we first introduce the total framework of TST. Then, we present the simple knowledge distillation in Stage III, the search for desirable data in Stage II, and how to introduce priori bias into the data augmentation module in Stage I. Finally, the concrete implementation of the data augmentation module is described.

#### 3.1 Total Framework

As illustrated in Fig. 1, TST can be divided into three stages, namely Stage I, Stage II, and Stage III. To be specific, Stage I introduces the reasonable priori bias into the neural network-based data augmentation module before formal training, which helps ensure the stability of training in Stage II and Stage III, and improves the student’s ultimate generalization ability. Then, Stages II and III obtain an excellent student by alternately training them. In Stage II, we initialize the learnable magnitudes and probabilities, and search augmented samples that the teacher is good at but the student is not, by minimizing the cross-entropy of the teacher and maximizing the cross-entropy of the student. We expect that the student will serve its deficiencies in Stage III and obtain good generalization ability after training. In particular, Stage III can be viewed as a simple knowledge distillation framework that only applies a traditional cross-entropy loss and a Kullback-Leibler divergence to optimize the student. Therefore, the core of TST is to find more desirable training data for Stage III through Stage I and Stage II. The more detailed algorithmic procedure of TST can be found in Appendix C.

#### 3.2 Simple Knowledge Distillation

Stage III is a simple knowledge distillation framework, which can be interpreted as follows: given a teacher  $f_T$  with parameter  $\theta_T$  and a student  $f_S$  with parameter  $\theta_S$ , when a training set  $\mathcal{X}$  including the original samples and augmented samples are utilized for training, we can sample the mini-batch  $\{x^k\}_{k=1}^B$  ( $B$  denotes the batch size) from it and get the student output  $\{f_S(x^k)\}_{k=1}^B$  and the teacher output  $\{f_T(x^k)\}_{k=1}^B$  by forward propagation. Let us define the normalized exponential function  $p$ , i.e., softmax, to calculate the probability distribution of the model output. The goal of knowledge distillation is to minimize the cross-entropy loss  $\mathcal{L}_{CE}$  between  $\{p(f_S(x^k))\}_{k=1}^B$  and the ground truth label  $\{y^k\}_{k=1}^B$ , and the Kullback-Leibler divergence  $\mathcal{L}_{KL}$  between  $\{p(f_T(x^k)/\tau)\}_{k=1}^B$  and  $\{p(f_S(x^k)/\tau)\}_{k=1}^B$ , where  $\tau$  refers to the temperature weight. Thus, the total loss function  $\mathcal{L}_{total}$

is formulated as

$$\mathcal{L}_{total} = \frac{1}{B} \sum_{k=1}^B w_{ce} \mathcal{L}_{CE} \left( p(f_S(x^k)), y^k \right) + w_{kl} \tau^2 \mathcal{L}_{KL} \left( p(f_T(x^k)/\tau) || p(f_S(x^k)/\tau) \right), \quad (1)$$

where  $w_{ce}$  and  $w_{kl}$  are balanced weights. Note that Eq. 1 can also be appended with other distillation loss functions, e.g. Mean Squared Error function ( $\mathcal{L}_{MSE}$ ), to improve the performance of distillation, which is applied in TST for the object detection task.

#### 3.3 Search Desirable Data

The goal of TST is to assist the student in overcoming its deficiencies in Stage III by discovering more desirable data that the teacher excels at but the student does not in Stage II. We denote the neural network-based data augmentation module  $f_{DE}$  contains a data augmentation meta-encoder set  $\{f_E^i\}_{i=1}^N$  with a parameter set  $\{\theta_E^i, \theta_p^i\}_{i=1}^N \cup \{\theta_m^i\}_{i=1}^{N-N_{nl}}$ , where  $N$  and  $N_{nl}$  refer to the total number of all sub-policies and the total number of sub-policies without magnitudes, respectively. Note that each  $f_E$  has the frozen parameter  $\theta_E$  containing priori bias, and its other parameters, i.e., the magnitude<sup>1</sup>  $\theta_m$  and the probability  $\theta_p$ , are learnable in Stage II. Therefore, as shown in the upper right corner of Fig. 1, we denote the loss function of TST in Stage II as

$$\mathcal{L}_{TST} = \frac{1}{B} \sum_{k=1}^B \alpha \mathcal{L}_{CE} \left( p(f_T(\hat{x}^k)), y^k \right) + \beta \mathcal{L}_{CE} \left( 1 - p(f_S(\hat{x}^k)), y^k \right), \quad (2)$$

where  $\hat{x}$  denotes the augmented sample obtained from the original sample after  $f_{DE}$ , and  $\alpha$  and  $\beta$  denote the balanced weights. Specific details about  $f_{DE}$  will be presented in Sec. 3.5. The optimization objective of Stage II  $\min_{\{\theta_m^i, \theta_p^i\}_{i=1}^N} \mathcal{L}_{TST}$  is to let the student misclassify but let the teacher correctly classify so that the teacher can transfer the most helpful knowledge to students. To better find desirable data that the student cannot classify correctly, we use  $\mathcal{L}_{CE} (1 - p(f_S(\hat{x}^k)), y^k)$  instead of  $-\mathcal{L}_{CE} (p(f_S(\hat{x}^k)), y^k)$  to avoid non-convex optimization, and the proof can be found in Appendix D. Moreover, the parameters  $\{\theta_E^i\}_{i=1}^N$  are frozen because they contain priori bias in favor of distillation, which is imported in Stage I. As demonstrated later in Sec. 4.2, distillation can easily fail if  $\{\theta_E^i\}_{i=1}^N$  is not frozen due to the instability of the data augmentation module.

During standard distillation training, Stage II and Stage III alternate. Specifically, TST first employs Stage II to find new samples suitable for distillation, and then utilizes Stage III to improve the student’s generalization ability. Inspired by [Carlini and Wagner, 2017], we claim that it is easy to search augmented samples in Stage II that match what the teacher is good at but the student is not, while it is difficult for the student to absorb knowledge contained in the augmented samples in Stage III. Therefore, the number of iterations  $n_{encoder}$

<sup>1</sup>Magnitudes indicate the strength of the sub-policies transition. Some sub-policies do not have this variable, such as Cutout, Equalize, and Invert. These details will be introduced later.

Architecture		Same						Different		
Distillation Type	Teacher	ResNet110	ResNet110	WRN-40-2	WRN-40-2	ResNet32×4	VGG13	WRN40-2	ResNet32×4	VGG13
	Student	74.31	74.31	75.61	75.61	79.42	74.64	75.61	79.42	75.61
		ResNet20 [2016]	ResNet32	WRN-40-1 [2016b]	WRN-16-2	ResNet8×4	VGG8	ShuffleNet-V1	ShuffleNet-V1	MobileNet-V2
		69.06	71.14	71.98	73.26	72.50	70.36	70.50	70.50	64.60
Feature-based	FitNet [2014]	68.99	71.06	72.24	73.58	73.50	71.02	73.73	73.59	64.14
	ATKD [2016a]	70.22	70.55	72.77	74.08	73.44	71.43	73.32	72.73	59.40
	SPKD [2019]	70.04	72.69	72.43	73.83	72.94	72.68	74.52	73.48	66.30
	CCKD [2019]	69.48	71.48	72.21	73.56	72.97	70.71	71.38	71.14	64.86
	RKD [2019]	69.25	71.82	72.22	73.35	71.90	71.48	72.21	72.28	64.52
	VID [2019]	70.16	70.38	73.30	74.11	73.09	71.23	73.61	73.38	65.56
	CRD [2020]	71.46	73.48	74.14	75.48	75.51	73.94	76.05	75.11	69.73
	OFD [2019]	-	73.23	74.33	75.24	74.95	73.95	75.85	75.98	69.48
	ReviewKD [2021]	-	71.89	75.09	76.12	75.63	74.84	77.14	<b>77.45</b>	70.37
Logit-based	KD [2015]	70.67	73.08	73.54	74.92	73.33	72.98	74.83	74.07	67.37
	DKD [2022]	-	74.11	74.81	76.24	76.32	74.68	76.70	76.45	69.71
	DIST [2022]	69.94	73.55	74.42	75.29	75.79	73.74	75.23	75.23	68.48
Data-based	Ours	<b>72.44</b>	<b>75.04</b>	<b>75.32</b>	<b>76.75</b>	<b>76.72</b>	<b>75.03</b>	<b>77.38</b>	76.71	<b>70.82</b>

Table 1: Results on the CIFAR-100 test set. “Same” and “Different” in the first row refer to whether the model architecture is the same for teachers and students.

Architecture			Accuracy		Feature-based					Logit-based			Data-based
Teacher	Student		Teacher	Student	OFD [2019]	RKD [2019]	CRD [2020]	SRRL [2021]	ReviewKD [2021]	KD [2015]	DKD [2022]	DIST [2022]	Ours
ResNet-34	ResNet-18	Top-1	73.31	69.76	71.08	70.34	71.17	71.73	71.61	70.66	71.70	72.07	72.22
		Top-5	91.42	89.08	90.07	90.37	90.13	90.60	90.51	89.88	90.41	90.42	90.68
ResNet-50	MobileNet-V1	Top-1	76.16	70.13	71.25	-	71.37	72.49	72.56	70.68	72.05	73.24	72.31
		Top-5	92.86	89.49	90.34	-	90.41	90.92	91.00	90.30	91.05	91.12	90.70
Swin-Large	Swin-Tiny	Top-1	86.30	81.30	-	81.20	-	81.50	-	81.50	-	82.26	82.21

Table 2: Results on the ImageNet validation set. We use ResNet-34 and ResNet-50 released by Torchvision [Marcel and Rodriguez, 2010] and Swin-Large released by [Liu *et al.*, 2021] as our teacher’s pre-training weight.

of Stage II will be much smaller than the number of iterations  $n_{\text{student}}$  of Stage III. Also, Sec. 4.2 illustrates that the student trained by TST will have better performance when  $n_{\text{encoder}} \ll n_{\text{student}}$ .

### 3.4 Introduce Prior Bias into $f_{DE}$

Noise-based generative models require expensive computational costs and suffer from training instability (more detailed can be found in Appendix D.5). Thus, Stage I is applied to introduce the augmented priori bias into  $f_{DE}$  and let  $\{\theta_p^i\}_{i=1}^N \cup \{\theta_m^i\}_{i=1}^{N-N_{\text{nl}}}$  be differentiable. Thanks to priori bias, TST is able to distill a strong student in a limited number of iterations. For simplicity, we consider each meta-encoder  $f_E$  in the set  $\{f_E^i\}_{i=1}^N$  as a black box in this paragraph. In Stage I, since Equalize, Invert and Cutout do not have the property of magnitude, they do not need to be fitted and can be called directly in Stage II and Stage III. Of course, their probabilities are learnable and will still be optimized in Stage II. Then, we categorize a series of single data augmentations as follows: (a) magnitude-unlearnable transformations, including Equalize, Invert, and Cutout; (b) learnable affine transformations, including Rotate ShearX, ShearY, TranslateX, and TranslateY; (c) learnable color transformations, including Posterize, Solarize, Brightness, Color, Contrast, and Sharpness [Cubuk *et al.*, 2019; Cubuk *et al.*, 2020]. For the learnable affine transformations and color transformations, we apply Spatial Transformer Network [Jaderberg *et al.*, 2015] and Color Network for fitting, respectively. Since both types (a) and (b) are fitted in the same way, we define a set of the manual data-augmentation mappings  $\{f_A^i\}_{i=1}^{N-N_{\text{nl}}}$  that includes all single data augmentations with magnitude. For each matched pair  $f_A$  and  $f_E$  in  $\{f_A^i\}_{i=1}^{N-N_{\text{nl}}}$  and  $\{f_E^i\}_{i=1}^{N-N_{\text{nl}}}$ , we will optimize  $\mathcal{L}_{MSE}$  by

$n_{\text{fitting}}$  iterations. The loss function in Stage I can be formulated as

$$\mathcal{L}_{\text{encoder}} = \frac{1}{B} \sum_{k=1}^B \|f_A(x^k, m_r) - f_E(x^k, m_r)\|_2^2, \quad (3)$$

where  $m_r \sim \mathcal{U}(0, 1)$  refers to the magnitude.  $\mathcal{L}_{\text{encoder}}$  will be optimized to a very small error bound after  $n_{\text{encoder}}$  iterations. This guarantees that all meta-encoders are well-trained.

### 3.5 Network-Based Data Augmentation

We will describe the whole process of how  $x$  goes through  $f_{DE}$  and finally becomes  $\hat{x}$  in this sub-section. We first describe how  $f_{DE}$  calls  $\{f_E^i\}_{i=1}^N$  to accomplish the combination of various sub-generated samples, and then we present the details of Spatial Transformer Network [Jaderberg *et al.*, 2015] and Color Network.

After normalizing  $\{\theta_m^i\}_{i=1}^{N-N_{\text{nl}}}$  and  $\{\theta_p^i\}_{i=1}^N$  to  $[0, 1]$  by applying the sigmoid activation function, we employ the Relaxed Bernoulli Distribution to sample new magnitudes and probabilities for solving the non-differentiable problem [Lim *et al.*, 2019; Li *et al.*, 2020]. The process can be formulated as

$$\begin{aligned} \text{RBD}(p) &= \text{sigmoid}((\log(p) + L)/\tau_l), L \sim \text{Logistic}(0, 1), \\ \{\theta_m^i\}_{i=1}^{N-N_{\text{nl}}} &= \left\{ \text{RBD} \left( \text{sigmoid}(\theta_m^i) \right) \right\}_{i=1}^{N-N_{\text{nl}}}, \\ \{\theta_p^i\}_{i=1}^N &= \left\{ \text{RBD} \left( \text{sigmoid}(\theta_p^i) \right) \right\}_{i=1}^N, \end{aligned} \quad (4)$$

where  $\text{Logistic}$  and  $\tau_l$  stand for the logistic distribution and the temperature, respectively, and  $\tau_l$  is set as 0.05. Then, we randomly select  $N_A$  non-repeating numbers  $\{Z^i\}_{i=1}^{N_A}$  from  $\{i\}_{i=1}^N$ .  $N_A$  can be interpreted as the number of randomly sampled primitives, i.e., a larger  $N_A$  means that the model is

T→S	CM RCNN-X101→Faster RCNN-R50						RetinaNet-X101→RetinaNet-R50						T→S	FCOS-R101→FCOS-R50					
Type	Two-stage detectors						One-stage detectors						Type	Anchor-free detectors					
Method	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>	Method	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
Teacher	45.6	64.1	49.7	26.2	49.6	60.0	41.0	60.9	44.0	23.9	45.2	54.0	Teacher	40.8	60.0	44.0	24.2	44.3	52.4
Student	38.4	59.0	42.0	21.5	42.1	50.3	37.4	56.7	39.6	20.0	40.7	49.7	Student	38.5	57.7	41.0	21.9	42.8	48.6
KD [2015]	39.7	61.2	43.0	23.2	43.3	51.7	37.2	56.5	39.3	20.4	40.4	49.5	KD [2015]	39.9	58.4	42.8	23.6	44.0	51.1
COFD [2019]	38.9	60.1	42.6	21.8	42.7	50.7	37.8	58.3	41.1	21.6	41.2	48.3	FitNet [2014]	39.9	58.6	43.1	23.1	43.4	52.2
FKD [2021]	41.5	62.2	45.1	23.5	45.0	55.3	39.6	58.8	42.1	22.7	43.3	52.5	GID [2021]	42.0	<b>60.4</b>	<b>45.5</b>	25.6	45.8	54.2
DIST [2022]	40.4	61.7	43.8	23.9	44.6	52.6	39.8	59.5	42.5	22.0	43.7	53.0	FRS [2021]	40.9	60.3	<b>43.6</b>	25.7	45.2	51.2
DIST+mimic [2022]	41.8	62.4	45.6	23.4	46.1	55.0	40.1	59.4	43.0	23.2	44.0	53.6	FGD [2022]	<b>42.1</b>	-	-	<b>27.0</b>	<b>46.0</b>	<b>54.6</b>
Ours (KD)	40.5	62.4	44.1	24.0	44.6	52.1	39.9	59.6	42.8	23.3	43.8	53.3	Ours (KD)	40.1	58.3	43.2	23.9	44.1	51.6
Ours (KD+mimic)	<b>42.2</b>	<b>63.4</b>	<b>46.1</b>	<b>24.1</b>	<b>46.5</b>	<b>55.6</b>	<b>40.5</b>	<b>60.0</b>	<b>43.4</b>	<b>23.9</b>	<b>44.5</b>	<b>54.4</b>	Ours (KD+mimic)	41.0	60.0	44.3	24.9	45.0	51.8

Table 3: Results on the COCO validation set (T→S refers to the distillation from T to S). Here, the content in brackets to the right of “Ours” refers to the methods applied in the distillation process. In addition, CM RCNN-X101 stands for Cascade Mask RCNN-X101.

Method	mIoU (%)	Method	mIoU (%)
T: PSPNet-R101	78.55	S: PSPNet-R18	70.09
SKDD [2020]	74.08	SKDS [2019]	72.70
IFVD [2020]	74.54	CWD [2021]*	75.54
DIST [2022]	75.74	Ours*	<b>76.55</b>

Table 4: Results on the Cityscapes validation set. \*: The experiments are performed based on mmrazor [Contributors, 2021].

more difficult to identify the augmented samples.  $N_A$  is set as 4 by default in our experiments. We obtain  $\{x_E^i\}_{i=1}^{N_A}$  from  $x$  through all meta-encoders and get the final  $\hat{x}$  by simple sum denoted as

$$x_E^i = \begin{cases} \theta_p^{Z^i} \odot f_E(x, \theta_m^{Z^i}) - \theta_p^{Z^i} \odot x, & Z^i \leq N - N_{nl} \\ \theta_p^{Z^i} \odot f_E(x) - \theta_p^{Z^i} \odot x, & Z^i > N - N_{nl} \end{cases} \quad (5)$$

$$\hat{x} = x + \sum_{i=1}^{N_A} x_E^i,$$

where  $\odot$  denotes the element-wise product. For object detection tasks, the summation in Eq. 5 can easily lead to the undesired result that a single target object turns into multiple target objects. Therefore, we use a separate algorithm for this special task, which is explained in Appendix D.

Spatial Transformer Network  $f_{STN}$  and Color Network  $f_{CN}$  are two different neural network-based meta-encoders. Each has parameters  $\theta_E$  to store priori bias learned from Stage I. However,  $f_{STN}$  and  $f_{CN}$  employ different ways to complete the calculation. As shown in the next Eq., when  $x$  and magnitude  $m$  are fed into  $f_{STN}$ ,  $\theta_E$  (can be considered as a vector) is first concatenated with  $m$  to form a new vector  $\hat{\theta}_E$ , and then passes through a fully connected layer  $\mathcal{FC} := \mathbb{R}^{\text{len}(\theta_E)+1} \rightarrow \mathbb{R}^6$  to obtain the matrix  $A \in \mathbb{R}^{2 \times 3}$ , denoted as  $f_{STN}(x, m) = \text{Affine}(A, x)$ , where  $A = \text{reshape}(\mathcal{FC}([\theta_E, m]))$ . For  $f_{CN}$ , its forward propagation is composed of convolution and color transformation. So,  $\theta_E$  not only has a vector  $\theta_{E,V}$  but also a convolution weight  $\theta_{E,C}$  to record priori bias. When  $x$  and  $m$  are fed into  $f_{CN}$ ,  $\theta_{E,V}$  and  $m$  concatenate a new vector  $\hat{\theta}_{E,V}$ , and we let it through a fully connected layer  $\mathcal{FC} := \mathbb{R}^{\text{len}(\theta_{E,V})+1} \rightarrow \mathbb{R}^2$  to obtain the scale and shift parameters, i.e.,  $\theta_{\text{scale}}$  and  $\theta_{\text{shift}}$ . After that, the output of  $f_{CN}$  is denoted as

$$f_{CN}(x, m) = \mathcal{C}(x \odot (0.5 + \text{sigmoid}(\theta_{\text{scale}})) + \text{sigmoid}(\theta_{\text{shift}}) - 0.5),$$

where  $\mathcal{C}$  refers to the convolutional layer. In particular, in our experiments, we perform RBD on  $A$ ,  $\theta_{\text{shift}}$  and  $\theta_{\text{scale}}$  additionally to guarantee the diversity of data augmentation to prevent

$\{\theta_m^i\}_{i=1}^{N-N_{nl}}$  and  $\{\theta_p^i\}_{i=1}^N$  from converging to a locally optimal solution.

## 4 Experiment

We conduct comparison experiments on three major tasks: image classification, object detection, and semantic segmentation. The image classification datasets include CIFAR-100 [Krizhevsky and Hinton, 2009] and ImageNet-1k [Russakovsky *et al.*, 2015]; the target detection dataset includes MS-COCO [Lin *et al.*, 2014]; the semantic segmentation dataset includes Cityscapes [Cordts *et al.*, 2016]. More details about these datasets can be found in Appendix A. Besides, all the experiment results on CIFAR-100 are the average over five trials, while the related experiment results on other datasets are the average over three trials. We apply batch size 128 and initial learning rate 0.1 on CIFAR-100. And we follow the settings in [Huang *et al.*, 2022] for the ResNet34-ResNet18 pair and the ResNet50-MobileNet pair on ImageNet-1k. The settings of other classification, detection and segmentation tasks can be found in Appendix B.

### 4.1 Comparison with SOTA Methods

**Classification on CIFAR-100.** We compare many state-of-the-art feature-based and logit-based distillation algorithms on nine student-teacher pairs. For these teacher-student pairs, the teacher and student of six pairs have the same structure, and the teacher and student of the other three pairs have different architecture. The experimental results are presented in Table 1. Obviously, we can find that TST, as the only data-based distillation approach, outperforms all other algorithms on eight student-teacher pairs except for ResNet32×4-ShuffleNet-V1. Especially on three teacher-student pairs, including ResNet110-ResNet20, ResNet110-ResNet32 and VGG13-VGG8, our TST surpasses the latest state-of-the-art methods by almost more than one percent. Besides, in order to more fully demonstrate the excellent performance of TST, we conduct simulations of few-shot scenarios on CIFAR-100. Here, we follow the training settings in [Tian *et al.*, 2020] and randomly discard 25%, 50%, and 75% samples for training. As the experimental results shown in Appendix D.6, TST also performs well in this few-shot scenario.

**Classification on ImageNet-1k.** To further demonstrate whether TST can work robustly on ImageNet-1k, we conduct experiments with two different architecture pairs, including Conv-Conv and ViT-ViT pairs. For Conv-Conv pair, we consider two teacher-student pairs: ResNet34-ResNet18 and

ResNet50-MobileNet-V1 pairs, and apply the same hyperparameter settings to show the effectiveness of TST. The results are illustrated in Table 2. We can find that TST beats all state-of-the-art methods in ResNet34-ResNet18 pair but is slightly inferior to SRRL, ReviewKD and DIST in ResNet50-MobileNet-V1 pair. The possible reason is that the teacher is stronger, which causes the teacher overconfident in discriminating on the original and generating augmented samples. In our discussion, we consider this phenomenon in isolation and give two solutions for mitigating poor student performance under TST training with the stronger teacher. In particular, this is not a defect of TST, but rather due to the fact that  $\mathcal{L}_{KL}$  is not adapted to the distillation process with the stronger teacher. In fact, there are many approaches, including DIST [Huang *et al.*, 2022] and DKD [Zhao *et al.*, 2022], that have been proposed to alleviate the gap between the teacher and student. For the ViT-ViT pair, we regard Swin Transformer [Liu *et al.*, 2021], which is widely known and applied by researchers, as the model architecture in distillation. We treat Swin Transformer Tiny (Swin-Tiny) as the student and Swin-Transformer Large (Swin-Large) as the teacher. The experimental results are presented in Table 2, where TST surpasses all methods except DIST, showing that TST is applicable to ViT-based architectures.

**Detection on MS-COCO.** Comparison experiments are run on three kinds of different detectors, i.e., *two-stage detectors*, *one-stage detectors*, *anchor-free detectors*. In particular, TST introduces additional losses on Stage II that drove the student box regression to be inaccurate and the teacher box regression to be accurate. As shown in Table 3, TST breaks the vanilla KD bottleneck by locating augmented samples that are conducive to distillation. Due to the additional loss on box regression in Stage II and the fact that the detection task depends more on the network’s ability to produce good features, we believe that aligning the student and teacher feature maps will improve the performance of TST. Thus, we follow [Huang *et al.*, 2022] by adding auxiliary loss mimic, i.e., translating the student feature map to the teacher feature map by a convolution layer and supervising them utilizing  $\mathcal{L}_{MSE}$ , to the detection distillation task. Ultimately, we can conclude from Table 3 that TST based on the vanilla KD and mimic achieves the best performance on Cascade RCNN-X101-Cascade RCNN-R50 and RetinaNet-X101-RetinaNet-R50 pairs.

**Segmentation on Cityscapes.** We conduct comparative experiments of semantic segmentation on PSPNet-R101-PSPNet-R18 pair [Zhao *et al.*, 2017]. As shown in Table 4, TST outperforms all the state-of-the-art methods and demonstrates that data-based distillation is effective in the field of semantic segmentation.

## 4.2 Ablation Study

We conduct ablation studies in three aspects: **(a)** the effect of Stage II on the student performance; **(b)** the effect of different iteration numbers in Stage II; **(c)** the impact of varying data encoders on TST.

As illustrated in Table 5, no matter what kind of teacher-student pairs or whatever value of  $N_A$ , the augmented sample

Teacher	Student	Learnable	$N_A$		
			2	4	6
WRN-40-2	WRN-16-2	Yes	76.75	76.58	76.55
		No	76.41	76.05	75.66
ResNet110	ResNet20	Yes	72.34	72.44	72.43
		No	71.86	71.74	71.80

Table 5: Top-1 test accuracy (%) comparison of whether the neural network-based data augmentation module is learnable or not on CIFAR-100.

Teacher	Student	$n_{\text{encoder}}$			
		$0 \cdot \frac{ \mathcal{X} }{B}$	$1 \cdot \frac{ \mathcal{X} }{B}$	$15 \cdot \frac{ \mathcal{X} }{B}$	$30 \cdot \frac{ \mathcal{X} }{B}$
WRN-40-2	WRN-16-2	76.05	76.72	76.53	76.58
ResNet110	ResNet20	71.74	72.46	72.41	72.44

Table 6: Top-1 test accuracy (%) comparison on CIFAR-100. Here, we performed ablation experiments on  $n_{\text{encoder}}$ .  $N_A$  for all experiments in this table is set as 4.

search strategy of TST, i.e., Stage II, is always beneficial for distillation.

In addition, we suppose that TST may easily find augmented samples that the student is not good at, but it is difficult for the student to absorb the corresponding knowledge fully. Based on this conjecture, TST should perform best in the case that  $n_{\text{encoder}} \ll n_{\text{student}}$ . The experimental results in Table 6 verify our assumption. Note that  $\frac{|\mathcal{X}|}{B}$  actually refers to the iteration number within an epoch.

At last, we show the impact of different types of data encoders on the performance of TST in Table 7 and analyze it. One of the data encoders, Attack, denotes the direct manipulation of samples in a form similar to PGD [Madry *et al.*, 2018], which means that TST must let Stage II and Stage III iterate once each in turn and repeat the process. Based on the findings in Table 7, we can indicate that it is an extremely sensible choice to introduce the augmented priori bias into the data encoder and to freeze the parameters associated with the bias during distillation process.

## 5 Discussion

**How to alleviate the gap between the teacher and student?** To analyze this situation, we train two stronger teachers, including WRN-40-4 and WRN-28-10, achieve 80.7% and 82.0% Top-1 accuracy on the test set of CIFAR-100, respectively. Then, we employ these two teachers for a more in-depth exploration. As illustrated in Table 8, when  $w_{ce}$ ,  $w_{kl}$ ,  $\alpha$  and  $\beta$  take default values (i.e., 1, 1, 1 and 1), stronger teachers (i.e., WRN-40-4 and WRN-28-10) usually achieve worse performance compared to WRN-40-2. To analyze this phenomenon, we show in Fig. 3 that the teacher’s response to the ground truth label, i.e., the probabilities of correctly discriminating samples. Specifically, in each epoch of Stage III, we get the teacher’s responses to the correct category corresponding to all training samples, and plot the expectation of the responses. This indicates that the stronger the teacher is, the closer its response is to one-hot encoding and the less “dark knowledge” attached to the logit. Consequently, the performance degradation of students caused by stronger teachers can be mitigated by making the teacher logit not too close

Teacher	Student	Mode			
		Attack	Ours <sup>+</sup>	Ours <sup>++</sup>	Ours
WRN-40-2	WRN-16-2	74.76	66.01	NAN	76.58
ResNet110	ResNet20	71.15	52.91	NAN	72.44

Table 7: Top-1 test accuracy (%) comparison for different data encoders on CIFAR-100. Here, + and \* stand for not freezing all the parameters in  $\{f_E^i\}_{i=1}^N$  and not introducing bias of a priori data augmentation into  $\{f_E^i\}_{i=1}^N$ , respectively. The results “NAN” represents the gradient explosion. Besides,  $\mathcal{N}_A$  and  $n_{\text{encoder}}$  for all experiments in this table is set as 4 and  $30 \cdot \frac{|\mathcal{X}|}{B}$ , respectively.

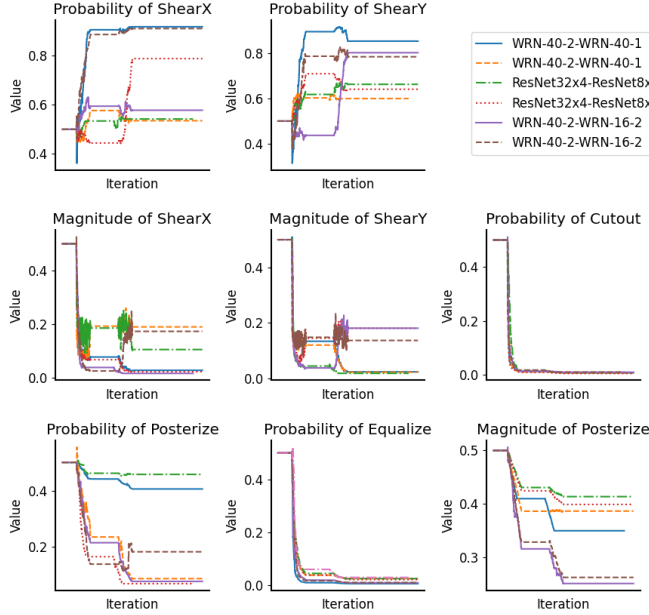


Figure 2: The plot of magnitudes and probabilities variation with iteration under TST on CIFAR-100. Here we have selected representative sub-policies, including ShearX, ShearY, Cutout, Posterize, and Equalize.

to one-hot coding. We propose two solutions: (a) decrease  $w_{ce}$  and increase  $w_{kl}$ , simultaneously, to transfer more “dark knowledge” from the teacher to the student; (b) decrease  $\alpha$  and increase  $\beta$ , simultaneously, to make the teacher not over-confident about the augmented samples searched by Stage II. As exhibited in Table 8, both methods assist in bridging the gap between the teacher and student to some extent.

**What are the strengths and probabilities of execution for augmentations that the teacher excels at, but the student does not?** To answer this question, we choose 8 magnitudes and probabilities from  $\{\theta_m^i\}_{i=1}^{N-N_{nl}}$  and  $\{\theta_p^i\}_{i=1}^N$  with distinct patterns and plot their variation with iteration in Fig. 2. The full TST visualization of search results is shown in Appendix D. Note that there are 3 teacher-student pairs, i.e., WRN-40-2-WRN-16-2, WRN-40-2-WRN-40-1 and ResNet32 $\times$ 4-ResNet8 $\times$ 4 pairs in these figures, and

Hyperparameters				Teacher	
$w_{ce}$	$w_{kl}$	$\alpha$	$\beta$	WRN-40-4	WRN-28-10
1.0	1.0	1.00	1.00	75.93	75.79
1.0	1.0	0.95	1.05	76.13	76.04
1.0	1.0	0.85	1.15	76.35	76.18
1.0	1.0	0.75	1.25	76.42	76.34
1.0	1.0	0.5	1.5	76.45	75.80
1.0	1.0	0.25	1.75	76.19	75.81
0.8	1.2	1.00	1.00	76.32	75.89
0.6	1.4	1.00	1.00	76.37	75.96
0.4	1.6	1.00	1.00	76.47	76.02
0.2	1.8	1.00	1.00	76.29	76.17

Table 8: Top-1 accuracy (%) comparison on CIFAR-100. Analytical experiments of  $w_{ce}$ ,  $w_{kl}$ ,  $\alpha$  and  $\beta$  hyperparameters. Here, all students are WRN-16-2.

each pair has two changing curves. Although the magnitudes and probabilities searched by the different models are various, they have macroscopic similarities. For instance, ShearX and ShearY converge to larger probabilities and smaller magnitudes; sub-policies without learnable magnitudes, such as Equalize and Cutout, converge to minimal probabilities. In fact, such a result is intuitive. For instance, the co-characteristics of the dataset let the magnitudes and probabilities searched by the data augmentation methods (e.g., PBA and AutoAugment) on one model can also be applied to another one. Besides, each curve in Fig. 2 differs at a microscopic level due to the difference between teacher-student pairs and the stochastic nature of deep learning.

## 6 Conclusion

In this paper, inspired by realistic teaching scenarios, we design a data-based distillation algorithm called TST. To be specific, TST locates augmented samples that the teacher is good at while the student is not, and transfers the knowledge of these augmented samples from the teacher to the student with the expectation that an excellent student can be learned. Our experimental results, including both qualitative and quantitative ones, demonstrate the feasibility of the data-based distillation method and the validity of the “Teach what you Should Teach” strategy. In future research, the data-based distillation, which is as competitive as logit-based and feature-based distillation, may deserve more attention in the field of knowledge distillation. Meanwhile, combining data-based distillation with logit-based distillation and feature-based distillation may lead to greater performance breakthroughs in distillation.

## Acknowledgments

This work was supported in part by the Nature Science Foundation of China (NSFC) under Grant No 62072041.

## References

- [Ahn *et al.*, 2019] Sungsoo Ahn, Shell Xu Hu, et al. Variational information distillation for knowledge transfer. In *Computer Vision and Pattern Recognition*, pages 9163–9171, Long Beach, CA, USA, Jun. 2019. IEEE.
- [Carlini and Wagner, 2017] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy*, pages 39–57. IEEE, 2017.



- [Chen *et al.*, 2019] Kai Chen, Jiaqi Wang, and Pang et al. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.
- [Chen *et al.*, 2021] Pengguang Chen, Shu Liu, et al. Distilling knowledge via knowledge review. In *Computer Vision and Pattern Recognition*, pages 5008–5017, Virtual Event, Jun. 2021. IEEE.
- [Chen *et al.*, 2022] Huanran Chen, Shitong Shao, Ziyi Wang, Zirui Shang, Jin Chen, Xiaofeng Ji, and Xinxiao Wu. Bootstrap generalization ability from loss landscape perspective. *arXiv preprint arXiv:2209.08473*, 2022.
- [Chen *et al.*, 2023] Huanran Chen, Shitong Shao, Ziyi Wang, Zirui Shang, Jin Chen, Xiaofeng Ji, and Xinxiao Wu. Bootstrap generalization ability from loss landscape perspective. In Leonid Karlinsky, Tomer Michaeli, and Ko Nishino, editors, *ECCV 2022 Workshops*, Cham, 2023. Springer.
- [Contributors, 2020] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark, 2020.
- [Contributors, 2021] MMRazor Contributors. Openmmlab model compression toolbox and benchmark, 2021.
- [Cordts *et al.*, 2016] Marius Cordts, Mohamed Omran, and Ramos et al. The cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition*, pages 3213–3223, Las Vegas, NV, USA, Jun.-Jul. 2016. IEEE.
- [Cubuk *et al.*, 2019] Ekin D Cubuk, Barret Zoph, et al. Autoaugment: Learning augmentation strategies from data. In *Computer Vision and Pattern Recognition*, pages 113–123, Long Beach, CA, USA, Jun. 2019. IEEE.
- [Cubuk *et al.*, 2020] Ekin D Cubuk, Barret Zoph, et al. Randaugment: Practical automated data augmentation with a reduced search space. In *Computer Vision and Pattern Recognition workshops*, pages 702–703, Seattle, WA, USA, Jun. 2020. IEEE.
- [Dai *et al.*, 2019] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, Jul. 2019. ACL.
- [Dai *et al.*, 2021] Xing Dai, Zeren Jiang, et al. General instance distillation for object detection. In *Computer Vision and Pattern Recognition*, pages 7842–7851, Virtual Event, Jun. 2021.
- [Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [Fang *et al.*, 2022] Gongfan Fang, Kanya Mo, et al. Up to 100x faster data-free knowledge distillation. In *Association for the Advancement of Artificial Intelligence*, number 6, pages 6597–6604, Virtual Event, Feb.-Mar. 2022.
- [Frankle and Carbin, 2019] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.
- [He *et al.*, 2016] Kaiming He, Xiangyu Zhang, et al. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, NV, USA, Jun. 2016. IEEE.
- [He *et al.*, 2022] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Computer Vision and Pattern Recognition*, pages 16000–16009, New Orleans, LA, USA, Jun. 2022. IEEE.
- [Heo *et al.*, 2019] Byeongho Heo, Jeessoo Kim, and Sangdoo et al. A comprehensive overhaul of feature distillation. In *International Conference on Computer Vision*, pages 1921–1930, Seoul, Korea (South), Oct.-Nov. 2019. IEEE.
- [Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.
- [Huang *et al.*, 2022] Tao Huang, Shan You, et al. Knowledge distillation from a stronger teacher. In *Neural Information Processing Systems*, New Orleans, LA, USA, Nov.-Dec. 2022. NIPS.
- [Jaderberg *et al.*, 2015] Max Jaderberg, Karen Simonyan, et al. Spatial transformer networks. In *Neural Information Processing Systems*, volume 28, Montréal, Canada, Dec. 2015. Curran Associates, Inc.
- [Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 2009.
- [Li *et al.*, 2020] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy M. Hospedales, Neil Martin Robertson, and Yongxing Yang. DADA: differentiable automatic data augmentation. In *European Conference on Computer Vision*, Virtual Event, Aug. 2020. Springer.
- [Lim *et al.*, 2019] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *Neural Information Processing Systems*, pages 6662–6672, Vancouver, Canada, Jan. 2019. NIPS.
- [Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, et al. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, Zurich, Switzerland, Sept. 2014. Springer.
- [Liu *et al.*, 2019] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Computer Vision and Pattern Recognition*, pages 2604–2613, Long Beach, CA, USA, Jun. 2019. IEEE.
- [Liu *et al.*, 2020] Yifan Liu, Changyong Shu, Jingdong Wang, and Chunhua Shen. Structured knowledge distillation for dense prediction. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [Liu *et al.*, 2021] Ze Liu, Yutong Lin, et al. Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision*, pages 10012–10022, Virtual Event, Mar. 2021. IEEE.
- [Madry *et al.*, 2018] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, Vancouver, Canada, Apr.-May 2018. OpenReview.net.
- [Marcel and Rodriguez, 2010] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *ACM international conference on Multimedia*, pages 1485–1488, Firenze, Italy, Oct. 2010. ACM.
- [Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.



- [Park *et al.*, 2019] Wonpyo Park, Dongju Kim, et al. Relational knowledge distillation. In *Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, Long Beach, CA, USA, Jun. 2019. IEEE.
- [Peng *et al.*, 2019] Baoyun Peng, Xiao Jin, et al. Correlation congruence for knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5007–5016, 2019.
- [Purwins *et al.*, 2019] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019.
- [Romero *et al.*, 2014] Adriana Romero, Nicolas Ballas, et al. Fit-nets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.
- [Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- [Shu *et al.*, 2021] Changyong Shu, Yifan Liu, et al. Channel-wise knowledge distillation for dense prediction. In *International Conference on Computer Vision*, pages 5311–5320, Montreal, Canada, Oct. 2021. IEEE.
- [Suzuki, 2022] Teppei Suzuki. Techaugment: Data augmentation optimization using teacher knowledge. In *Computer Vision and Pattern Recognition*, pages 10904–10914, New Orleans, LA, USA, Jun. 2022. IEEE.
- [Tian *et al.*, 2020] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, Addis Ababa, Ethiopia, Apr. 2020. OpenReview.net.
- [Tung and Mori, 2019] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *International Conference on Computer Vision*, pages 1365–1374, Seoul, Korea (South), Oct–Nov. 2019. IEEE.
- [Wang *et al.*, 2020] Yukang Wang, Wei Zhou, Tao Jiang, Xiang Bai, and Yongchao Xu. Intra-class feature variation distillation for semantic segmentation. In *European Conference on Computer Vision*, pages 346–362, Virtual Event, Aug. 2020. Springer.
- [Wieczorek *et al.*, 2021] Mikołaj Wieczorek, Barbara Rychalska, and Jacek Dabrowski. On the unreasonable effectiveness of centroids in image retrieval. In *International Conference on Neural Information Processing*, pages 212–223, Bali, Indonesia, Dec. 2021. Springer.
- [Wu *et al.*, 2016] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, Jun. 2016. IEEE.
- [Yang *et al.*, 2021] Jing Yang, Brais Martinez, et al. Knowledge distillation via softmax regression representation learning. In *International Conference on Learning Representations*. OpenReview.net, 2021.
- [Yang *et al.*, 2022] Zhendong Yang, Zhe Li, et al. Focal and global knowledge distillation for detectors. In *Computer Vision and Pattern Recognition*, pages 4643–4652, New Orleans, LA, USA, Jun. 2022. IEEE.
- [Yin *et al.*, 2020] Hongxu Yin, Pavlo Molchanov, et al. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Computer Vision and Pattern Recognition*, pages 8715–8724, Seattle, WA, USA, Jun. 2020. IEEE.
- [Zagoruyko and Komodakis, 2016a] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations (ICLR)*, 2016.
- [Zagoruyko and Komodakis, 2016b] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, pages 1–15, York, UK, Sept. 2016. BMVA.
- [Zhang and Ma, 2021] Linfeng Zhang and Kaisheng Ma. Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors. In *International Conference on Learning Representations*, Vienna, Austria, May 2021, pages 1–14. OpenReview.net, 2021.
- [Zhao *et al.*, 2017] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Computer Vision and Pattern Recognition*, Honolulu, HI, USA, Jul. 2017. IEEE.
- [Zhao *et al.*, 2020] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Computer Vision and Pattern Recognition*, pages 10076–10085, Seattle, WA, USA, Jun. 2020. IEEE.
- [Zhao *et al.*, 2022] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Conference on Computer Vision and Pattern Recognition*, pages 11953–11962, New Orleans, LA, USA, Jun. 2022. IEEE.
- [Zhixing *et al.*, 2021] Du Zhixing, Rui Zhang, Ming Chang, Shaoli Liu, Tianshi Chen, Yunji Chen, et al. Distilling object detectors with feature richness. *Neural Information Processing Systems*, 34:5213–5224, Dec. 2021.

## A Datasets

**CIFAR-100.** Dataset CIFAR-100 [Krizhevsky and Hinton, 2009] is the subsets of the tiny image dataset and consists of 60,000 images with the size  $32 \times 32$ . Specifically, the training set contains 50,000 images, and the testing set contains 10,000 images.

**ImageNet-1k.** Dataset ImageNet-1k [Russakovsky *et al.*, 2015], also commonly referred to as ILSVRC 2012, has 1000 classes, and the benchmark is trained using the training set and tested using the validation set. Its training and validation sets contain 1281,167 and 50,000 images, respectively.

**MS-COCO.** Dataset MS-COCO [Lin *et al.*, 2014] is a large-scale object detection dataset. The benchmark is the same as ImageNet-1k, using the training set for training and the validation set for testing. The training/validation split was changed from 83K/41K to 118K/5K in 2017. Researchers commonly apply the 2017 version for experiments.

**Cityscapes.** Dataset Cityscapes [Cordts *et al.*, 2016] is a new large-scale dataset for semantic segmentation. It provides 5,000 images that have been meticulously annotated, with 2,975 images for training and 500 images for validation, where 30 common classes are provided and 19 classes are used for evaluation and testing. Each image is  $2048 \times 1024$  in size.

## B Hyperparameter Settings

### B.1 Basic settings

**Classification.** For the classification experiments on CIFAR-100, the batch size is 128, the total number of epochs is 240, and the learning rate is initialized to 0.1 and divided by 10 at 150, 180 and 210 epochs. In addition, we employ an SGD optimizer for training and set the weight decay and momentum as  $5e-4$  and 0.9, respectively. For the classification experiments on ImageNet-1k (ResNet34-ResNet18 pair and ResNet50-MobileNet pair), the total batch size is 512, the total number of epochs is 100, the batch size in every GPU is 128, the number of GPUs is 4 and the learning rate is initialized to 0.1 and divided by 10 at 30, 60 and 90 epochs. Besides, we employ an SGD optimizer for training and set the weight decay and momentum as  $1e-4$  and 0.9, respectively. For the Swin-Large-Swin-Tiny pair, we follow the setting in [Liu *et al.*, 2021], except for the batch size and initial learning rate. Specifically, the batch size is 512, the total number of epochs is 300, the batch size in every GPU is 64, the number of GPUs is 8, and the learning rate is initialized to 0.0005 and is decayed by a cosine scheduler. Furthermore, we utilized an Adam optimizer for training and set the weight decay as  $5e-2$ . The reason for halving the batch size is that the GPU memory is insufficient to support the training with the original hyperparameter settings.

**Detection.** For the detection experiments on MS-COCO, we utilize mmdetection [Chen *et al.*, 2019] and mmrazor [Contributors, 2021] for both training and testing. Following [Shu *et al.*, 2021; Park *et al.*, 2019], we use the same standard training strategies on the Cascade RCNN-X101-Faster RCNN-R50 and RetinaNet-X101-RetinaNet-

R50 pairs. To be specific, the total batch size is 16, the total number of epochs is 24, the batch size in every GPU is 2, the number of GPUs is 8 and the learning rate is divided by 10 at 16 and 22 epochs. The initial learning rate is set as 0.02 and 0.01 on Cascade RCNN-X101-Faster RCNN-R50 and RetinaNet-X101-RetinaNet-R50 pairs, respectively. Besides, the setting on the FCOS-R101-FCOS-R50 pair is following [Yang *et al.*, 2022]. Compared with the RetinaNet-X101-RetinaNet-R50 pair, the only difference is we apply a warm-up learning rate on the FCOS-R101-FCOS-R50 pair.

**Segmentation.** For the segmentation experiments on Cityscapes, we apply mmsegmentation [Contributors, 2020] and mmrazor [Contributors, 2021] for distillation. we follow the setting in [Contributors, 2021]. In specific, the total batch size is 16, the total number of iterations is 80,000, the batch size in every GPU is 2, the number of GPUs is 8 and the learning rate is 0.01. We make the learning rate decay to 0.9 in each iteration and constrain the minimum learning rate in training to be  $1e-4$ . And we utilized a SGD optimizer for training and set the weight decay and momentum as  $5e-4$  and 0.9, respectively.

### B.2 Advanced Settings

**Classification.** On CIFAR-100, the loss weight is set as 1 for all comparison results and the temperature  $\tau$  of TST (i.e., vanilla KD) is 4. We let  $f_{DE}$  update the magnitudes  $\{\theta_m^i\}_{i=1}^{N-N_{nl}}$  and probabilities  $\{\theta_p^i\}_{i=1}^N$  (i.e., conduct Stage II) at 30 and 90 epochs and continuously train  $f_{DE}$  with ALRS [Chen *et al.*, 2022] for 1 epoch. Moreover, on ImageNet-1k, the loss weight is set as 1 for all comparison results and the temperature  $\tau$  of TST (i.e., vanilla KD) is 1. The magnitudes  $\{\theta_m^i\}_{i=1}^{N-N_{nl}}$  and probabilities  $\{\theta_p^i\}_{i=1}^N$  are updated at 10 and 20 epochs. Unlike CIFAR-100 training, we train 5 epochs for  $f_{DE}$  continuously.

**Detection.** On MS-COCO, the loss weight is set as 1 for all comparison results and the temperature  $\tau$  of TST (i.e., vanilla KD) is 1. We add  $\mathcal{L}_{KL}$  and  $\mathcal{L}_{MSE}$  to the final predictions of classes and the neck, respectively. On Stage II, we let not only the student classification predictions but also the student box regression be inaccurate. Compare with the student, we make the teacher’s categorical predictions and box regressions as accurate as possible. Furthermore, We make  $f_{DE}$  update the magnitudes  $\{\theta_m^i\}_{i=1}^{N-N_{nl}}$  and probabilities  $\{\theta_p^i\}_{i=1}^N$  at 6 and 12 epochs and continuously train  $f_{DE}$  for 1 epoch.

**Segmentation.** On Cityscapes, we set the loss weight on  $\mathcal{L}_{KL}$  between the teacher decode head’s final predictions and the student decode head’s final predictions as 1.5. In addition, we set the loss weight on  $\mathcal{L}_{KL}$  between the teacher auxiliary head’s final predictions and the student auxiliary head’s final predictions as 0.5. Also, the temperature  $\tau$  of TST (i.e., vanilla KD) is set as 1. Last but not least, the magnitudes  $\{\theta_m^i\}_{i=1}^{N-N_{nl}}$  and probabilities  $\{\theta_p^i\}_{i=1}^N$  is updated at 7,500 and 15,000 iterations on the total training. We continuously update  $\{\theta_m^i\}_{i=1}^{N-N_{nl}}$  and probabilities  $\{\theta_p^i\}_{i=1}^N$  with 1,250 iterations.

## C Training Process on TST

In this section, we show the full training process of TST in Algorithm 1. For the sake of simplicity, we do not categorize the different sub-policies here and consistently consider that they are all learnable. Moreover, for the object detection task, Eq. 1 in Algorithm1 can be replaced with other forms, i.e.,  $\mathcal{L}_{\text{mimic}} + \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{CE}}$ .

## D Supplementary Details

### D.1 Details about $f_{DE}$ in object detection tasks

Generally, a target object, after affine transformation and accumulation according to Eq. 5, generates multiple target objects, leading to ambiguity in the target detection task. Thus, we design a method to replace the weighted summation of the difference between the affine transform result and the original image. To be specific, we sum all affine transformation matrices generated from  $f_{\text{STN}}$  for obtaining only one new target affine transformation matrix  $\hat{A}$ . And  $f_{DE}$  simply learns  $\hat{A}$  that satisfies the objective that TST search. In here, we define two hyperparameters  $N_a$  and  $N_c$  (s.t.,  $N_a + N_c = N_A$ ) refer to the number of learnable affine transformations and color transformations that are actually selected, respectively. Due to all unlearnable transformations are color transformations, we can disregard them. First, for a set of affine meta-encoders  $\{f_{\text{STN}}^{Z^i}\}_{i=1}^{N_a}$  (s.t.,  $N_a \leq N_A$ )<sup>2</sup>, we can define the set of affine transformation matrices generated by this set as  $\{A^{Z^i}\}_{i=1}^{N_a}$ . Then, the final matrix  $\hat{A}$  generated by  $f_{DE}$  can be formulated as

$$\begin{aligned} A_E^i &= \theta_p^{Z^i} \odot A^{Z^i} - \theta_p^{Z^i} \odot I, \text{ s.t. } i \leq N_a \\ \hat{A} &= I + \sum_{i=1}^{N_a} A_E^i, \end{aligned} \quad (6)$$

where  $I$  refers to a matrix  $\sim \mathbb{R}^{2 \times 3}$  where only the values on the diagonal are 1 and the rest are 0. After that, we derive  $\hat{x}_a$  by the operator Affine.

$$\hat{x}_a = \text{Affine}(\hat{A}, x) \quad (7)$$

Now, we need to apply the remaining meta-encoders to complete the transform of  $\hat{x}_a$ . As a result, the corresponding computational form is shown in Eq. 8.

$$\begin{aligned} x_E^i &= \begin{cases} \theta_p^{Z^i} \odot f_E(\hat{x}_a, \theta_m^{Z^i}) - \theta_p^{Z^i} \odot \hat{x}_a, & N_a < i \leq N_A \text{ and } Z^i \leq N_{\text{nl}} \\ \theta_p^{Z^i} \odot f_E(\hat{x}_a) - \theta_p^{Z^i} \odot \hat{x}_a, & N_a < i \leq N_A \text{ and } Z^i > N_{\text{nl}} \end{cases} \\ \hat{x} &= \hat{x}_a + \sum_{i=N_a+1}^{N_A} x_E^i, \end{aligned} \quad (8)$$

Finally, we obtain  $\hat{x}$ , which is a part of the input to  $f_T$  and  $f_S$ . In particular,  $\hat{x}$  in Eq. 5 and Eq. 8 have the same meaning.

<sup>2</sup>Here, for simplicity, we let the first  $N_a$  of  $N_A$  actual applied meta-encoders all be  $f_{\text{STN}}$ .

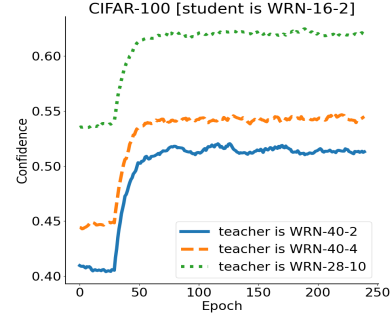


Figure 3: The probability of different-size teachers predicting samples correctly with epochs. Here, this figure is intended to supplement the discussion section.

### D.2 Full probability and magnitude visualization

Due to space constraints in the main paper, we only show 8 sub-policies' magnitudes and probabilities searched by TST. So, we illustrate the magnitudes and probabilities of all sub-policies in Fig. 4 (i.e., 11 magnitudes and 14 probabilities).

### D.3 Proof

Here, we will prove  $-\mathcal{L}_{\text{CE}}(x, y)$  is a non-convex function and  $\mathcal{L}_{\text{CE}}(1-x, y)$  is a convex function, where  $x$  is the logits output from the normalized activation function, i.e., softmax. And  $y$  is the ground truth label. Besides, the lengths of vectors  $x$  and  $y$  are the number of categories. Since  $y$  is one-hot encoding,  $-\mathcal{L}_{\text{CE}}(x, y)$  can be written as  $\log(x_t)$ , where  $x_t$  is the predicted logit in  $x$  corresponding to the target label. In particular,  $0 \leq x_t \leq 1$  due to the presence of softmax. As the definition of a convex function, we need to prove the following equation (note that deep learning is stochastic gradient descent for parameter updating, so the convex function declared here is downward convex):

$$\log(a+b) \leq \log(a) + \log(b), \quad \text{s.t. } 0 \leq a \leq b \leq 1. \quad (9)$$

Obviously,  $a+b \geq 2\sqrt{ab} \geq ab$ , so Eq. 9 does not hold and  $-\mathcal{L}_{\text{CE}}(x, y)$  is a non-convex function. Similarly, for  $\mathcal{L}_{\text{CE}}(1-x, y)$ , we can infer that it is a convex function by Eq. 10 since  $ab \leq 1$  is a known condition.

$$\begin{aligned} -\log(2-a-b) &\leq -\log(1-a) - \log(1-b), \\ \text{s.t. } 0 &\leq a \leq b \leq 1, \\ \Rightarrow \frac{1}{2-a-b} &\leq \frac{1}{(1-a)(1-b)}, \\ \Rightarrow ab &\leq 1, \end{aligned} \quad (10)$$

Table 9: Top-1 test accuracy (%) comparison of the few-shot scenario on CIFAR-100. Here, all teacher-student pairs are ResNet56-ResNet20 pairs.

Percentage	KD	CRD	Ours
25%	65.15	65.80	<b>66.28</b>
50%	68.61	69.91	<b>70.22</b>
75%	70.34	70.68	<b>71.80</b>

## D.4 Combine DIST and TST

TST is a novel data-based distillation algorithm that can be combined with other arbitrary feature-based and logit-based distillation algorithms and further enhance students' generalization ability. For instance, we combine DIST and TST on ResNet50-MobileNet-V1 on ImageNet-1k with 33 hours, which boosts the results from 73.24% to 73.55%.

## D.5 Generative Adversarial Network and Learnable Data Augmentation

We started by encoding the original samples based on Generative Adversarial Networks, but without some regularization constraints directly causing training collapse, and even with some constraints, the performance (75.60%) is still not as good as the current system, i.e., learnable data augmentation (76.75%) on WRN-40-2-WRN-16-2 pair on CIFAR-100.

## D.6 Supplementary Materials

Due to space constraints in the text, we present here the comparison experiments in the few-shot scenario (Table 9) and an iterative plot of the teacher's output of the logit's confidence as a function of epoch (Table 3).

---

### Algorithm 1 Training procedure of TST

---

**Input:** A student  $f_S$  with the parameter  $\theta_S$ , a teacher model  $f_T$  with the parameter  $\theta_T$ , a data augmentation encoder set  $\{f_E^i\}_{i=1}^N$  with the parameter set  $\{\theta_E^i\}_{i=1}^N$ , a set of the priori manual data-augmentation mappings  $\{f_A^i\}_{i=1}^N$ , the parameter of probabilities  $\theta_p$ , the parameter of magnitudes  $\theta_m$ , dataset  $\mathcal{X}$ , the probabilistic transformation function of the ground truth label  $p$ , the number of iterations required for the priori manual data augmentation fitting  $n_{\text{fitting}}$ , the number of iterations required for a encoder training phase  $n_{\text{encoder}}$ , the number of iterations required for a student training phase  $n_{\text{student}}$ , the iterative position set  $\mathbb{I}$  for probability and magnitude learning, the learning rate  $\eta_{\theta_S}$  and  $\eta_{\theta_E}$ .

- 1: Random initialization parameter  $\{\theta_E^i\}_{i=1}^N$ .
- 2: **for**  $i = 1, \dots, N$  **do**
- 3:   **for**  $j = 0, \dots, n_{\text{fitting}}$  **do**
- 4:     Randomly sample a mini-batch,  $\{x^k\}_{k=1}^B \sim \mathcal{X}$ .
- 5:     Randomly sample the magnitude  $m$ .
- 6:      $\{\mathbf{y}_E^{i,k}\}_{k=1}^B = \{f_E^i(x^k, m)\}_{k=1}^B$ .
- 7:      $\{\mathbf{y}_A^{i,k}\}_{k=1}^B = \{f_A^i(x^k, m)\}_{k=1}^B$ .
- 8:     Compute loss for the data augmentation encoder,  
 $\mathcal{L}_{\text{encoder}}^i = \sum_{k=1}^B \|\mathbf{y}_E^{i,k} - \mathbf{y}_A^{i,k}\|_2$ .
- 9:     Update  $\theta_E^i$  by the gradient ascent,  
 $\theta_E^i \leftarrow \theta_E^i - \eta_{\theta_E} \partial \mathcal{L}_{\text{encoder}}^i / \partial \theta_E^i$ .
- 10:   **end for**
- 11: **end for**
- 12: Random initialization parameter the parameter of probabilities  $\theta_p$  and the parameter of magnitudes  $\theta_m$ .
- 13: **for**  $i = 0, \dots, n_{\text{student}}$  **do**
- 14:   **if**  $i \in \mathbb{I}$  **then**
- 15:     **for**  $j = 0, \dots, n_{\text{encoder}}$  **do**
- 16:       Randomly sample a mini-batch,  $\{x^k\}_{k=1}^B \sim \mathcal{X}$ .
- 17:       Calculate new samples  $\{\hat{x}^k\}_{k=1}^B$  using Eq. 5.
- 18:       Compute loss for probabilities and magnitudes,  
 $\mathcal{L}_{\text{TST}} = \sum_{k=1}^B (-\log(p(f_S(\hat{x}^k))) - \log(1 - p(f_T(\hat{x}^k))))$
- 19:       Update  $\phi$  by the gradient ascent,  $\theta_p, \theta_m \leftarrow \theta_p - \eta_{\phi} \partial \mathcal{L}_{\text{TST}} / \partial \theta_p, \theta_m - \eta_{\phi} \partial \mathcal{L}_{\text{TST}} / \partial \theta_m$
- 20:     **end for**
- 21:   **end if**
- 22:   Randomly sample a mini-batch,  $\{x^k\}_{k=1}^B \sim \mathcal{X}$
- 23:   Calculate new samples  $\{\hat{x}^k\}_{k=1}^{2B}$  by Eq. 5 and the concatenation operator.
- 24:   Compute loss  $\mathcal{L}_S$  for the student by Eq. 1.
- 25:   Update  $\theta_S$  by the gradient descent,  
 $\theta_S \leftarrow \theta_S - \eta_{\theta_S} \partial \mathcal{L}_S / \partial \theta_S$
- 26: **end for**

---

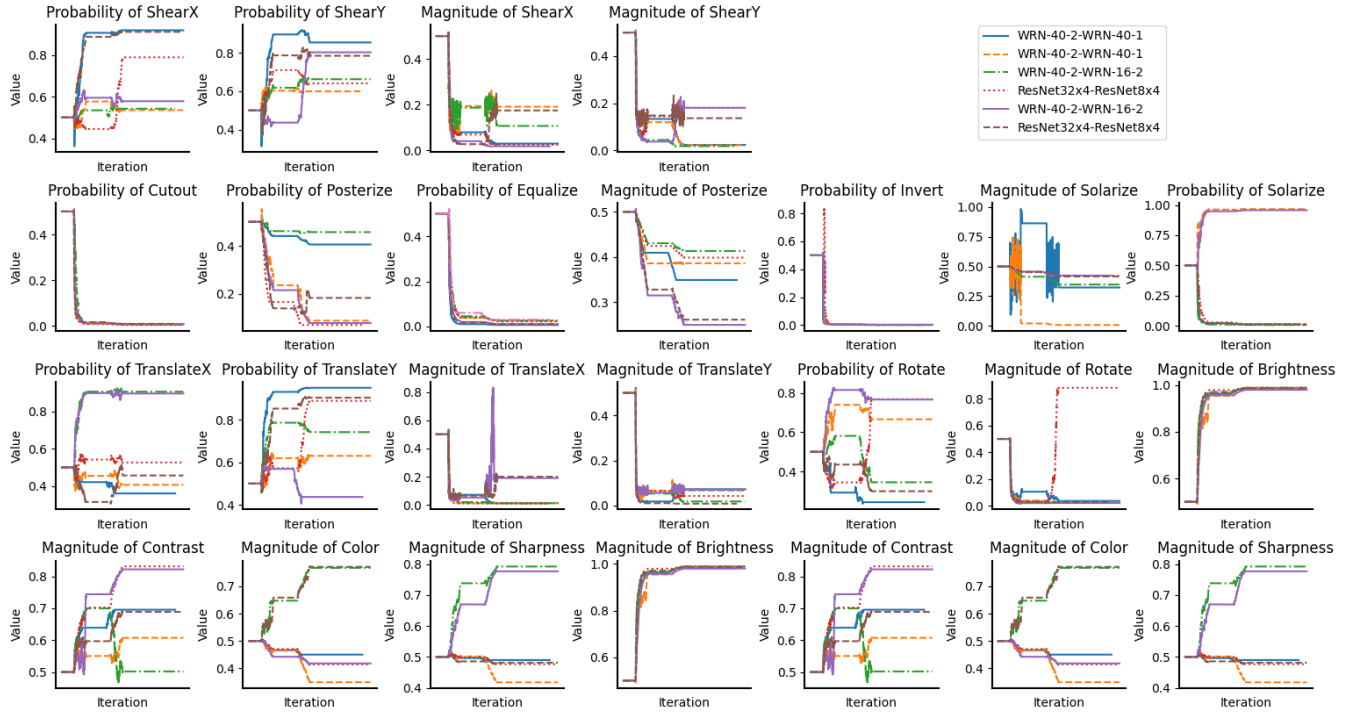


Figure 4: The plot of magnitudes and probabilities variation with iteration under TST on CIFAR-100. Here we visualize all sub-policies mentioned in this paper.