Understanding and Improving Knowledge Distillation

Jiaxi Tang [1 *] **Rakesh Shivanna** [2] **Zhe Zhao** [2] **Dong Lin** [2] **Anima Singh** [2] **Ed H. Chi** [2] **Sagar Jain** [2]

Abstract

Knowledge distillation is a model-agnostic technique to improve model quality while having a fixed capacity budget. It is a commonly used technique for model compression, where a higher capacity teacher model with better quality is used to train a more compact student model with better inference efficiency. Through distillation, one hopes to benefit from student's compactness, without sacrificing too much on model quality. Despite the large success of knowledge distillation, better understanding of how it benefits student model's training dynamics remains under-explored. In this paper, we dissect the effects of knowledge distillation into three main factors: (1) benefits inherited from label smoothing, (2) example re-weighting based on teacher's confidence on ground-truth, and (3) prior knowledge of optimal output (logit) layer geometry. Using extensive systematic analyses and empirical studies on synthetic and real-world datasets, we confirm that the aforementioned three factors play a major role in knowledge distillation. Furthermore, based on our findings, we propose a simple, yet effective technique to improve knowledge distillation empirically.

1. Introduction

Recent advances in artificial intelligence have largely been driven by deep neural networks using multi-layer perceptron, and thus, current state-of-the-art models typically require a high inference cost in computation and memory. Therefore, several works have been devoted to find a better quality and computation trade-off, such as pruning (Han et al., 2015b;a) and quantization-based approaches (Han et al., 2015a; Jacob et al., 2018). One promising and commonly used method for addressing this computational burden is called Knowledge Distillation (KD), proposed by Hinton et al. (2015), which uses a larger capacity teacher model (ensembles) to transfer its 'dark knowledge' to a more compact student model. Through distillation, one hopes to achieve a student model that not only inherits better quality from the teacher, but is also more efficient for inference due to its compactness. Recently, we have witnessed a huge success of knowledge distillation, irrespective of the model architecture and application domain (Kim & Rush, 2016; Chen et al., 2017; Tang & Wang, 2018; Anil et al., 2018; He et al., 2019).

Despite the large success of knowledge distillation, surprisingly sparse theoretical research has been done to better understand the mechanism of how it works, which could limit the applications of KD; and also raise unexpected or unexplainable results. For example, to successfully 'distill' a better student, one common practice is to have a teacher model with as good quality as possible. However, recently Mirzadeh et al. (2019) and Müller et al. (2019) have found this intuition would fail under certain circumstances. Furthermore, Anil et al. (2018) and Furlanello et al. (2018) have analyzed that even without using a powerful teacher, distilling a student model to itself using mutual or self-distillation also improves quality. To this end, some researchers have made attempts on understanding the mechanism of KD. For example, Yuan et al. (2019) connects label smoothing to KD. Furlanello et al. (2018) conjectures KD's effect on re-weighting training examples. In this work, we found the benefits of KD comes from a combination of multiple effects, and propose methods to dissect each effect.

This work is an attempt to shed light upon the 'dark knowledge' distillation, making this technique less mysterious. More specifically, we make the following contributions:

- We systematically break down the effects of Knowledge Distillation into: (1) label smoothing, (2) example re-weighting, and (3) prior knowledge of optimal output (logit) layer geometry. We provide theoretical analyses on how KD exhibits these effects, and helps improve student model's quality (Section 3).
- We propose several partial-distillation techniques using hand-crafted teacher's output distribution (Section 4) to simulate different effects of knowledge distillation.

[*]Work done while interning at Google. [1]School of Computing Science, Simon Fraser University, Canada. [2]Google Inc., Mountain View, USA. Correspondence to: Jiaxi Tang <jiaxit@sfu.ca>, Rakesh Shivanna <rakeshshivanna@google.com>, Zhe Zhao <zhezhao@google.com>, Dong Lin <dongl@google.com>, Anima Singh <animasingh@google.com>, Ed H. Chi <edchi@google.com>, Sagar Jain <sagarj@google.com>.

Preliminary work.

- Inspired by our analysis and empirical findings, we propose a simple, yet effective novel distillation technique to improve over vanilla KD method (Section 5).

- We empirically demonstrate and confirm our hypothesis on the effects of KD on both synthetic and real-world datasets. Also, experiments on image classification and language modeling verifies the effectiveness of our proposed technique to improve over KD.

2. Related Work

In the context of deep learning, knowledge transfer has been successfully used to effectively compress the power of a larger capacity model (a teacher) to a smaller neural network (a student). Adopting this teacher-student learning paradigm, many forms of *knowledge* have been investigated: layer activations (Romero et al., 2014), auxiliary information (Vapnik & Izmailov, 2015), Jacobian matrix of the model parameters (Czarnecki et al., 2017; Srinivas & Fleuret, 2018), Gram matrix derived from pairs of layers (Yim et al., 2017), activation boundary (Heo et al., 2019), etc. Among these, Knowledge Distillation (KD) – learning from teacher's output distribution (Hinton et al., 2015) is the most popular. Besides compression, KD has also been successfully applied to improve generalization (Furlanello et al., 2018), model reproducibility (Anil et al., 2018), defend adversarial attacks (Papernot et al., 2016), etc.

Though knowledge distillation has been successfully applied in various domains, there has been very few attempts on understanding how and why it helps neural network training. Hinton et al. (2015) argued that the success of KD could be attributed to the output distribution of the incorrect classes, which provides information on class relationships. From learning theory perspective, Vapnik & Izmailov (2015) studied the effectiveness of knowledge transfer using auxiliary information, known as *Privileged Information*. Following which, Lopez-Paz et al. (2015) established the connection between KD and privileged information. Recently, Phuong & Lampert (2019) showed a faster convergence rate from distillation. However, most of the existing theoretical results rely on strong assumptions (e.g., linear model, or discard ground-truth when training the student), and also fails to explain the recent failure cases of distilling from a better quality teacher (Mirzadeh et al., 2019; Müller et al., 2019).

The most relevant work to our own is (Furlanello et al., 2018). Though the main focus of their work is to propose KD techniques to boost quality, they also provide intuitions for the effectiveness of KD. In our work, we offer theoretical analysis on some of their conjectures, and improve on erroneous assumptions. Furthermore, we systematically investigate the mechanism behind knowledge distillation by decomposing its effects, and analyzing how each of these effects helps with student model's training and quality improvement using our proposed partial-KD methods.

3. Analyzing Mechanisms of Knowledge Distillation

In this section, we provide a systematic analyses for the mechanisms behind KD based on theoretical and empirical results. We start by introducing essential background, dissect distillation benefits from three main effects, and conclude by connecting and summarizing these effects.

3.1. Background

Consider the task of image classification over a set of classes $[K] := \{1, 2, \dots, K\}$, we have tuples of images and labels: $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$, with $\boldsymbol{y} \in \{0, 1\}^K$ to denote one-hot encoded label, and $t \in [K]$ to denote the ground-truth class. The goal is to learn a parametric mapping function $f(\boldsymbol{x}; \theta) : \mathcal{X} \mapsto \mathcal{Y}$ where $\theta \in \Theta$ is characterized by a neural network. We learn the parameters θ via Empirical Risk Minimization of the surrogate loss function, typically optimized using some variants of Stochastic Gradient Descent:

$$\theta^* = \arg\min_{\theta \in \Theta} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}; \theta)), \tag{1}$$

where $\mathcal{L}$ is the cross-entropy loss $\mathcal{H}(\boldsymbol{y}, \boldsymbol{q}) = \sum_{i=1}^{K} -y_i \log q_i$ between one-hot encoded label $\boldsymbol{y} \in \mathcal{Y}$, and network output distribution $\boldsymbol{q} = f(\boldsymbol{x}; \theta)$ computed by applying $\mathrm{softmax}$ function over the logits output $\boldsymbol{z}$:

$$q_i = \mathrm{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}. \tag{2}$$

We could also apply temperature T on the $\mathrm{softmax}$ to get a more smooth output distribution $\tilde{q}_i = \mathrm{softmax}(z_i / T)$.

Gradient of a single-sample *w.r.t.* logit z_i is given by:

$$\frac{\partial \mathcal{L}}{\partial z_i} = q_i - y_i \ \big(= \partial_i \big). \tag{3}$$

3.2. Benefit from Label Smoothing

Label Smoothing (Szegedy et al., 2016) is a technique to soften one-hot encoded label $\boldsymbol{y}$ by a factor of ϵ, such that the modified label becomes: $\tilde{y}_i^{\mathrm{LS}} = (1 - \epsilon) y_i + \epsilon / K$. Label smoothing mitigates the over-confidence issue of neural networks, and improves model calibration (Müller et al., 2019). Knowledge Distillation (KD) on the other hand, uses an additional teacher model's predictions $\boldsymbol{p}$:

$$\begin{aligned} \theta^* &= \arg\min_{\theta \in \Theta} \mathcal{L}^{\mathrm{KD}}(\boldsymbol{y}, \boldsymbol{p}, f(\boldsymbol{x}; \theta), \lambda, T) \\ &= (1 - \lambda) \mathcal{H}(\boldsymbol{y}, \boldsymbol{q}) + \lambda \mathcal{H}(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{q}}), \end{aligned} \tag{4}$$

where $\lambda \in [0, 1]$ is a hyper-parameter; $\tilde{\boldsymbol{q}}$ and $\tilde{\boldsymbol{p}}$ are softened student and teacher's predictions by applying temperature T in Equation (2). Logits gradient for KD is given by:

$$\frac{\partial \mathcal{L}^{\mathrm{KD}}}{\partial z_i} = (1-\lambda)(q_i - y_i) + \frac{\lambda}{T}(\tilde{q}_i - \tilde{p}_i) \ \big(= \partial_i^{\mathrm{KD}} \big). \quad (5)$$

Yuan et al. (2019) established the connection between KD and label smoothing – In terms of gradient propagation, Knowledge Distillation is equivalent to Label Smoothing, when temperature $T = 1$, and teacher's probability distribution $\boldsymbol{p}$ follows a uniform distribution. In other words, we can view KD as an adaptive version of label smoothing, suggesting it should inherit most of the benefits from label smoothing, such as model regularization and better calibration, not being over-confident (Müller et al., 2019).

In the next two subsections, we analyze the unique characteristics of real teacher's output distribution $\boldsymbol{p}$ over the uniform distribution, and demonstrate how they could potentially facilitate student model's training with distillation.

3.3. Benefit from Teacher's Prediction Confidence

One important characteristic of the teacher distribution $\boldsymbol{p}$ is that the prediction (confidence) p_t on the ground-truth class t is not a constant and varies across examples. Compared to vanilla loss function in Equation (1), we find that KD performs gradient rescaling in the logits space. Following is the ratio of gradients (eqs. (3) and (5)) from the two losses:

$$\omega_i = \partial_i^{\mathrm{KD}} / \partial_i = (1-\lambda) + \frac{\lambda}{T}\left(\frac{\tilde{q}_i - \tilde{p}_i}{q_i - y_i}\right). \quad (6)$$

Next, we show that KD performs example re-weighting based on teacher model's prediction confidence p_t on the ground-truth class. The gradient rescaling factor ω_i is larger on average, when teacher is more confident on making the right prediction. More specifically, we state the following:

Proposition 1 (Example Re-weighting). *Given any example $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$, let $\tilde{p}_t = \tilde{q}_t + \tilde{c}_t + \eta$, where $\tilde{c}_t > 0$ is teacher's relative prediction confidence on the ground-truth $t \in [K]$ and η is a zero-mean random noise. Then the gradient rescaling factor for all classes by applying Knowledge Distillation is given by:*

$$\mathbb{E}_\eta \left[\sum_{i \in [K]} |\partial_i^{KD}| \Big/ \sum_{i \in [K]} |\partial_i| \right] = (1-\lambda) + \frac{\lambda}{T}\left(\frac{\tilde{c}_t}{1-q_t}\right).$$

Proof. We first consider the ground-truth class $t \in [K]$. Using $\tilde{p}_t = \tilde{q}_t + \tilde{c}_t + \eta$ and $\mathbb{E}[\eta] = 0$ in equation 6, we get:

$$\mathbb{E}_\eta[|\omega_t|] = (1-\lambda) + \frac{\lambda}{T}\left(\frac{\tilde{c}_t}{1-q_t}\right)$$

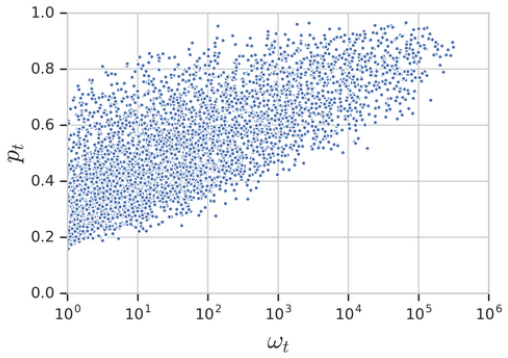


Figure 1: When applying KD for ResNet-20 student model with ResNet-56 as the teacher on CIFAR-100, we plot $\tilde{p}_t$ vs. ω_t (in log scale) with 10K samples at the end of training. Clearly, we can see that the two are positively correlated.

Now, sum of the incorrect class gradients is given by[1]:

$$\sum_{i \neq t} |\partial_i^{\mathrm{KD}}| = \sum_{i \neq t} \left[(1-\lambda) q_i + \frac{\lambda}{T}(\tilde{q}_i - \tilde{p}_i) \right]$$
$$= (1-\lambda)(1-q_t) + \frac{\lambda}{T}(\tilde{p}_t - \tilde{q}_t) = |\partial_t^{\mathrm{KD}}|$$

Penultimate equality follows from $\boldsymbol{q}$, $\boldsymbol{p}$ and $\tilde{\boldsymbol{q}}$ being probability masses. Similarly applies for ∂_i, and hence the proof.

□

At a given snapshot during training, we could assume $\tilde{c}_t$ to be a constant for all examples. Then for any pairs of examples $(\boldsymbol{x}, \boldsymbol{y})$ and $(\boldsymbol{x}', \boldsymbol{y}') \in \mathcal{X} \times \mathcal{Y}$, if the teacher is more confident on one of them, i.e., $p > p'$, then the average ω for all classes will be greater than ω' according to Proposition 1.

In Figure 1, we plot the relationship between ω_t and p_t at the end of training ResNet-20 (He et al., 2016) student model with ResNet-56 as the teacher on CIFAR-100 (Krizhevsky et al., 2009) dataset (more details of this dataset can be found in Suppl. Section A). The plot shows a clear positive correlation between the two. Also, we found the correlation to be stronger at the beginning of training.

In (Furlanello et al., 2018), the authors conjecture that example weight is associated with the largest value in $\boldsymbol{p}$. However, in the above proof, we show that once the teacher makes a wrong prediction, using the largest value gives a contradictory result. It is also trivial to prove that, when only having two classes, that is $\omega_{i \neq t} = \omega_t$, the only effect of using KD is example re-weighting. So we can regard the use of KD on

[1] Under the assumption of having a better quality teacher model, we assume $p_t > q_t$, and correspondingly $q_i \geq p_i, \ \forall i \in [K] \backslash t$.

binary classification (Anil et al., 2018) as taking the binary log-loss, and multiply with the weight ω_t.

In summary, we think incorporating teacher's supervision in knowledge distillation has an effect of example re-weighting, and the weight is associated with teacher's prediction confidence p_t on the ground-truth. Weight will be higher when p_t is larger. Alternatively, this suggests that KD would assign larger weights to training examples that are considered easier from teacher's perspective, and vice versa; which has a similar flavor to Curriculum Learning. Bengio et al. (2009) suggests this may not only speedup training convergence, but also helps to reach a better local minima. Also, according to (Roux, 2016), re-weighting examples during training with model prediction confidence results in a tighter bound on the classification error, leading to better generalization.

3.4. Prior of Optimal Geometry in Logit Layer

For binary classification, we showed in Section 3.3 that KD is essentially performing example re-weighting. However, for multi-class classification, we argue that KD also leverages relationship between classes as captured by teacher's probability mass distribution $\boldsymbol{p}$ over the incorrect classes. As argued by Hinton et al. (2015), on MNIST dataset, model assigns relatively high probability for class '7', when the ground-truth class is '2'. In this section, we first confirm their hypothesis using empirical studies. Then, we provide new insights to interpret the class relationship as a prior on optimal geometry in student's last logit layer.

To illustrate that the teacher's distribution $\boldsymbol{p}$ captures class relationships, we train ResNet-56 on CIFAR-100 dataset. CIFAR-100 contains 100 classes over 20 super-classes, with each super-class containing 5 sub-classes. Figures 2a and 2b show the heatmap for Pearson correlation coefficient on teacher's distribution $\boldsymbol{p}$ for different temperatures. We sort the class indexes to ensure that the 5 classes from the same super-class appear next to each other. We observe in Figure 2a that with lower temperature, there's no pattern on the heatmap showing class relationships. But as we increase the temperature in Figure 2b, classes within the same super-class clearly have high correlations to each other, as seen in the block diagonal structure. This observation verifies that teacher's distribution $\boldsymbol{p}$ indeed reveals class relationships, with proper tuning on the softmax temperature.

Before diving into the details of geometric interpretation, we recall the case of label smoothing (Szegedy et al., 2016).

- From an optimization point of view, He et al. (2019) showed that there is an optimal constant margin $\log(K(1-\epsilon)/\epsilon + 1)$, between the logit of the ground-truth z_t, and all other logits z_{-t}, using a label smoothing factor of ϵ. For fixed number of classes K, the margin is a monotonically decreasing function of ϵ.
- From geometry perspective, Müller et al. (2019) showed the logit $z_k = \boldsymbol{h}^\top \boldsymbol{w}_k$ for any class k is a measure of squared Euclidean distance $\|\boldsymbol{h} - \boldsymbol{w}_k\|^2$ between the activations of the penultimate layer $\boldsymbol{h}$[2], and weights $\boldsymbol{w}_k$ for class k in the last logit layer.

The above results suggests that label smoothing encourages $\|\boldsymbol{h}-\boldsymbol{w}_t\|^2 \geq \|\boldsymbol{h}-\boldsymbol{w}_{-t}\|^2$, and pushes all the other incorrect classes equally apart. Following a similar proof technique, we extend the above results to knowledge distillation:

Proposition 2. *Given $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$, at the optimal solution of the student for the final layer logits $\boldsymbol{w}_k^*$, $\forall k \in [K]$ at $T = 1$, Knowledge Distillation enforces different inter-class distances based on teacher's probability distribution $\boldsymbol{p}$:*

$$\|\boldsymbol{h} - \boldsymbol{w}_i^*\|^2 < \|\boldsymbol{h} - \boldsymbol{w}_j^*\|^2 \quad \textit{iff} \quad p_i > p_j,\ \forall i, j \in [K] \backslash t$$

Proof. At the optimal solution of the student, equating gradient in Equation (5) to 0, for $T = 1$, we get:

$$q_k^* = \begin{cases} \lambda p_k + (1 - \lambda) & \text{if } k = t, \\ \lambda p_k & \text{otherwise.} \end{cases} \tag{7}$$

Using a similar proof technique as Müller et al. (2019), $\|\boldsymbol{h} - \boldsymbol{w}_k^*\|^2 = \|\boldsymbol{h}\|^2 + \|\boldsymbol{w}_k^*\|^2 - 2\boldsymbol{h}^\top \boldsymbol{w}_k^*$, where $\boldsymbol{h}$ is the penultimate layer activations, and $\boldsymbol{w}_k^*$ are the weights of the last logits layer for class $k \in [K]$. Note that $\|\boldsymbol{h}\|^2$ is factored out when computing the $\mathrm{softmax}$, and $\|\boldsymbol{w}_k^*\|^2$ is usually a (regularized) constant across all classes.

Equating $z_k^* = \boldsymbol{h}^\top \boldsymbol{w}_k^*$, and using the property $\mathrm{softmax}(\boldsymbol{z}) = \mathrm{softmax}(\boldsymbol{z} + c),\ \forall c \in \mathbb{R}$, we get:

$$\begin{aligned} q_k^* &= \mathrm{softmax}(z_k^*) = \mathrm{softmax}(\boldsymbol{h}^\top \boldsymbol{w}_k^*) \\ &= \mathrm{softmax}\Big(-\frac{1}{2}\|\boldsymbol{h} - \boldsymbol{w}_k^*\|^2 \Big) = \lambda p_k,\ \forall k \in [K] \backslash t \end{aligned}$$

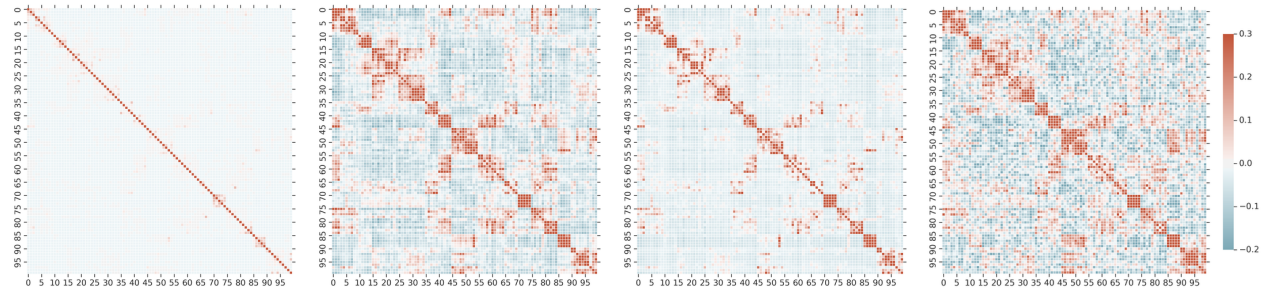
The last equality follows from equation 7, and thus proves the claim on class partition prior geometry at optimality. □

From Figure 2b, we know that the teacher assigns higher probability to the classes within the same super-class, and hence KD encourages hierarchical clustering of the logits layer weights based on the class relationships.

3.5. Summarizing Effects of Knowledge Distillation

Primarily, KD brings a regularization/calibration effect by introducing smoothed teacher distribution, although this effect is not considered as knowledge. Besides, we believe there are two types of knowledge teacher model will distill to its student – real teacher's probability distribution $\boldsymbol{p}$ not only benefits the student via *confidence on ground-truth*

[2]Here $\boldsymbol{h}$ can be concatenated with a "1" to account for the bias.



(a) (b) (c) (d)

Figure 2: Using 10K samples from CIFAR-100 for ResNet-56, we plot Pearson correlations of output probability $\boldsymbol{p}$ with varying $\mathrm{softmax}$ temperature (a) $T = 3$, (b) $T = 10$, and (c) $T = 10$ where only top-10 largest values in $\boldsymbol{p}$ are preserved. In (d), we show cosine similarities computed from the weights of the final logits layer. Since every 5 classes are within the same super-class, class correlations can be interpreted as the color 'squares' on the block-diagonal.

class to re-weight examples, but also from its probability mass on the *incorrect classes*. Intuitively, these values reflect class relationships, therefore provide the student with more guidance. We further interpret these values as applying different label smoothing factor ϵ for the incorrect classes. The difference in ϵ encourages student's optimal inter-class distance to be different for different classes. In other words, the distillation loss will get lower if more desired output logit layer's geometric inequalities hold. As a result, two sources of knowledge complementary to each other, which could potentially facilitate the student model's training process and further improve model generalization.

4. Isolating Effects by Partial Knowledge Distillation Methods

To further dissect the different effects of knowledge distillation, in this section, we synthesize hand-crafted teacher distributions, denoted by $\boldsymbol{\rho}$. Each synthetic teacher distribution $\boldsymbol{\rho}$ contains partial information from the real teacher's distribution $\boldsymbol{p}$, enabling us to isolate the effects of KD (namely, example re-weighting and optimal prior geometry through class relationship). We propose KD-pt and KD-sim – former only incorporates the example re-weighting effect and excludes class relationship information, and the latter only incorporates class relationships but not example re-weighting. We then try to combine the two effects together, to approximate the performance of vanilla KD.

4.1. Proposed Partial KD Methods

Examine Example Re-weighting Effect by KD-pt. Label smoothing does not have either re-weighting or the class relationships effect, due to its uniform teacher distribution. However, if we borrow p_t (prediction on ground truth class $t \in [K]$) from the real teacher's probability distribution $\boldsymbol{p}$, we can synthesize partial-teacher distribution that is able to incorporate example re-weighting effect. More specifically, we craft teacher probability distribution as follows:

$$\rho_i^{\text{pt}} = \begin{cases} p_t & \text{if } i = t, \\ (1 - p_t)/(K - 1) & \text{otherwise.} \end{cases} \tag{8}$$

From Proposition 1, it is trivial to see that KD-pt is capable of differentiating weights for different examples. However, it does not capture class relationships, since every class that is not the ground truth has the same probability mass.

Examine Optimal Prior Geometry Effect by KD-sim. Following the same methodology, we synthesize a teacher distribution that only capture class relationships, and assign the same weight to each example. To achieve this, we use the weights of the last logit layer $\boldsymbol{W} \in \mathbb{R}^{K \times d}$ from the teacher model to obtain class relationships. We believe the teacher, due to its larger capacity is able to encode class semantics in the weights of the last logit layer. Thus, we create a distribution $\boldsymbol{\rho}^{\text{sim}}$ as the $\mathrm{softmax}$ over cosine similarity[3] of the weights: $\boldsymbol{\rho}^{\text{sim}} = \mathrm{softmax}(\hat{\boldsymbol{w}}_t \hat{\boldsymbol{W}}^\top)$, where $\hat{\boldsymbol{W}}$ is the l_2-normalized logit layer weights, and $\hat{\boldsymbol{w}}_t$ is the t-th row of $\hat{\boldsymbol{W}}$ corresponding to the ground truth class $t \in [K]$. Though other distance metrics could be also used as a measure of class similarity, we leave the discussion of analysing the different choices as future work. To verify our assumption, we check the heatmap of cosine similarities in Figure 2d, which clearly shows a similar pattern as the Pearson correlation of the teacher distribution $\boldsymbol{p}$ in Figure 2b. We call this method KD-sim. From Propositions 1 and 2, our proposed method, though simple and straightforward, can achieve our purpose of factoring out the class relationships effect.

Note that KD-sim doesn't require the knowledge of class hierarchy. However, if the hierarchy is available (as in CIFAR-

[3] In practice, besides tuning the temperature of the $\mathrm{softmax}$, one could also raise the similarities to a power < 1 to amplify the resolution of cosine similarities. Please refer to Section A in Suppl. for more details of our implementation.

100), we could also synthesize a teacher distribution apriori. In Suppl. Section B, we synthesize $\boldsymbol{\rho}$ by setting different values for (1) ground-truth class t, (2) classes within the same super-class of t, and (3) other incorrect classes. The quality of resulting method is slightly worse than KD-sim but can still improve student model's quality.

Compounded Effects. To enjoy the benefits from multiple effects and approximate the functionality of KD, we could combine the two partial-KD methods introduced above. We explore simple linear combination of synthetic teacher's distribution – $(1-\alpha)\boldsymbol{\rho}^{\text{pt}} + \alpha\boldsymbol{\rho}^{\text{sim}}$ and name the method KD-pt+sim. It is easy to verify that this compounded method performs example re-weighting and injects optimal prior geometry through class relationships.

In the next section, we evaluate our proposed partial-distillation methods on synthetic and real-world datasets to better understand how much each of these effects could benefit the student. Based on our understanding, we propose a novel distillation method that only adopts top-k largest values from the teacher distribution $\boldsymbol{p}$. In Section 5.1, we illustrate how this method could reduce noise in $\boldsymbol{p}$ (see Figure 2c), and also result in a better quality student model.

4.2. Empirical Studies

4.2.1. Synthetic Dataset

Performance of KD is dependent on the dataset properties. To this end, a natural question is – *Does* KD *perform only example re-weighting when all the classes are uncorrelated to each other?* We proved this to be true for binary classification (Section 3.3). To answer the same for multi-class classification task, we generate synthetic dataset, where we can control the class similarities within the same super-class.

Setup. Inspired by (Ma et al., 2018), we synthesize a classification dataset with K classes, and C super-classes. Each super-class will have equal number of K/C classes, and each class will be assigned with a basis vector. These basis vectors are carefully generated, so that we could control the class correlations within the same super-class. More specifically, we generate a single data-point as follows:

1. Randomly sample C orthonormal basis vectors, denoted by $\boldsymbol{u}_i \in \mathbb{R}^d \; \forall i \in [C]$.
2. For each orthonormal basis $\boldsymbol{u}_i$, we sample $(K/C-1)$ unit vectors $\boldsymbol{u}_j \in \mathbb{R}^d$ that are τ cosine similar to $\boldsymbol{u}_i$.
3. Randomly sample an input data point in d-dimensional feature space $\boldsymbol{x} \sim \mathcal{N}_d(\mathbf{0}, \mathbf{I})$.
4. Generate one-hot encoded label $\boldsymbol{y} \in \mathcal{Y}$ with target: $t = \arg\max_{k\in[K]} \left(\boldsymbol{u}_k^\top \hat{\boldsymbol{x}} + \sum_{m=1}^{M} \sin(a_m \boldsymbol{u}_k^T \hat{\boldsymbol{x}} + b_m)\right)$, where $\hat{\boldsymbol{x}}$ is the l_2-normalized $\boldsymbol{x}$; $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^M$ are arbitrary constants; and we refer to the controlled sin complexity term $M \in \mathbb{Z}^+$ as *task difficulty*.

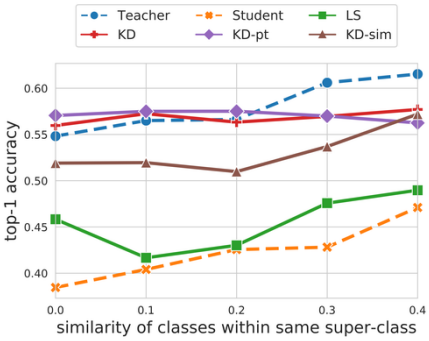


Figure 3: Best performance over 4 runs on a synthetic dataset with different levels of controlled class similarities.

After producing basis vectors with procedure (1) and (2), we run procedure (3) and (4) for $|\mathcal{D}|$ times with fixed basis vectors to generate a synthetic dataset $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})\}$. By tuning the cosine similarity parameter τ, we can control the classes correlations within the same super-class. Setting task-difficulty $M = 0$ generates a linearly separable dataset; and for $M > 0$, more non-linearities will be introduced by the sin function (see Figure 5 in Suppl. for visualization on a toy example). In the following experiments, we set input dimension $d = 500$ with $K = 50$ classes, and $C = 10$ super-classes. We use $|\mathcal{D}| = 500k$ data-points for training, and $|\mathcal{D}_{\text{valid}}| = 50k$ for validation. We use a simple 2-layer fully-connected neural network with tanh activation, and hidden layer dimensions 64 for the student, and 128 for the teacher. Finally, we set $M = 10$, hoping that this is the right task difficulty trade-off (i.e., not too easy, but hard enough to have a large margin between the two models for KD).

Results and Analysis. Figure 3 shows the classification accuracy on the validation set of all the methods, when varying τ from $0.0 \rightarrow 0.4$. We notice a large margin between the teacher and student. Label Smoothing (LS) marginally helps the student, while Knowledge Distillation (KD) benefits significantly. Interestingly, regardless of the class similarity τ; KD-pt has comparable performance with KD, suggesting that example re-weighting effect plays a major role in distillation. Thus, even when the classes are uncorrelated, KD could still benefit the student through example re-weighting. Note that for this task, the data-points which are close to the decision boundary are harder to classify, and can be regarded as difficult examples. Furthermore, when increasing τ, we see a significant improvement in performance of KD-sim, suggesting that the injected prior knowledge of class relationships boosts student model's quality.

Method	CIFAR-100	ImageNet
Teacher	75.68	77.98
Student	72.51	76.32
LS	73.87	76.83
KD	75.94	77.49
KD-pt	75.08	77.00
KD-sim	74.30	76.95
KD-pt+sim	75.24	77.17
KD-topk	**76.17**	**77.75**

Table 1: We report the mean Top-1 accuracy (%) over 4 individual runs with different initializations. Best k for KD-topk is 25 and 500 for CIFAR-100 and ImageNet, resp.

Method	#Params	Validation	Test
Teacher	24.2M	60.90	58.58
Student	9.1M	64.17	61.55
KD	9.1M	64.04	61.33
KD-topk	9.1M	**63.59**	**60.85**

Table 2: Validation and test Perplexity (lower is better) of compared methods on PTB language modeling. We report the best result over 4 individual runs with different initializations. Best k value for KD-topk is 100.

4.2.2. Real-world Datasets

We use two popular image classification datasets – CIFAR-100 (Krizhevsky et al., 2009) and ImageNet (Russakovsky et al., 2015) to analyze the quality of our proposed partial-distillation methods, and also to verify if we could approximate the performance of KD by compounding effects.

Setup. CIFAR-100 is a relatively small dataset with 100 classes. We use ResNet-20 as the student, and ResNet-56 as the teacher. ImageNet on the other hand is a large-scale dataset covering 1000 classes. We use ResNet-50 as the student, and ResNet-152 as the teacher. For more details, please refer to Section A in Suppl. Note that instead of using different model families as in (Furlanello et al., 2018; Yuan et al., 2019), we use same model architecture (i.e., ResNet) with different depths for the student and teacher to avoid unknown effects introduced by model family discrepancy.

Results and Analysis. Table 1 shows the overall performance when using the best hyper-parameters for each of the methods. On both the datasets, teacher model is much better than the student, and LS improves student model's generalization. KD can further boost student model's quality by a large margin, especially on CIFAR-100, where KD even outperforms the teacher. We try to uncover the different benefits from distillation using partial-KD methods. Both KD-pt, and KD-sim outperforms LS; especially KD-pt on CIFAR-100. This suggests that the different effects from distillation benefits the student in different aspects. Furthermore, by combining the two effects together in KD-pt+sim (using $\alpha = 0.5$), we see a further improvement in quality.

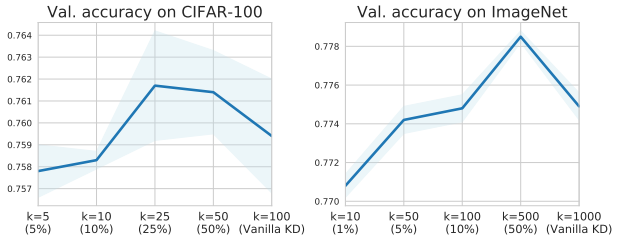


Figure 4: Top-1 accuracy vs. k for KD-topk on CIFAR-100 and ImageNet. We report the mean and standard deviation from 4 individual runs. On both datasets, using around $k = 10\%K$ for KD-topk can achieve comparable performances of KD, while best results are obtained when using $k < K$.

5. Improving and Diagnosing Knowledge Distillation

With a better understanding of KD's primary effects, in this section, we explore potential ways to improve student's quality as well as its training efficiency. Furthermore, we dive into failure cases of KD and diagnose the root causes.

5.1. Empirical Improvements for Distillation Quality

Though KD-sim is able to capture class relationships using different probability masses over the incorrect classes, there's a major drawback – all the examples from the same class will have the same relationships to the other classes. However, this is not a realistic assumption for most real-world datasets. For instance, on MNIST dataset, only some versions of '2' looks similar to '7'. To overcome this drawback, we propose KD-topk, which simply borrows the top-k largest values of teacher's probability $\boldsymbol{p}$, and uniformly distributes the rest of the probability mass to the other classes. Figure 2c shows that only preserving top-10 largest values could closely approximate the class correlations as in the full teacher's distribution $\boldsymbol{p}$, and is also less noisy. The above finding shows that only a few incorrect classes that are strongly correlated with the ground-truth class are useful for KD, and the probability mass on other classes are random noise (which is not negligible under high temperature T), and only has the effect of label smoothing in expectation.

Using the above intuition, we test KD-topk for image classification task on CIFAR-100 and ImageNet. Beyond computer vision, we also test KD-topk for language modeling

Teacher variants	**Teacher**	**KD**	**KD-pt**	**KD-sim**
ResNet-56 ($\epsilon = 0.0$)	75.39	76.00	74.81	74.40
ResNet-56 ($\epsilon = 0.1$)	76.69 $\uparrow$	75.02 $\downarrow$	74.13 $\downarrow$	74.01 $\downarrow$

Table 3: On CIFAR-100, Top-1 accuracy (in percentage) of the teachers with or without label smoothing, and the performances of the corresponding 'distilled' student for various knowledge distillation methods.

task on Penn Tree Bank (PTB) dataset. We apply state-of-the-art LSTM model (Merity et al., 2017) with different capacities for the teacher and student. Details of PTB dataset and model specifications are in Section A of Suppl. For image classification, the performance of KD-topk is shown in the last row of Table 1. We see that KD-topk outperforms KD in both the datasets. For language modeling, the results are shown in Table 2, which shows a similar trend for KD-topk. We plot the performance uplift of KD-topk along with k in Figure 4, which suggests that the best performance is achieved with proper tuning of $k < K$, which captures class relationships and also reduces noise. Note that the improvements of KD-topk over KD is simple and free with easy modifications. This is also the reason we omit a more sophisticated comparison with other advanced distillation techniques, such as (Romero et al., 2014; Yim et al., 2017).

5.2. Potential Improvements for Training Efficiency

Vanilla KD method requires loading a pre-trained teacher model in-memory, and computing forward-pass to get the full output distribution. While all of our proposed partial-KD methods and KD-topk can be achieved with better computational efficiency when training the student model. In terms of computation cost, we can pre-compute $K \times K$ class similarity matrix before student training, and directly use it for KD-sim. For KD-topk, we only need the top-k predictions from the teacher, which could be efficiently (approximately) computed using SVD-softmax (Shim et al., 2017), when the output space is large. Alternatively for vanilla KD, one could also save computation by storing teacher's predictions on-disk, which could again be optimized using our proposed methods. We only need to store a single value, i.e., teacher's confidence on ground-truth class for KD-pt; and top-k predictions for KD-topk. As shown in our experiments in Figure 4, using $k = 10\%K$, we can achieve comparable performance with vanilla KD.

5.3. Diagnosis of Failure Cases

A good understanding of KD enables us to diagnose failure cases. Müller et al. (2019) observed that although label smoothing improves teacher model's quality, it results in a worse student model when applying KD. Verified in CIFAR-100 (see Table 3), we found the unfavorable distillation performance could be attributed to two factors – Firstly, as argued by the Müller et al. (2019) and illustrated in Figure 7 in Suppl., LS destroys class relationships. Secondly, we found that the skewed teacher's prediction distribution on the ground-truth (see Figure 6 in Suppl.) also hinders the effectiveness of KD, because the example re-weighting effect will be less useful. Results of KD-sim and KD-pt from last two columns of Table 3 verify these findings. For another failure case, Mirzadeh et al. (2019) showed that the 'distilled' student model's quality gets worse as we continue to increase teacher model's capacity. The larger capacity teacher might overfit, and predict high (uniform) confidence on the ground truth on all the examples; thereby hindering the effectiveness of example re-weighting in knowledge distillation. Another explanation could be that there exists an optimal model capacity gap between the teacher and student, which could otherwise result in inconsistency between teacher's confidence on the ground-truth, and the desired example difficulty for the student. Perhaps, an 'easy' example for larger capacity teacher is overly difficult for the student, due to the large difference in model expressiveness.

6. Conclusion

We provide novel methods to understand the mechanism of knowledge distillation (KD). Through systematic analyses, we uncover two key beneficial effects of KD over label smoothing. Firstly, supervision from teacher's output distribution re-weights the training examples based on teacher's prediction on the ground-truth. Secondly, teacher's probability mass on the incorrect classes reveals class relationships, and more importantly, inject prior knowledge of the optimal geometry of student's output logit layer. These effects also explain why sometimes a better teacher may not be suitable for distillation, and self-distillation gives quality gains. To have a closer look at these two effects, we proposed partial-KD methods, and evaluated their performance on both synthetic and real-world datasets. Experimental results not only supported our claims, but also inspire ways to improve KD. Besides the applications suggested in Section 5, we think that our findings could also help with the recent advances of KD in private learning (Papernot et al., 2018), semi-supervised learning (Yalniz et al., 2019), etc.

In future work, we would like to extend our understanding of knowledge distillation under different data distributions, e.g., uniform vs long-tail distribution; and also consider the

effect of noisy inputs and labels. We would like to also investigate other cheaper and effective ways of distillation *e.g.,* looking into approximate versions of KD-topk.

References

Anil, R., Pereyra, G., Passos, A., Ormándi, R., E. Dahl, G., and E. Hinton, G. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Chen, Y., Wang, N., and Zhang, Z. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. *arXiv preprint arXiv:1707.01220*, 2017.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, pp. 4278–4287, 2017.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 558–567, 2019.

Heo, B., Lee, M., Yun, S., and Choi, J. Y. Knowledge distillation with adversarial samples supporting decision boundary. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3771–3778, 2019.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Kim, Y. and Rush, A. M. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. Unifying Distillation and Privileged Information. *arXiv preprint arXiv:1511.03643*, 2015.

Ma, J., Zhao, Z., Yi, X., Chen, J., Hong, L., and Chi, E. H. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1930–1939. ACM, 2018.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. Recurrent neural network based language model. In *Interspeech*, 2010.

Mirzadeh, S.-I., Farajtabar, M., Li, A., and Ghasemzadeh, H. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *arXiv preprint arXiv:1902.03393*, 2019.

Müller, R., Kornblith, S., and Hinton, G. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597. IEEE, 2016.

Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., and Erlingsson, Ú. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

Phuong, M. and Lampert, C. Towards understanding knowledge distillation. In *International Conference on Machine Learning*, pp. 5142–5151, 2019.

Ranjan, R., Castillo, C. D., and Chellappa, R. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Roux, N. L. Tighter bounds lead to improved classifiers. *arXiv preprint arXiv:1606.09202*, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Shim, K., Lee, M., Choi, I., Boo, Y., and Sung, W. Svd-softmax: fast softmax approximation on large vocabulary neural networks. In *Advances in Neural Information Processing Systems*, pp. 5463–5473, 2017.

Srinivas, S. and Fleuret, F. Knowledge transfer with jacobian matching. *arXiv preprint arXiv:1803.00443*, 2018.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Tang, J. and Wang, K. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2289–2298. ACM, 2018.

Vapnik, V. and Izmailov, R. Learning Using Privileged Information: Similarity Control and Knowledge Transfer. *Journal of machine learning research*, 16(2023-2049):2, 2015.

Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

Yim, J., Joo, D., Bae, J., and Kim, J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4133–4141, 2017.

Yuan, L., Tay, F. E., Li, G., Wang, T., and Feng, J. Revisit knowledge distillation: a teacher-free framework. *arXiv preprint arXiv:1909.11723*, 2019.

Zhang, X., Yu, F. X., Karaman, S., Zhang, W., and Chang, S.-F. Heated-up softmax embedding. *arXiv preprint arXiv:1809.04157*, 2018.

Supplementary Material

A. Experimental details

Implementation of KD**.** In practice, the gradients from the RHS of Equation (5) are much smaller compare to the gradients from LHS when temperature T is large. Thus, it makes tuning the balancing hyper-parameter λ become non-trivial. To mitigate this and make the gradients from two parts in the similar scale, we multiply T^2 to the RHS of Equation (5), as suggested in (Hinton et al., 2015).

Implementation of KD**-sim.** When synthesizing the teacher distribution for KD-sim, we use $\boldsymbol{\rho}^{\text{sim}} = \text{softmax}(\hat{\boldsymbol{w}}_t \hat{\boldsymbol{W}}^\top)$, where $\hat{\boldsymbol{W}}$ is the l_2-normalized logit layer weights and $\hat{\boldsymbol{w}}_t$ is the t-th row of $\hat{\boldsymbol{W}}$. However, the cosine similarities computed for softmax are limited in the range of $[0, 1]$. Therefore the resulting distribution is highly likely to be uniform. To mitigate this and bring more resolution to be cosine similarities, we use the following:

$$\boldsymbol{\rho}^{\text{sim}} = \text{softmax}((\hat{\boldsymbol{w}}_t \hat{\boldsymbol{W}}^\top)^\alpha / \beta).$$

Here $\alpha < 1$ is a hyper-parameter to amplify the resolution of cosine similarities, β is another hyper-parameter indicating the temperature for softmax.

Synthetic dataset. First, we follow the data generation procedure described in Section 4.2.1, we get a toy synthetic dataset where we only have input dimensionality $d = 2$ with $K = 4$ classes and $C = 2$ super-classes. Figure 5 shows a series of scatter plots with different settings of class similarity τ and task difficulty M. This visualization gives a better understanding of the synthetic dataset and helps us imagine what it will look like in high-dimensional setting that used in our experiments. For the model used in our experiments, besides they are 2-layer network activated by $\tanh$, we use residual connection (He et al., 2016) and and batch normalization (Ioffe & Szegedy, 2015) for each layer. Following (Ranjan et al., 2017; Zhang et al., 2018), we found using l_2-normalized logits layer weight $\hat{\boldsymbol{W}}$ and penultimate layer $\hat{\boldsymbol{h}}$ provides more stable results. The model is trained by ADAM optimizer (Kingma & Ba, 2014) for a total of 3 million steps without weight decay and we report the best accuracy. Please refer to Table 4a for the best setting of hyper-parameters.

CIFAR-100 dataset. CIFAR-100 is a relatively small dataset with low-resolution (32×32) images, containing $50k$ training images and $10k$ validation images, covering 100 classes and 20 super-classes. It is a perfectly balanced dataset – we have the same number of images per class (*i.e.,* each class contains 500 training set images) and 5 classes per super-class. To process the CIFAR-100 dataset, we use the official split from Tensorflow Dataset[4]. Both data aug-

Method	Hyper-parameter setting
LS	$\epsilon = 0.3$ for any τ.
KD	$\lambda = 0.7, T = 3$ when $\tau = 0.0$
	$\lambda = 0.7, T = 5$ when $\tau = 0.1$
	$\lambda = 0.7, T = 2$ when $\tau = 0.2$
	$\lambda = 0.7, T = 3$ when $\tau = 0.3$
	$\lambda = 0.7, T = 10$ when $\tau = 0.4$
	$\lambda = 0.7, T = 5$ when $\tau = 0.5$;
KD-pt	$\lambda = 0.7, T = 3$ when $\tau = 0.0$
	$\lambda = 0.7, T = 5$ when $\tau = 0.1$
	$\lambda = 0.7, T = 2$ when $\tau = 0.2$
	$\lambda = 0.7, T = 3$ when $\tau = 0.3$
	$\lambda = 0.7, T = 10$ when $\tau = 0.4$
	$\lambda = 0.7, T = 5$ when $\tau = 0.5$
KD-sim	$\lambda = 0.7, \alpha = 0.5, \beta = 0.5$ for any τ

(a) Synthetic

Method	Hyper-parameter setting
LS	$\epsilon = 0.1$
KD	$\lambda = 0.3, T = 5$
KD-pt	$\lambda = 0.3, T = 5$
KD-sim	$\lambda = 0.3, \alpha = 0.3, \beta = 0.3$
KD-topk	$k = 25, \lambda = 0.5, T = 5$

(b) CIFAR-100

Method	Hyper-parameter setting
LS	$\epsilon = 0.1$
KD	$\lambda = 0.7, T = 20$
KD-pt	$\lambda = 0.2, T = 25$
KD-sim	$\lambda = 0.3, \alpha = 0.5, \beta = 0.3$
KD-topk	$k = 500, \lambda = 0.5, T = 3$

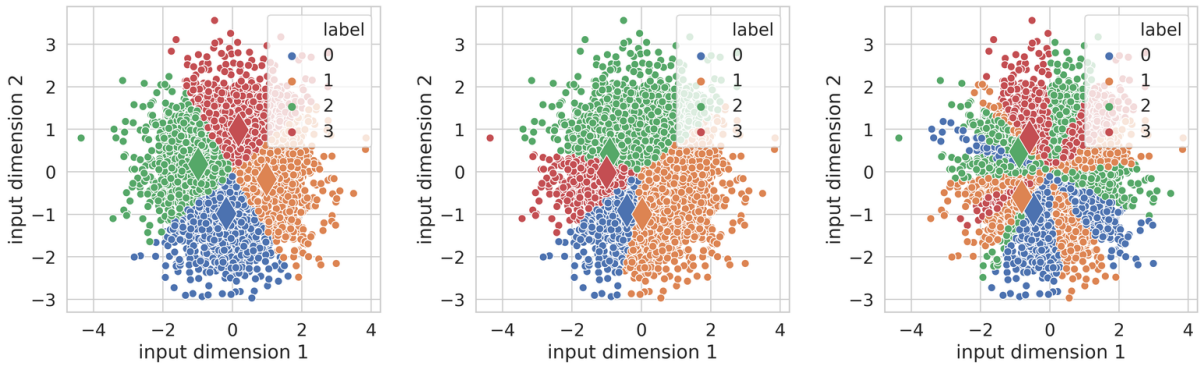
(c) ImageNet

Method	Hyper-parameter setting
KD	$\lambda = 0.1, T = 50$
KD-topk	$k = 100, \lambda = 0.1, T = 50$

(d) Penn Tree Bank (PTB)

Table 4: Hyper-parameter settings for different methods on different datasets.

[4] https://www.tensorflow.org/datasets/catalog/cifar100



(a) (b) (c)

Figure 5: Visualization of 5K synthetic data points (with input dimensionality $d = 2$) on 2-D plane. We use $K = 4$, $C = 2$, means there are two super-classes, one associate with label $\{0,1\}$ and the other one associate with label $\{2,3\}$. We vary τ and M and produce 3 plots: (a) $\tau = 0.0$, no sine function is used; (b) $\tau = 0.9$, no sine function is used and (c) $\tau = 0.9$, $M = 2$.

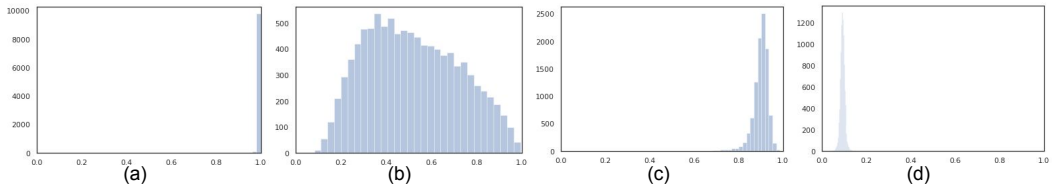


Figure 6: On CIFAR-100, we plot the histograms of teacher's confidence on ground-truth p_t, here teacher model is ResNet-56 with different levels ϵ of label smoothing. From left to right, we use: (a) $\epsilon = 0.0$ and $T = 1$; (b) $\epsilon = 0.0$ and $T = 5$; (c) $\epsilon = 0.1$ and $T = 1$ and (d) $\epsilon = 0.1$ and $T = 3$. The distribution of p_t becomes skewed after enabling label smoothing.

mentation [5][6] for CIFAR-100 and the ResNet model[7] are based on Tensorflow official implementations. Also, following the conventions, we train all models from scrach using Stochastic Gradient Descent (SGD) with a weight decay of 1e-3 and a Nesterov momentum of 0.9 for a total of 10K steps. The initial learning rate is 1e-1, it will become 1e-2 after 40K steps and become 1e-3 after 60K steps. We report the best accuracy for each model. Please refer to Table 4b for the best setting of hyper-parameters.

ImageNet dataset. ImageNet contains about 1.3M training images and $50k$ test images, all of which are high-resolution (224×224), covering 1000 classes. The distribution over the classes is approximately uniform in the training set, and strictly uniform in the test set. Our data preprocessing and model on ImageNet dataset are follow Tensorflow TPU official implementations[8]. The Stochastic Gradient Descent (SGD) with a weight decay of 1e-4 and a Nesterov momentumof 0.9 is used. We train each model for 120 epochs, the mini-batch size is fixed to be 1024 and low precision (FP16) of model parameters is adopted. We didn't change the learning rate schedule scheme from the original implementation. Please refer to Table 4c for the best setting of hyper-parameters.

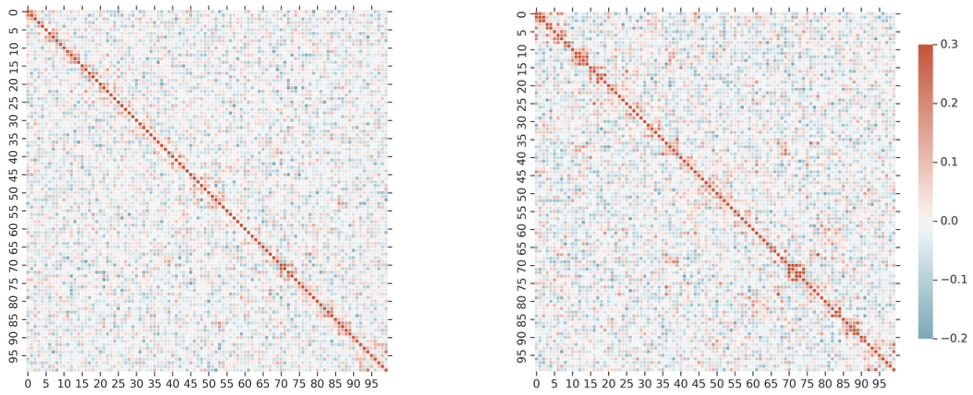
Penn Tree Bank dataset. We use Penn Tree Bank (PTB) dataset for word-level language modeling task using the standard train/validation/test split by (Mikolov et al., 2010). The vocabulary is capped at 10K unique words. We consider the state-of-the-art LSTM model called AWD-LSTM proposed by Merity et al. (2017). The model used several regularization tricks on top of a 3-layer LSTM, including DropConnect, embedding dropout, tied weight, etc. We use different capacity (indicated by hidden size and embedding size) as our Teacher and Student. To be specific, Teacher has a hidden size of 1150 and an embedding size of 400, while Student has a smaller hidden size of 600 and a smaller

[5] https://github.com/tensorflow/models/blob/master/research/resnet/cifar_input.py

[6] We turn on the random brightness/saturation/constrast for better model performance.

[7] https://github.com/tensorflow/models/blob/master/research/resnet/resnet_model.py

[8] https://github.com/tensorflow/tpu/tree/master/models/official/resnet



(a) (b)

Figure 7: Using 10K samples from CIFAR-100 for ResNet-56 trained with smoothed labels ($\epsilon = 0.1$), we plot (a) Pearson correlations of $\boldsymbol{p}$ when $T = 10$ and (b) Cosine similarities computed from the weights of the final logits layer. Since every 5 classes are within the same super-class, class correlations can be interpreted as the color 'squares' on the diagonal.

embedding size of 300. We follow the official implementation[9] with simple changes for KD-topk. Besides capacity, we keep the default hyper-parameter as in the official implementation to train our Teacher. However, when training smaller Student model, we follow another implementation[10] to: (1) lower the learning rate to 0.2, (2) increase training epochs to 1000, (3) use 0.4 for embedding dropout rate and (4) use 0.225 for RNN layer dropout rate.

B. Additional Experiments

Method	% top-1 accuracy
Student	72.51
KD	75.94
KD-rel	74.14
KD-sim	74.30
KD-pt+rel	75.07
KD-pt+sim	75.24

Table 5: Performance of KD-rel on CIFAR-100. We report the mean result for 4 individual runs with different initializations. We use $\beta_1 = 0.6, \beta_2 = \frac{0.1}{4}, \beta_3 = \frac{0.3}{95}$.

Examine optimal geometry prior effect with class hierarchy. In section 4, we mentioned the optimal geometry prior effects of KD can also be examined using existing class hierarchy. Suppose our data has a pre-defined class hierarchy (e.g., on CIFAR-100), we can also use it to examine the optimal geometry prior effects of KD. To be specific, let $\mathbb{S}_t \subset [K] \backslash t$ denote the other classes that share same parent of t. We simply assign different probability masses to different types of classes:

$$\rho_i^{\text{rel}} = \begin{cases} \beta_1 & \text{if } i = t, \\ \beta_2 & \text{if } i \in \mathbb{S}_t, \\ \beta_3 & \text{otherwise}, \end{cases} \tag{9}$$

where $\beta_1 > \beta_2 > \beta_3$ are a hyper-parameters we could search and optimize, and we name this method as KD-rel. As shown in Table 5, we found KD-rel performs slightly worse than KD-sim on CIFAR-100. The trend is still hold when we compound each effect with KD-pt.

[9] https://github.com/salesforce/awd-lstm-lm

[10] https://github.com/zihangdai/mos