BETTER SUPERVISORY SIGNALS
BY OBSERVING LEARNING PATHS

ABSTRACT

Better supervision might lead to better performance. In this paper, we first clarify what makes supervision good for a classification problem, and then explain two existing label refining methods: label smoothing and knowledge distillation, in terms of our proposed criterion. To further answer why and how better supervision emerges, we observe the learning path, i.e., the trajectory of the network's predicted distribution, for each training sample during training. We surprisingly find that the model can spontaneously refine "bad" labels with through a "zig-zag" learning path, which occurs on both toy and real datasets. Observing the learning path not only provides a new perspective for understanding knowledge distillation, overfitting, and learning dynamics, but also reveals that the supervisory signal of a teacher network can be very unstable near the best points in training on real tasks. Inspired by this, we propose a new algorithm, Filter-KD, which further enhances classification performance of knowledge distillation, as verified by experiments in various settings.

1 INTRODUCTION

In multi-class classification problems, we usually supervise our model with "one-hot" labels: label vectors y which have $y_i = 1$ for one i, and 0 for all other dimensions. Over time, however, it has gradually become clear that this "default" setup is not always the best choice in practice, as measured by the generalization performance of the resulting model.[1] One alternative is to summarize a distribution of human annotations, as Peterson et al. (2019) did on CIFAR-10. An alternative approach is label smoothing (e.g. Szegedy et al., 2016), mixing between a one-hot label and the uniform distribution. Knowledge distillation (KD), first training a teacher network on the training set and then a student network on the teacher's output probabilities, was originally proposed for model compression (Hinton et al., 2015) but can also be thought of as refining the supervision signal: it provides "soft" teacher outputs rather than hard labels to the student.

Knowledge distillation is promising because it requires no additional annotation effort, but, unlike label smoothing, can still provide sample-specific refinement. Perhaps surprisingly, knowledge distillation can improve student performance even when the teacher is of *exactly the same form as the student*; this is known as self-distillation (Furlanello et al., 2018; Zhang et al., 2019). There have been many recent attempts to explain the success of knowledge distillation and specifically self-distillation (e.g. Menon et al., 2021; Allen-Zhu & Li, 2020; Tang et al., 2020), from both optimization and supervision perspectives. We focus on the latter area, where it is usually claimed that the teacher provides useful "dark knowledge" to the student through its labels.

Inspired by this line of work, we further explore why and how this improved supervisory signal emerges during the teacher's one-hot training. Specifically, we first clarify that given any input sample $\mathbf{x}$, good supervision signals should be close (in L_2 distance) to the ground truth categorical distribution, i.e., $p^*(y \mid \mathbf{x})$. We then show that a neural network (NN) can automatically refine "bad labels", those where $p^*(y \mid \mathbf{x})$ is far from the one-hot vector from the training set.[2] During one-hot training, the model prediction on such a sample first moves towards $p^*(y \mid \mathbf{x})$, and then slowly

---

[1]We are concerned here only with traditional i.i.d. generalization from the training set to held-out test sets.

[2]This might be because $\mathbf{x}$ is ambiguous (perhaps $p^*(y \mid \mathbf{x})$ is flat, or we simply got a sample from a less-likely class), or because the one-hot label has been corrupted through label noise or otherwise is "wrong."

converges to its supervisory label, following a "zig-zag" pattern. A well-trained teacher, one that does not overfit to particular training labels, can thus provide supervisory signals closer to $p^*(y \mid \mathbf{x})$. We find this phenomenon is quite common in gradient descent training, and provide some analytical justification (Section 3.3). Based on our explanations, we point out that this signal can be better achieved if we take a moving average of the teacher's prediction, an algorithm we term Filter-KD. This approach provides better supervision and hence better final performance, especially when there are more bad labels.

2 SUPERVISION INFLUENCES GENERALIZATION

We will begin by establishing the classification setting, and making clearer how the choice of supervisory signal affects the learned model.

2.1 CHOICES OF SUPERVISION SIGNAL

In K-way classification, our goal is to learn a mapping $f : \mathcal{X} \to \Delta^K$ that can minimize the risk

$$R(f) \triangleq \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathbb{P}}[L(y, f(\mathbf{x})] = \int p(\mathbf{x})\, p(y|\mathbf{x})\, L(y, f(\mathbf{x}))\, \mathrm{d}\mathbf{x}\, \mathrm{d}y, \tag{1}$$

where $L(y, f(\mathbf{x}))$ is the loss function measuring the gap between model's prediction and the true label (e.g. cross-entropy or square loss). The true label y is an integer ranging from 1 to K, and the input signal $\mathbf{x}$ is usually high dimensional, e.g., an image or a sequence of word embeddings. The joint distribution of $(\mathbf{x}, y)$ is $\mathbb{P}$, whose density[3] can be written as $p(\mathbf{x})p(y|\mathbf{x})$. In practice, as $\mathbb{P}$ is unknown, we instead (approximately) minimize the empirical risk

$$R_{\mathrm{emp}}(f, \mathcal{D}) \triangleq \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{1}{N}\mathbb{1}(y_n\!=\!k)L\left(k, f(\mathbf{x}_n)\right) = \sum_{n=1}^{N}\frac{1}{N}\mathbf{e}_{y_n}^{\mathsf{T}}\mathbf{L}(f(\mathbf{x}_n)), \tag{2}$$

where $\mathbb{1}(y_n = k)$ is an indicator function which equals 1 if $y_n = k$ or 0 otherwise, and $\mathbf{e}_{y_n} \in \{0, 1\}^K$ is its one-hot vector form. $\mathbf{L}(f(\mathbf{x}_n)) = (L(1, f(\mathbf{x}_n)), \ldots, L(K, f(\mathbf{x}_n))) \in \mathbb{R}^K$ is the loss for each possible label. In R_{emp}, the N training pairs $\mathcal{D} \triangleq \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ are sampled i.i.d. from $\mathbb{P}$.

Comparing (2) to (1), we find that $p(\mathbf{x})$ is approximated by an uniform distribution over the samples, which is reasonable. However, using an indicator function (i.e., one-hot distribution) to approximate $p(y \mid \mathbf{x})$ bears more consideration. For example, if a data point $\mathbf{x}$ is quite vague and its true $p(y|\mathbf{x})$ is flat or multimodal, we might hope to see $\mathbf{x}$ multiple times with different label y during training. But actually, most datasets have only one copy of each $\mathbf{x}$, so we only ever see one corresponding $\mathbf{e}_y$. Although R_{emp} is an unbiased estimator for R, if we used a better (e.g. lower-variance) estimate of $p(y \mid \mathbf{x})$, we could get a better estimate for R and thus, hopefully, better generalization.

Specifically, suppose we were provided a "target" distribution $p_{\mathrm{tar}}(y \mid \mathbf{x})$ (written in vector form as $\mathbf{p}_{\mathrm{tar}}(\mathbf{x})$) for each training point x, as $\mathcal{D}' = \{(\mathbf{x}_n, \mathbf{p}_{\mathrm{tar}}(\mathbf{x}_n))\}_{n=1}^{N}$. Then we could use

$$R_{\mathrm{tar}}(f, \mathcal{D}') \triangleq \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{1}{N}p_{\mathrm{tar}}(y_n = k \mid \mathbf{x}_n)L(k, f(x_n)) = \sum_{n=1}^{N}\frac{1}{N}\mathbf{p}_{\mathrm{tar}}(\mathbf{x}_n)^{\mathsf{T}}\mathbf{L}(f(\mathbf{x}_n)). \tag{3}$$

Standard training with R_{emp} is a special case of R_{tar}, using $\mathbf{p}_{\mathrm{tar}}(\mathbf{x}_n) = \mathbf{e}_{y_n}$. The CIFAR-10H dataset (Peterson et al., 2019) is one attempt at a different $\mathbf{p}_{\mathrm{tar}}$, using multiple human annotators to estimate $\mathbf{p}_{\mathrm{tar}}$. Label smoothing (e.g. Szegedy et al., 2016) sets $\mathbf{p}_{\mathrm{tar}}$ to a convex combination of $\mathbf{e}_y$ and the constant vector $\frac{1}{K}\mathbf{1}$. In knowledge distillation (KD; Hinton et al., 2015), a teacher is first trained on $\mathcal{D}$, then a student learns from $\mathcal{D}'$ with $\mathbf{p}_{\mathrm{tar}}$ based on the teacher's outputs. All three approaches yield improvements over standard training with R_{emp}.

---

[3]Because we are working with classification problems, we use densities with respect to a product of some arbitrary measure on $\mathbf{x}$ (probably Lebesgue) with counting measure on y, and assume that these densities exist for notational convenience. None of our arguments will depend on the choice of base measure.

(a) tSNE of toy dataset.   (b) ACC vs $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ (c) ECE v.s. $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$

Figure 1: Experiments on a toy dataset when learning from different $\mathbf{p}_{\text{tar}}$. In (b-c), the horizontal axis represents $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$, and the vertical axis is the generalization performance. OHT means one-hot training (on R_{emp}), LS means label smoothing, GT means ground truth training with $\mathbf{p}^*$, KD is knowledge distillation, and ESKD is early-stopped KD. The Spearman correlation coefficient for results in (b) is -0.930 with p-value 1.9×10^{-53}; for (c) is 0.895 with p-value 2.7×10^{-43}.

2.2 Measuring the Quality of Supervision

Choosing a different $\mathbf{p}_{\text{tar}}$, then, can lead to a better final model. Can we characterize which $\mathbf{p}_{\text{tar}}$ will do well? We propose the following, as a general trend.

Hypothesis 1. *Suppose we train a model supervised by $\mathbf{p}_{tar}$, that is, we minimize $R_{tar}(f, \mathcal{D}')$. Then, smaller average L_2 distance between $\mathbf{p}_{tar}$ and the ground truth $\mathbf{p}^*$ on these samples, i.e. small $\mathbb{E}_{\mathbf{x}}\left[\|\mathbf{p}_{tar}(\mathbf{x}) - \mathbf{p}^*(\mathbf{x})\|_2\right]$, will in general lead to better generalization performance.*

This hypothesis is suggested by Proposition 3 of Menon et al. (2021), which shows that for any predictor f and loss bounded as $L(y, \hat{y}) \leq \ell$,

$$\mathbb{E}_{\mathbf{x}}\left[(R_{\text{tar}}(f, \mathcal{D}') - R(f))^2\right] \leq \frac{1}{N} \underset{\mathbf{x}}{\text{Var}}\left[\mathbf{p}_{\text{tar}}^{\mathsf{T}} \mathbf{L}(f(\mathbf{x}))\right] + K\ell^2 \left(\mathbb{E}_{\mathbf{x}}\left[\|\mathbf{p}_{\text{tar}}(\mathbf{x}) - \mathbf{p}^*(\mathbf{x})\|_2\right]\right)^2. \quad (4)$$

When N is large, the second term will dominate the right-hand side, implying smaller average $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|$ will lead to R_{tar} being a better approximation of the true risk R. Minimizing a better R_{tar} should then lead to a better model, suggesting that the quality of the supervision signal can be roughly measured by its L_2 distance to the ground truth $\mathbf{p}^*$.

To further support this hypothesis, we conduct experiments on a synthetic Gaussian problem (Figure 1 (a); details in Appendix A), where we can easily calculate $p^*(y \mid \mathbf{x})$ for each sample. We first generate several different $\mathbf{p}_{\text{tar}}$ by adding noise[4] to the ground truth $\mathbf{p}^*$, then train simple 3-layer NNs under that supervision. We also show five baselines: one-hot training (OHT), label smoothing (LS), KD, early-stopped KD (ESKD), and ground truth (GT) supervision (using $\mathbf{p}^*$). KD refers to a teacher trained to convergence, while ESKD uses a teacher stopped early based on validation accuracy. We early-stop the student's training in all settings. From Figure 1 (b-c), it is clear that smaller $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ leads to better generalization performance, as measured either by accuracy (ACC) or expected calibration error (ECE)[5] on a held-out test set. Appendix A has more detailed results.

3 Insights From the Learning Path

In the toy example of Section 2, we see that ESKD outperforms other baselines in accuracy by a substantial margin (and all baselines are roughly tied in ECE). We expect that supervision with smaller $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ leads to better generalization performance, but it is not clear how better $\mathbf{p}_{\text{tar}}$ emerges from when the teacher in ESKD is trained using one-hot labels. This section will answer this, by observing the learning paths of training samples.

---

[4]Whenever we mention adding noise to $\mathbf{p}^*$, we mean we add independent noise to each dimension, and then re-normalize it to be a distribution. Large noise can thus flip the "correct" label.

[5]ECE measures the calibration of a model (Guo et al., 2017). Briefly, lower ECE means the model's confidence in its predictions is more accurate. See Appendix E for details.

Figure 2: Normalized (divided by $\sqrt{2}$) distance between output distribution $\mathbf{q}$ and $\mathbf{p}^*$ during the one-hot training in different stages (left to right: initial, early stop, convergence). In these figures, $\|\mathbf{q} - \mathbf{p}^*\|_2$ is the mean height of all points in the figure. We provide more results about the NNs trained under different supervisions using this fashion in Appendix F.

3.1 PAY MORE ATTENTION TO HARDER SAMPLES

For a finer-grained understanding of early stopping the teacher, we would like to better understand how the teacher's predictions evolve in training. Assuming Hypothesis 1, the main factor is $\mathbb{E}_{\mathbf{x}}\|\mathbf{q}(\mathbf{x}) - \mathbf{p}^*(\mathbf{x})\|_2$, where $\mathbf{q}$ is the teacher's output probability distribution. We expect, though, that this term will vary for different $\mathbf{x}$, in part because some samples are simply more difficult to learn; as a proxy for this, we define **base difficulty** as $\|\mathbf{e}_y - \mathbf{p}^*(\mathbf{x})\|_2$. This measure is high if:

- $\mathbf{x}$ is ambiguous: $\mathbf{p}^*$ has several large components, so there is no one-hot label near $\mathbf{p}^*$.
- $\mathbf{x}$ is not very ambiguous (there is a one-hot label near $\mathbf{p}^*$), but the sample was "unlucky" and drew y from a low-probability class.

Figure 2 shows these two quantities at three points in training: initialization, the point where ESKD's teacher stops, and convergence. At initialization, most points[6] have large $\|\mathbf{q}(\mathbf{x}) - \mathbf{p}^*(\mathbf{x})\|_2$. By the point of early stopping, most $\mathbf{q}(\mathbf{x})$ values are roughly near $\mathbf{p}^*$. At convergence, however, $\mathbf{q}(\mathbf{x}) \approx \mathbf{e}_y$, as the classifier has nearly memorized the training inputs, leading to a diagonal line in the plot.

It is clear the biggest contributors to $\mathbb{E}_{\mathbf{x}}\|\mathbf{q}(\mathbf{x}) - \mathbf{p}^*(\mathbf{x})\|_2$ when training a model to convergence are the points with high base difficulty. Per Hypothesis 1, these are the points we should most focus on.

3.2 LEARNING PATH OF DIFFERENT SAMPLES

To better understand how $\mathbf{q}$ changes for each sample, we track the model's outputs of all training samples at each training step. We plot the vectors of three probabilities as points on the simplex (details in Appendix E), and visualize four samples with different base difficulty in Figure 3.

The easy samples very quickly move to the correct location near $\mathbf{e}_y$ (as indicated by the light color until reaching the corner). The medium sample takes a less direct route, drifting off slightly towards $\mathbf{p}^*$, but still directly approaches $\mathbf{e}_y$. The hard sample, however, does something very different: it first approaches $\mathbf{p}^*$, but then veers off towards $\mathbf{e}_y$, giving a "zig-zag" path. In both the medium and hard cases, there seems to be some "unknown force" dragging the learning path towards $\mathbf{p}^*$; in both cases, the early stopping point is not quite perfect, but noticeably closer to $\mathbf{p}^*$ than the final converged point near $\mathbf{e}_y$. In other words, during one-hot training, the NNs can *spontaneously refine the "bad labels."* Under Hypothesis 1, this partly explains the superior performance of ESKD to KD with a converged teacher.[7]

We only show four points, because they are quite representative of all samples, under different toy-dataset settings. We also provide a "zig-zagness score" to further illustrate the correlation between base difficulty and the learning path patterns (refer to Appendix C for more details).

---

[6]The curve structure is expected: points with $\mathbf{p}^* \approx (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ are near the middle of the base difficulty range, and their initialization is also fairly ambiguous.

[7]KD's practical success is also related to the temperature and optimization effects, among others; better supervision is not the whole story. We discuss the effect of various hyperparameters in Appendix B.

Figure 3: Learning path of four samples with different base difficulty. Corners correspond to one-hot vectors for each class. Colors represent training time: transparent at initialization (marked with a green diamond), fully dark blue at the end of training. Also marked: the training label y (red cross), the unobserved true $\mathbf{p}^*$ (gray cross), the early stopping epoch (red thin cross).

3.3 EXPLANATION OF PATTERNS IN THE LEARNING PATH

Where does this "unknown force" pushing $\mathbf{q}$ towards $\mathbf{p}^*$ come from? The following decomposition, proved in Appendix G, will help elucidate its source:

Proposition 1. *Let $\mathbf{z}^t(\mathbf{x}) \triangleq f(\mathbf{w}^t, \mathbf{x})$ denote the NN output logits with parameters $\mathbf{w}^t$, and $\mathbf{q}^t(\mathbf{x}) = $ Softmax$(\mathbf{z}^t(\mathbf{x}))$ the corresponding probabilities. Let $\mathbf{w}^{t+1} \triangleq \mathbf{w}^t - \eta \, \nabla_\mathbf{w} \left(\mathbf{p}_{tar}(\mathbf{x}_u)^\mathsf{T} \mathbf{L}(\mathbf{q}^t(\mathbf{x}_u)) \right)$ be the result of applying one step of SGD to $\mathbf{w}^t$ on the data point $(\mathbf{x}_u, \mathbf{p}_{tar}(\mathbf{x}_u))$ with learning rate η. Then, the change in predictions for a particular observed sample, $\mathbf{x}_o$, is*

$$\mathbf{q}^{t+1}(\mathbf{x}_o) - \mathbf{q}^t(\mathbf{x}_o) = \eta \, \mathcal{A}^t(\mathbf{x}_o) \, \mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u) \, \left(\mathbf{p}_{tar}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u) \right) + \mathcal{O}(\eta^2),$$

where $\mathcal{A}^t(\mathbf{x}_o) = \nabla_\mathbf{z} \mathbf{q}^t(\mathbf{x}_o)$ and $\mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u) = (\nabla_\mathbf{w} \mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t}) \, (\nabla_\mathbf{w} \mathbf{z}(\mathbf{x}_u)|_{\mathbf{w}^t})^\mathsf{T}$ are $K \times K$ matrices.

The matrix $\mathcal{K}^t$ is known as the *empirical neural tangent kernel*, which we can think of roughly as a notion of "similarity" between $\mathbf{x}_o$ and $\mathbf{x}_u$ based on the network's representation at time t. In general, $\mathcal{K}^t$ changes during training in potentially complex ways, but in very wide networks with appropriate random initialization, it is nearly invariant throughout training and nearly independent of the initialization (Jacot et al., 2018; Arora et al., 2019). For more details, see Appendix G.

Figure 4 shows the learning path of a hard sample during one-hot training, say $\mathbf{x}_o$, where the label $\mathbf{e}_{y_o}$ is far from $\mathbf{p}^*(\mathbf{x}_o)$. In each epoch, $\mathbf{w}$ will receive $N-1$ updates based on the loss of $\mathbf{x}_u \neq \mathbf{x}_o$ (the small blue path), and one update based on $\mathbf{x}_o$ (the big red path). At any time t, "dissimilar" samples will have small $\mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u)$ (as measured, e.g., by its trace), and hence only slightly affect the predicted label for $\mathbf{x}_o$. Similar samples will have large $\mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u)$, and hence affect its updates much more; because $\mathbf{p}^*$ is hopefully similar for similar $\mathbf{x}$ values, it is reasonable to expect that the mean of $\mathbf{e}_y$ for data points with similar $\mathbf{x}$ will be close to $\mathbf{p}^*(\mathbf{x}_o)$. Thus the net effect of updates for $\mathbf{x}_u \neq \mathbf{x}_o$ should be to drag $\mathbf{q}(\mathbf{x}_o)$ towards $\mathbf{p}^*(\mathbf{x}_o)$. This is the "unknown force" we observed earlier.

The other force affecting $\mathbf{q}(\mathbf{x}_o)$ during training is, of course, the update based on $\mathbf{x}_o$, driving $\mathbf{q}(\mathbf{x}_o)$ towards $\mathbf{e}_{y_o}$. How do these two forces balance over the course training? Early on, $\|\mathbf{e}_{y_u} - \mathbf{q}^t(\mathbf{x}_u)\|_2$ is relatively large for nearly any $\mathbf{x}_u$, because near initialization $\mathbf{q}^t(\mathbf{x}_u)$ will be relatively flat. Hence the size of the updates for "similar" $\mathbf{x}_u$ and the update for $\mathbf{x}_o$ should be comparable, meaning that, if there are at least a few "similar" training points, $\mathbf{q}^t(\mathbf{x}_o)$ will move towards $\mathbf{p}^*(\mathbf{x}_o)$. Throughout training, some of these similar samples will become well-classified, so that $\|\mathbf{e}_{y_u} - \mathbf{q}^t(\mathbf{x}_u)\|_2$ becomes small, and their updates will no longer exert much force on $\mathbf{q}(\mathbf{x}_o)$. Thus, the $\mathbf{x}_o$ updates begin to dominate, causing the zig-zag pattern as the learning path turns towards e_{y_o}.

Figure 4: Update of $\mathbf{x}_o$ in different epochs.

For easy samples, where $\mathbf{p}^*$ and $\mathbf{e}_{y_o}$ are in the same direction, these forces agree and lead to fast convergence. On samples like the "medium" point in Figure 3, the two forces broadly agree early on, but the learning path deviates slightly towards $\mathbf{p}^*$ en route to $\mathbf{e}_{y_o}$.

Figure 5: Learning path on CIFAR-10. In the first two panels we record $\mathbf{q}^t$ for each batch, while in the last panel we record it for each epoch. See Appendix C for the learning paths of more samples.

4   LEARNING PATHS ON REAL TASKS

Our analysis in Sections 2 and 3 demonstrate that the model can spontaneously refine the bad labels during one-hot training: the zig-zag learning path first moves towards the unknown $\mathbf{p}^*$. But is this also true on real data, for more complex network architectures? We will now show that they do, although seeing the patterns requires a little more subtlety. We then propose an algorithm called Filter-KD to take further advantage of these observations for improved generalization.

To verify this explanation on real tasks – we use CIFAR10 and CIFAR100 as examples – the first conundrum is that we can no longer calculate the base difficulty $\|\mathbf{e}_y - \mathbf{p}^*(\mathbf{x})\|_2$ of a sample, because $\mathbf{p}^*$ is unknown.[8] Furthermore, as the labels of CIFAR dataset are relatively clean, we cannot reliably find many samples with bad labels to see whether the pattern is representative. Recall that there are two cases for a label to be bad: ambiguous sample with flat $\mathbf{p}^*$, and unambiguous samples with "wrong" labels. It might be more persuasive to see that the model can spontaneously "correct" those wrong labels by early stopping at the inflection of the zig-zagging path. Thus, we choose a random 1,000 of the 50,000 training samples, and flip their labels to a random $y' \neq y$.

With these changes, we visualize the learning path of data points while training a ResNet-18 (He et al., 2016) on CIFAR10 for 200 epochs as an example. The first panel of Figure 5 shows the learning path of an easy sample with good label. We can see that $\mathbf{q}^t$ converges quickly towards the left corner, the one-hot distribution for the correct class, because the color of the scattered points in that figure are quite light. At the early stopping epoch, $\mathbf{q}^t$ has already converged to $\mathbf{e}_y$. However, for a sample with a bad label, it is very difficult to observe any patterns from the raw path (blue points): there are points almost everywhere in this plot. This is likely caused by the more complex network and dataset, as well as high learning rate in early training. To find the hidden pattern in this high-variance path, we treat $\mathbf{q}^t$ as a time series signal with many high-frequency components. Thus we can collect its low-frequency components via a low-pass filter and then draw it, i.e., taking a moving average of the $\mathbf{q}^t$ (red points). This makes it clear that, overall, the pattern zig-zags, first moving towards the unknown true label before eventually turning to memorize the wrong label.

This smoothing method not only helps us observe the zig-zag pattern, but also improves the predictions. Figure 6 tracks how often the most likely prediction of the network, $\mathrm{argmax}\,\mathbf{q}^t$, recovers the true original label, rather than the provided random label, for the corrupted data points. At the beginning of the model's training, the initialized model randomly guesses labels, getting about 10% of the 1,000 flipped labels correct. During training, the model spontaneously corrects some predictions, as the learning path first moves towards $\mathbf{p}^*$. Eventually, though, the model memorizes all its training labels. Training with the predictions from the 400th epoch corresponds to standard KD (no

Figure 6: Number of label corrections.

corrected points). Early stopping as in ESKD would choose a point with around 700 corrected labels. The smoothed path, though, performs best, with over 800 corrections.

---

[8]We do not believe the human annotations of the aforementioned CIFAR10H (Peterson et al., 2019) are a perfect estimate of $\mathbf{p}^*$; see Appendix D.

5 SMOOTHING AS A METHOD FOR KNOWLEDGE DISTILLATION

Figure 6 gives a clear motivation for a new knowledge distillation method, which we call Filter-KD (Algorithm 1): train the student from the smoothed predictions of the teacher network. Specifically, we maintain a look-up table $\mathbf{q}_{\mathrm{smooth}} \in \mathbb{R}^{N \times K}$ to store a moving average of $\mathbf{q}_t$ for each training sample. Note that in one epoch, each $\mathbf{q}_{\mathrm{smooth}}(\mathbf{x}_n)$ will be updated only once. We check the early stopping criterion with the help of a validation set. Afterwards, the teaching supervision $\mathbf{q}_{\mathrm{smooth}}$ is ready, and we can train a student network under its supervision. This corresponds to using a moving average of the teacher model "in function space," i.e. averaging the outputs of the function over time.

Compared to ESKD, Filter-KD can avoid the extremely high variance of $\mathbf{q}^t$ during training. Note that unlike Figure 5 which plots $\mathbf{q}^t$ after every iteration of training, most practical implementations only consider early-stopping at the end of each epoch. This is equivalent to down-sampling the noisy learning path (as in Figure 5 (c)), further exacerbating the variance of $\mathbf{q}^t$. Thus ESKD will likely select a bad $\mathbf{p}_{\mathrm{tar}}$ for many data points. Filter-KD, by contrast, has much more stable predictions.

We will show the effectiveness of this algorithm shortly, but first we discuss its limitations. Compared to ESKD, the running time of Filter-KD might be slightly increased (but not too much), because we only do a few extra multiplications and summations for each batch. More importantly, compared to the teaching model in ESKD, the Filter-KD requires a teaching table $\mathbf{q}_{\mathrm{smooth}}$, which will require more memory when the dataset is large. One alternative avoiding the need for this table would be to instead take an average "in parameter space," like e.g. the momentum parameters updating as in Ruppert (1988); Szegedy et al. (2015); He et al. (2020). We empirically find that, although this helps the model converge faster, it does not lead to a better teacher network; see Appendix H. Thus, although Filter-KD has clear drawbacks, we hope that our explanations here may lead to better practical algorithms for large datasets in the future.

Input: Dataset $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, where $I_n = \{1, 2, ..., N\}$ is the index for each pair
 # Train the teacher
Initialize a network model, initialize an $N \times K$ matrix called $\mathbf{q}_{\mathrm{smooth}}$
Go through the entire dataset, calculate $\mathbf{q}_{\mathrm{smooth}}[n] = \mathrm{Softmax}(\mathrm{model}(\mathbf{x}_n))$
for $epoch = 1, 2, ..., T$ **do**
    for $n \in \{1, 2, \ldots, N\}$ in random order **do**
        $\hat{p} = \mathrm{Softmax}(\mathrm{model}(\mathbf{x}_n))$
        $\mathbf{q}_{\mathrm{smooth}}[n] = (1 - \alpha) \cdot \mathbf{q}_{\mathrm{smooth}}[n] + \alpha \cdot \hat{p}$
        $loss = \mathrm{CrossEntropy}(\hat{p}, y_n)$
        Backpropagate and update the parameters
    end for
    Check the early stopping criterion; stop training if satisfied
end for
 # Train the student
For each input $\mathbf{x}_n$, set $\mathbf{p}_{\mathrm{tar}} = \mathbf{q}_{\mathrm{smooth}}[n]$; train the network under the supervision of $\mathbf{p}_{\mathrm{tar}}$

Algorithm 1: Filter-KD. α controls the cut-off frequency of low-pass filter (0.05 here).

5.1 QUANTITATIVE RESULTS OF FILTER-KD

We now compare the performance of Filter-KD and other baselines on a real dataset. We focus on self-distillation and a fixed temperature $\tau = 1$ (except Table 2), as we want the only difference among these methods to be $\mathbf{p}_{\mathrm{tar}}$. Thus we can conclude the improvement we achieved purely comes from the refined supervision. Appendix B for discussion and results related to the choice of temperature; $\tau = 1$ is reasonably good for our current setting.

The first task we consider is noisy-label classification, where we train and validate the model on a dataset with some labels randomly flipped. After training, all the models are evaluated on a clean held-out test set. The experiments are conducted on CIFAR (Figure 7) and TinyImageNet (Table 1), under different noise ratios σ: $\sigma = 0.1$ means 10% of the labels are flipped. In Figure 7, an interesting trend can be observed: the enhancement brought by Filter-KD is not significant when σ is too small or too large. That is reasonable because for small σ, the few samples have bad labels,

Figure 7: Test accuracy of different supervisions under different noisy ratio σ. Solid lines are the means while shade region are the standard errors for 3 runs with different random seeds.

Noise σ	0%	5%	10%	20%
OHT	56.95	53.02	52.02	30.52
ESKD	58.61	53.53	52.99	31.55
FilterKD	59.32	56.43	55.51	40.81

Table 1: Results on TinyImageNet dataset.

	Eff→Res	Res→Mob	Res→VGG
Teacher	86.83	77.23	77.98
Student	78.09	72.58	71.04
ESKD	81.16	73.13	72.64
FilterKD	83.03	75.79	74.49

Table 2: Teacher→student, on CIFAR100.

thus the possible enhancement might not be large. When σ is too high, the labels of similar points become less reliable, and the learning path will no longer head as reliably towards $\mathbf{p}^*$. Thus, for very high noise ratios, the performance of Filter-KD decays back towards that of OHT.

Filter-KD also outperforms other methods in both accuracy and calibration on the clean dataset, as illustrated in Table 3. This remains true in the more common KD case when the teacher network is bigger than the student; see results in Table 2. Note that in Table 2, we no longer keep $\tau = 1$, as the baselines are more sensitive to τ when teacher is large (see Appendix B for more discussion); instead we optimize τ for each setting. Here "Effi" is short for EfficientNet-B1 (Tan & Le, 2019), "Res" is short for ResNet18, "Mobi" is short for MobileNetV2 (Sandler et al., 2018), "VGG" is short for VGG-11 (Simonyan & Zisserman, 2015).

	Accuracy				ECE			
	OHT	**KD**	**ESKD**	**FilterKD**	**OHT**	**KD**	**ESKD**	**FilterKD**
CIFAR10	95.34	95.39	95.42	95.63	0.026	0.027	0.027	0.007
CIFAR100	78.07	78.40	78.83	80.09	0.053	0.061	0.067	0.029

Table 3: Quantitative comparison of generalization performance for ResNet18 self-distillation (mean value of 5 runs). Refer to Table 6 for detailed results.

6 RELATED WORK

Human label refinement. Peterson et al. (2019) use the distribution of annotations from multiple annotators to replace the one-hot label in training, and find that doing so enhances both the generalization performance of trained models and their robustness under adversarial attacks. However, this approach is relatively expensive, as it requires more annotation effort than typical dataset creation techniques; several experienced annotators are required to get a good estimate of $\mathbf{p}^*$. It is also unclear in general how well the distribution of decisions by human annotators corresponds with the true variability under the data distribution. Appendix D has experiments based on this dataset.

Label smoothing. Another popular method is label smoothing (Szegedy et al., 2016), discussed in the previous sections. We believe that suitable smoothing can decrease $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ to some extent, but the lower bound of this gap should be large, because label smoothing treats each class and each sample in the same way. Much prior research (e.g. Müller et al., 2019) shows that KD usually outperforms label smoothing as well.

KD and ESKD. Knowledge distillation, by contrast, can provide a different $\mathbf{p}_{\text{tar}}$ for each training sample. KD was first proposed by Hinton et al. (2015) for model compression, under the assumption that it would help to have a more complex teacher extract more information from the dataset. Later, Furlanello et al. (2018); Cho & Hariharan (2019); Zhang et al. (2019); Yuan et al. (2020) found that distillation can bring enhancement even when teacher and student have identical structure. These surprising findings sparked studies of self-distillation; our work applies to both self-distillation and larger-teacher cases.

Another active direction in research about KD is finding explanations for its success, which is often still considered somewhat mysterious. Tang et al. (2020) decompose the "dark knowledge" provided by the teacher into three parts: uniform, intra-class and inter-class. This explanation also overlaps with ours: if we observe the samples with different difficulty in the same class, we are discussing intra-class knowledge; if we observe samples with two semantically similar classes, we then have inter-class knowledge. Among previous work, the most relevant study is that of Menon et al. (2021), who proposes a statistical explanation of KD. Our Hypothesis 1 is suggested by the main claim of that paper, which focuses on the question of *what is a good* $\mathbf{p}_{tar}$. Our work builds off of theirs by helping explain *why this good* $\mathbf{p}_{tar}$ *emerges, and how to find a better one*. Besides, by looking deeper into the learning path of samples with different difficulty, our work might shed more light on KD, anti-noisy learning, supervised learning, overfitting, etc.

Noisy label task. Learning useful information from noisy labels is a long-standing task (Angluin & Laird, 1988; Natarajan et al., 2013). There are various ways to cope with it; for instance, Menon et al. (2019) use gradient clipping, Patrini et al. (2017) use loss correction, Huang et al. (2020) change the supervision during training, and Zhang et al. (2020) take advantage of extra information to achieve good performance. It is possible to combine KD-based methods with traditional ones to further enhance performance; our perspective of learning paths also provides further insight into why KD can help in this setting.

7 DISCUSSION

In this paper, we first claim that better supervision signal, i.e., $\mathbf{p}_{\text{tar}}$ that is closer to $\mathbf{p}^*$, leads to better generalization performance; this is supported by results of Menon et al. (2021) and further empirical suggestions given here. Based on this hypothesis, we explain why LS and KD outperform one-hot training using experiments on a toy Gaussian dataset. To further understand how such better supervision emerges, we directly observe the behavior of samples with different difficulty by projecting them on a 2-D plane. The observed zig-zag pattern is well-explained by considering the tradeoff between two forces, one pushing the prediction to be near that of similar inputs, the other pushing the prediction towards its training label; we give an intuitive account based on Proposition 1 of why this leads to the observed zig-zag pattern.

To apply these findings on real tasks, in which the data and network are more complex, we conduct low-pass filter on the learning path, and propose Filter-KD to further enhance $\mathbf{p}_{\text{tar}}$. Experimental results on various settings (datasets, network structures, and label noise) not only show the advantage of the proposed method as a method for a teacher in knowledge distillation methods, but also help verify our analysis of how generalization and supervision work in training neural network classifiers.

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, 2019.

Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4794–4802, 2019.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *Proceedings of International Conference on Machine Learning*, pp. 1607–1616. PMLR, 2018.

Mengya Gao, Yujun Shen, Quanquan Li, Junjie Yan, Liang Wan, Dahua Lin, Chen Change Loy, and Xiaoou Tang. An embarrassingly simple approach for knowledge distillation. *arXiv preprint arXiv:1812.01819*, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

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

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

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

Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. *Advances in Neural Information Processing Systems*, 33, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. Characterizing structural regularities of labeled data in overparameterized models. *Proceedings of International Conference on Machine Learning*, 2021.

Taehyeon Kim, Jaehoon Oh, Nakyil Kim, Sangwook Cho, and Se-Young Yun. Understanding knowledge distillation. *https://openreview.net/forum?id=tcjMxpMJc95*, 2019.

Aditya K Menon, Ankit Singh Rawat, Sashank Reddi, Seungyeon Kim, and Sanjiv Kumar. A statistical perspective on distillation. In *International Conference on Machine Learning*, pp. 7632–7642. PMLR, 2021.

Aditya Krishna Menon, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Can gradient clipping mitigate label noise? In *International Conference on Learning Representations*, 2019.

Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26:1196–1204, 2013.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1944–1952, 2017.

Joshua C Peterson, Ruairidh M Battleday, Thomas L Griffiths, and Olga Russakovsky. Human uncertainty makes classification more robust. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9617–9626, 2019.

Karsten Roth, Timo Milbich, Björn Ommer, Joseph Paul Cohen, and Marzyeh Ghassemi. S2sd: Simultaneous similarity-based self-distillation for deep metric learning. *Proceedings of International Conference on Machine Learning*, 2021.

David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Proceedings of International Conference on Learning Representations*, 2015.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

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

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

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

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3903–3911, 2020.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *Proceedings of International Conference on Learning Representations*, 2017.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *ICCV*, 2019.

Zhilu Zhang and Mert R Sabuncu. Self-distillation as instance-specific label smoothing. *Advances in neural information processing systems*, 2020.

Zizhao Zhang, Han Zhang, Sercan O Arik, Honglak Lee, and Tomas Pfister. Distilling effective supervision from severe label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9294–9303, 2020.

A   TOY GAUSSIAN DATASET

Figure 8: Correlation between test accuracy and $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ for different settings.

Figure 9: Correlation between test ECE and $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ for different settings.

In Sections 2 and 3, we apply a toy Gaussian dataset to verify two facts derived from Hypothesis 1, so as the learning path of specific samples. Here we provide more details about this dataset.

Generate the dataset: here, we have N samples, each sample a 3-tuple $(\mathbf{x}, y, \mathbf{p}^*)$. To get one sample, we first select the label $y = k$ following an uniform distribution over all K classes. After that, we sample the input signal $\mathbf{x}|_{y=k} \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma^2 I)$, where σ is the noisy level for all the samples. $\boldsymbol{\mu}_k$ is the mean vector for all the samples in class k. Each $\boldsymbol{\mu}_k$ is a 30-dim vector, in which each dimension is randomly selected from $\{-\delta_\mu, 0, \delta_\mu\}$. Such a process is similar to selecting 30 different features for each class. Finally, we calculate the true Bayesian probability of this sample, i.e., $p^*(y|\mathbf{x})$.

Calculate the ground truth probability: we use the fact that $p^*(y|\mathbf{x}) \propto p(\mathbf{x}|y)p(y)$. As y follows an uniform distribution, we have $p^*(y|\mathbf{x}) = \frac{p(\mathbf{x}|y=k)}{\sum_{j \neq k} p(\mathbf{x}|y=j)}$. Following $p(\mathbf{x}|y = k) \sim \mathcal{N}(\boldsymbol{\mu}_k, \sigma^2 I)$, we find $p^*(y|\mathbf{x})$ should have a Softmax form, i.e., $p = \frac{e^{s_k}}{\sum_{j \neq k} e^{s_j}}$. Specifically, we have:

$$p^*(y = k|\mathbf{x}) = \frac{e^{s_k}}{\sum_{j \neq k} e^{s_j}}; \quad s_i = -\frac{1}{2\sigma^2}\|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2. \tag{5}$$

Setup of experiments in Figure 1: in this experiment, we generate a toy Gaussian dataset with $K = 3$, $\sigma = 2$ and $N = 10^5$. To reduce the variance of test error, we make a train/valid/test split with ratio [0.05 0.05, 0.9]. We train an MLP with 3 hidden layers, each with 128 hidden units and ReLU activations. We first conduct experiments on some baseline settings, i.e., learning from one-hot supervision (OHT for short), from smoothed label supervision (LS for short), from a converged teacher's prediction (KD for short), and from an early-stopped teacher's prediction (ESKD for short). In OHT case, an NN is trained under the supervision of $\mathbf{e}_y$. If we train this NN until the convergence of training accuracy, we obtain the $\mathbf{p}_{\text{tar}}$ for the KD case. If we select the snapshot of that NN based on the best validation accuracy, we obtain the $\mathbf{p}_{\text{tar}}$ for the ESKD case. If we directly set $\mathbf{p}_{\text{tar}} = 0.9\mathbf{e}_y + 0.1\mathbf{u}$, where $\mathbf{u}$ is an uniform distribution over K classes, we obtain the supervision for LS case. With different supervisions, we train a new network with identical structure until the validation accuracy no longer increase. Furthermore, to see a trend between generalization ability and $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$, we run the experiment 200 times under different noisy supervisions.

B   TEMPERATURE IN DIFFERENT KD METHODS

Temperature is an important hyper-parameter in different kinds of KD methods, which might influence the performance a lot. In this appendix, we will discuss why we prefer $\tau = 1$, and why this is a reasonably good empirical choice in our settings.

	Run1				Run2				Run3			
	τ=0.5	τ=1	τ=2	τ=10	τ=0.5	τ=1	τ=2	τ=10	τ=0.5	τ=1	τ=2	τ=10
OHT	94.87	-	-	-	95.15	-	-	-	94.48	-	-	-
ESKD	95.02	95.27	**95.34**	95.05	95.41	**95.41**	95.39	94.76	94.11	94.65	**94.88**	94.75
FilterKD	95.18	**95.87**	95.66	94.81	95.11	95.56	**95.71**	94.81	95.09	**95.31**	95.22	95.16
OHT	77.85	-	-	-	78.23	-	-	-	77.99	-	-	-
ESKD	78.28	78.56	**79.28**	79.09	78.50	78.20	**79.3**	78.81	78.12	78.31	**78.84**	78.36
FilterKD	78.43	79.57	**79.72**	79.65	79.23	**79.61**	79.45	79.01	79.75	80.32	**80.41**	79.99

Table 4: The influence of temperature in self-distillation on CIFAR10/100 dataset.

Figure 10: The influence of label smoothing factor on LS (first row) and different τ on KD (other rows) under two different settings of the toy dataset. It is clear that these hyper-parameters won't influence the performance too much as long as they are in a good region.

The role played by τ:

In general, the loss function in KD has the form:

$$L = \beta \cdot \left(\frac{1}{\tau^2} \right) \cdot \mathcal{H}(\mathbf{q}^\tau, \mathbf{p}_{\text{tar}}^\tau) + (1 - \beta) \cdot \mathcal{H}(\mathbf{q}, \mathbf{e}_y), \tag{6}$$

where $\beta \in [0, 1]$ is another hyper-parameter to trade-off the importance between one-hot label and teacher's predictions. Furthermore, for this loss, the gradient of L to logits $\mathbf{z}$ is:

$$\frac{\partial L}{\partial \mathbf{z}_i} = \beta \cdot \left(\frac{1}{\tau} \right) \cdot (\mathbf{q}_i^\tau - \mathbf{p}_{\text{tar}_i}^\tau) + (1 - \beta) \cdot (\mathbf{q}_i - \mathbf{e}_{y_i}). \tag{7}$$

Why we use $\tau = 1$:

The most important reason for us to choose $\tau = 1$ (as well as $\beta = 1$), even with the risk of providing sub-optimal performance, is that we want to observe how much enhancement is brought by refining the label. In our mind, KD's success comes from the following two aspects:

    (a) better label (i.e., $\mathbf{p}_{\text{tar}}$ is better than $\mathbf{e}_y$);
    (b) better learning dynamics (soften $\mathbf{q}$ to $\mathbf{q}^\tau$ make the training easier).

The first one provides the student with more useful knowledge, and the second one helps the student to extract it. The focus of our paper is the improvement in labels. When the temperature is not 1, both (a) and (b) are influenced, as we are using $\mathbf{q}^\tau$ (i.e., the smoothed student output) to match $\mathbf{p}_{\text{tar}}^\tau$ (i.e., the smoothed target) during training and using $\mathbf{q}$ to inference during testing. If we fix $\tau = 1$, the only difference between Filter-KD, ESKD, KD, LS, and OHT training is $\mathbf{p}_{\text{tar}}$. Under such a condition, we believe the fact that Filter-KD/ESKD outperforms OHT (strictly better in each run) is enough to conclude KD methods can provide better labels. Furthermore, as Filter-KD is proposed to mitigate the high-variance issue in ESKD, and we can directly observe the zig-zag learning path of samples with bad labels, we believe it is reasonable to make the conclusion in the paper.

Furthermore, using $\tau \neq 1$ doesn't substantially change our analysis. As illustrated in Equation (7), the gradient shows that $\mathbf{q}^{\tau}$ moves towards $\mathbf{p}_{\text{tar}}^{\tau}$ in each update. Hence our analysis in proposition 1 still holds, which means the trade-off between the two forces still exists. But considering $\tau \neq 1$ makes the explanation awkward because the prediction of the model is $\mathbf{q}$, not $\mathbf{q}^{\tau}$: readers might feel confused why we visualize $\mathbf{q}$ and $\mathbf{p}_{\text{tar}}$ in the learning path but in fact, the gradient comes from $\mathbf{q}^{\tau}$ and $\mathbf{p}_{\text{tar}}^{\tau}$. If we directly visualize $\mathbf{q}^{\tau}$ and $\mathbf{p}_{\text{tar}}^{\tau}$, how could we read the value of $\mathbf{q}$ from the figure? So, to make the paper easier to follow, we prefer $\tau = 1$.

$\tau = 1$ is reasonably good in our settings:

In fact, the optimal τ depends on many settings, e.g., teacher's size, optimizer, learning rate (scheduler), and etc: there is still no consensus on what is the best choice of τ for all settings. So for the experiments in the main context, we fix τ and fine-tune other hyper-parameters to achieve good results. However, to verify this choice of τ is not terrible, we performed a coarse grid search on both CIFAR and toy examples. As illustrated in Table 4 and Figure 10, choosing $\tau \in [1, 2]$ seems to be good choice: the performance doesn't significantly decay until τ is too large.

Some readers might also curious about why our results don't match the common notion that the optimal τ should be 4, which is first provided in Hinton et al. (2015) and followed by much later work (Gao et al., 2018; Zagoruyko & Komodakis, 2017; Cho & Hariharan, 2019, e.g.). Some works also use a grid search to conclude that a temperature as high as 20 should be the best choice (Yuan et al., 2020; Kim et al., 2019). We find this mismatch comes from the relative difference between the network size of teacher and student. In short, when distilling from a large teacher to a small student (as most of the cases in aforementioned works), high τ is preferred. When conducting self-distillation, τ need not be that high. For example, Zhang et al. (2019) and Roth et al. (2021) claim $\tau = 1$

Figure 11: First row: large teacher to small student; second row: self-distill. Left column: τ in a larger range; right column: τ in a small range.

is the best choice, while Zhang & Sabuncu (2020) claim $\tau \approx 2$ is the best. To further verify this, we compare the temperature trend between two cases on the toy dataset. In Figure 11, the first row is distilling a 10-layer, 256-width MLP to a 3-layer, 32-width MLP; the second row is self-distillation between 3-layer, 32-width MLPs.

At the same time, we also run a grid search the smoothing factor α of our Filter-KD in Table 5.

Smoothing α	0.01	0.05	0.1	0.2	0.5	1 (ESKD)
FilterKD	79.50	**80.00**	79.83	79.59	78.48	78.39

Table 5: The influence of the smoothing factor in FilterKD on CIFAR100 (mean of 3 runs). For comparison, OHT obtained 77.64.

C HOW REPRESENTATIVE IS THE ZIG-ZAG PATTERN?

In the main paper, we only visualize the zig-zag learning path of a few samples in Figures 3 and 5. The readers might then wonder whether this pattern is representative across the whole dataset. To verify this, we define a quantitative metric called **zig-zag score**. Specifically, we first calculate the integral of each dimension of the prediction:

$$Q_{i \in \{1,...,K\}} \triangleq \sum_{t=1}^{T} \mathbf{q}_i^t(\mathbf{x}_n). \tag{8}$$

We then use the highest Q_i among $i \in \{1, ..., K\} \setminus \{y\}$, where y is the label's class, as the zig-zag score. In other words, we focus on the behavior of $\mathbf{q}$ on those dimensions that are *not* the training label. If this score is large, we might expect the neighbouring samples exert high influence on the

Figure 12: The learning path of samples with correct and wrong labels.

Figure 13: Correlation between base difficulty and zig-zag score in four different toy datasets.

path, and vice versa. Note that as we have $\sum_i Q_i =$ constant for any sample (as $\mathbf{q}^t$ is a K-simplex), this zig-zag score might correlated with C-score of Jiang et al. (2021). However, their focuses are different: C-score is similar to $-Q_{i=y}$ and focuses more on how fast the prediction converge to the label; zig-zag score, on the other hand, is more about how much the path deviates towards a different class, possibly the label close to $\mathbf{p}^*$.

With this score, we test the correlation between base difficulty and zig-zag score on toy datasets under different settings. From Figure 13, it is safe to conclude that samples with higher base difficulty will have a more zig-zagging path during training. We also compare the expected zig-zag score of the 1000 samples with flipped label in CIFAR10 (recall the experiment in Figure 6). In Figure 12, it is clear that the zig-zagness of these wrong label samples (which we are quite have high base difficulty) is significantly higher than the average.

Finally, we also randomly select some data samples in each class of CIFAR10 and visualize their learning paths. Figure 14 shows random samples for the noisy label experiment. It is clear that the learning path of easy samples with correct labels converge fast while that of samples with wrong labels is zig-zag. In Figure 15, which shows the learning path with high zig-zag score when the training set is clean, we can still observe some samples with zig-zag path. These samples might be ambiguous, with a quite flat $\mathbf{p}^*$. However, as we do not know the true $\mathbf{p}^*$ of these samples, it is impossible to provide a result like Figure 12.

D   DISCUSSION OF CIFAR10H

As previously mentioned, Peterson et al. (2019) propose a CIFAR10H dataset to refine the one-hot label by human annotators. However, we decided not to use this baseline in the main body of the paper. We have the following three reasons:

- CIFAR10H only annotate the test set of CIFAR10, which make it hard to compare with other standard baselines;
- We cannot control the difficulty of the task like what we do for the toy dataset;
- We believe the labels provided by human annotators is not a perfect estimation of $\mathbf{p}^*$.

The first two reasons are straightforward, we explain why we don't believe the labels in CIFAR10H is the precise $\mathbf{p}^*$. From our definition, $\mathbf{p}^* \triangleq p^*(y|\mathbf{x})$, which is related to the generating process of the image. Although Peterson et al. (2019) show that training using this label can bring enhancement

(a) Samples with clean labels.

(b) Samples with wrong labels.

Figure 14: Random selection of samples in CIFAR10 with 1000 flipped labels.

Figure 15: Random selection of samples with high zig-zag score in clean CIFAR10.

Figure 16: Learn from labels in CIFAR10H and its noisy versions.

and hence make the prediction more robust, we still believe that there is a bias of human's cognitive system in it. In short, labels from CIFAR10H is a good estimation that is far better than one-hot, but not perfect. To verify it, we conduct the following two experiments.

First, we add noise to the labels of CIFAR10H and use them to train a student network. We then evaluate these models on the original CIFAR10 training set (Peterson et al. (2019) also applies this strategy to make an evaluation). The results are provided in the first two panels of Figure 16, which is similar to what we do for the toy dataset in Figure 8. For the first panel, when the added noise are high enough to flip some labels, we see a clear negative correlation between $\|\mathbf{p}_{\text{tar}} - \mathbf{p}_{\text{cifarh}}\|_2$ and test accuracy. However, when we add small amount of noise to $\mathbf{p}_{\text{cifarh}}$, as in the second panel in the figure, the test accuracy will increase. This means $\mathbf{p}_{\text{cifarh}}$ might be a little conservative when estimating $\mathbf{p}^*$: adding noise and hence making $\mathbf{p}_{\text{tar}}$ smoother can slightly enhance the generalization ability. However, in the toy dataset where $\mathbf{p}^*$ is precisely the generating probability, learning from $\mathbf{p}^*$ always performs the best.

Figure 17: How to project a probability vector on to the plane of Barycentric coordinate system.

Furthermore, we also compare the supervisions provided by Fitler-KD and $\mathbf{p}_{\text{cifarh}}$. As illustrated in the last panel in Figure 16, Filter-KD can achieve higher accuracy than CIFAR10H case, which we believe is another evidence for our concern.

E SOME BACKGROUNDS

Calculation of ECE: expected calibration error is a measurement about how well the predicted confidence represent the true correctness likelihood (Guo et al., 2017). For example, if our model gives 100 predictions, each with confidence, say, $q(y = k|\mathbf{x}) \in [0.7, 0.8]$). Then we might expect there are $70 \sim 80$ correct predictions among those 100 ones. To calculate this, we first uniformly divide $[0, 1]$ into M bins, with each bin represented by $I_m \in \left(\frac{m-1}{M}, \frac{m}{M}\right]$ and B_m is all the $\mathbf{x}$ samples whose confidence falls into I_m. Then, ECE is calculated as:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| acc(B_m) - conf(B_m) \right|, \tag{9}$$

where $acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(\hat{y}_i = k_i)$, $conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} q(y = k_i|\mathbf{x}_i)$, k_i is the true label and $\hat{y}_i = argmax[q(y|\mathbf{x}_i)]$ is the model's prediction. All the ECE mentioned in this paper is calculated by setting $M = 10$.

Barycentric coordinate system: when visualizing the learning path, it is problematic to directly choose two dimensions and draw them in the Cartesian coordinate system, as illustrated in the first panel in Figure 17. In geometry, a suitable way to project a 3-simplex vector onto a 2-D plane is converting it to a point in Barycentric coordinate system. Specifically, we have three basis vectors: $\mathbf{v}_0 = [0, 0]^\mathsf{T}$, $\mathbf{v}_1 = [1, 0]^\mathsf{T}$, and $\mathbf{v}_2 = \left[\frac{1}{2}, \frac{\sqrt{3}}{2}\right]^\mathsf{T}$, which are the corner and two edges of an equilateral triangle respectively. Then the 2D-coordinate is calculated as $(x, y) = [\mathbf{v}_0; \mathbf{v}_1; \mathbf{v}_2] \cdot \mathbf{q}$. So every points in the left corner of a Cartesian system plane can be converted to the triangle in Barycentric system, as illustrated in the last panel in Figure 17.

F DISTANCE GAP UNDER DIFFERENT SUPERVISIONS

Figure 2 provides the distance gap between $\mathbf{q}$ and corresponding $\mathbf{p}^*$ for each training sample in different training stages, under the supervision of one-hot label $\mathbf{e}_y$. From the results in this figure, we notice that the behavior of hard samples contributes more to the success of ESKD. Here we further visualize how these gaps changes when the model is trained under different types of supervision, i.e., ground truth, LS and ESKD.

In the first row, which is the result of LS training, we see the $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ (i.e., red dots) has a 'V'-shape. The vertex of it depends on the smoothing parameter we choose. However, as we claimed in the section 3.1 and section 6, the LS method has only one parameter to control $\mathbf{p}_{\text{tar}}$, which is like using a linear model to fit a high-order function. So, although a proper smoothing value can

bring better supervision than one-hot label, the upper bound of it might be limited. Regarding the training dynamics, we can see a similar trend as the results shown in the one-hot case, i.e., all the samples first move down and then converge to $\mathbf{p}_{\text{tar}}$. From the middle panel, we might expect the LS training outperforms the one-hot case, because the scatters are closer to the x-axis, which represents the ground truth $\mathbf{p}^*$.

The second and the third rows demonstrate the KD and ESKD case. Results in KD case is quite similar to the one-hot case in Figure 2, because the converged $\mathbf{p}_{\text{tar}}$ is close to $\mathbf{e}_y$. However, we can still expect KD outperforms one-hot training, because the $\mathbf{p}_{\text{tar}}$ is closer to $\mathbf{p}^*$ than $\mathbf{e}_y$ does. The last panel in this row also demonstrates that $\|\mathbf{q} - \mathbf{p}^*\|_2$ might be smaller than $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$, which can be considered as an explanation for why iterated self-distillation, e.g., Born Again Network proposed in (Furlanello et al., 2018), can improve the performance. For the ESKD case, we see the overfitting almost disappear: the distribution of the blue points do not change too much after the early stopping criterion is satisfied.

Figure 18: Distance gap of each sample under different supervisions.

In the last row, which illustrates the training under the supervision of $\mathbf{p}^*$, we find all the blue points move toward the x-axis, i.e., their $\mathbf{p}_{\text{tar}} = \mathbf{p}^*$, and finally converge to it. There is also no overfitting in this case. From the last two panels, we see the disperse of blue points is significantly smaller than all other settings, which means the network's prediction is quite close to $\mathbf{p}^*$. Hence the performance of this case is the best.

G   MORE ABOUT THE DECOMPOSITION AND NTK MODEL

Proof of Proposition 1: to observe how $\mathbf{q}^t(\mathbf{x}_o)$ changes when the network's parameters update after observing $\mathbf{x}_u$, we can directly write down $\Delta^t(\mathbf{x}_o) = \mathbf{q}^{t+1}(\mathbf{x}_o) - \mathbf{q}^t(\mathbf{x}_o)$ and simplify it (recall $\mathbf{z}(\mathbf{x}) = f(\mathbf{w}, \mathbf{x})$ is the logits of NN and $\mathbf{q} = \text{Softmax}(\mathbf{z})$ is the output probability):

$$\Delta^t(\mathbf{x}_o) = \nabla_{\mathbf{w}} \mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \left(\mathbf{w}^{t+1} - \mathbf{w}^t\right) + \mathcal{O}\left(\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2\right) \tag{10}$$

$$= \left(\nabla_{\mathbf{z}} \mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \nabla_{\mathbf{w}} \mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t}\right) \cdot \left(\mathbf{w}^{t+1} - \mathbf{w}^t\right) + \mathcal{O}(\epsilon^2) \tag{11}$$

$$= \left(\nabla_{\mathbf{z}} \mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \nabla_{\mathbf{w}} \mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t}\right) \cdot \left(-\eta \nabla_{\mathbf{w}} L(\mathbf{x}_u)|_{\mathbf{w}^t}\right)^{\mathsf{T}} + \mathcal{O}(\eta^2) \tag{12}$$

$$= \left(\nabla_{\mathbf{z}} \mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \nabla_{\mathbf{w}} \mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t}\right) \cdot \left(-\eta \nabla_{\mathbf{z}} L(\mathbf{x}_u)|_{\mathbf{w}^t} \cdot \nabla_{\mathbf{w}} \mathbf{z}(\mathbf{x}_u)|_{\mathbf{w}^t}\right)^{\mathsf{T}} + \mathcal{O}(\eta^2) \tag{13}$$

$$= -\eta \nabla_{\mathbf{z}} \mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \left[\nabla_{\mathbf{w}} \mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \left(\nabla_{\mathbf{w}} \mathbf{z}(\mathbf{x}_u)|_{\mathbf{w}^t}\right)^{\mathsf{T}}\right] \cdot \left(\nabla_{\mathbf{z}} L(\mathbf{x}_u)|_{\mathbf{w}^t}\right)^{\mathsf{T}} + \mathcal{O}(\eta^2) \tag{14}$$

$$= \eta \cdot \mathcal{A}^t(\mathbf{x}_o) \cdot \mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u) \cdot \left(\mathbf{p}_{\text{tar}}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u)\right) + \mathcal{O}(\eta^2), \tag{15}$$

where $\epsilon^2 \triangleq \|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 = \eta^2 \cdot \|\nabla_{\mathbf{w}} L\|^2$. Here we have $\|\nabla_{\mathbf{w}} L\|^2 \leq \|\nabla_{\mathbf{z}} L\|^2 \cdot \|\nabla_{\mathbf{w}} \mathbf{z}\|^2$. In the right hand set, $\|\nabla_{\mathbf{z}} L\| = \|\mathbf{p}_{\text{tar}} - \mathbf{q}^t\|$ is bounded because they are probabilities. The term $\|\nabla_{\mathbf{w}} \mathbf{z}\|$ is also bounded if we use gradient clipping during training. Thus in most of the case, we have $\mathcal{O}(\epsilon^2) = \mathcal{O}(\eta^2)$, which can be ignored when the learning rate η is sufficiently small.

Figure 19: Upper panels: how $tr(\mathcal{A}^t(\mathbf{x}_n))$ changes during training. Each panel represents a specific $\mathbf{x}_n$. The panels are ordered by their integral difficulty, from left-to-right and up-to-down. The x-axis is the number of epoch and the y-axis is $tr(\mathcal{A}^t(\mathbf{x}_n))$. Lower panels: the correlation between Cosine($\mathbf{x}_o, \mathbf{x}_u$) and their $tr(\mathcal{K}^0)$. Panels are ordered by the integral difficulty of $\mathbf{x}_o$. The subtitle of the panel claims the sample ID, the class it belongs to (i.e., A, B, or C) and the color of their corresponding class (i.e., A is blue, B is orange and C is green).

The first line is a Taylor expansion, where $\Delta^t(\mathbf{x}_o)$ is a $K \times 1$ vector, $\nabla_{\mathbf{w}}\mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t}$ is a $K \times d$ matrix, and $\mathbf{w}$ is a $d \times 1$ vector (d is the number of parameters). The second line follows the chain rule, where $\nabla_{\mathbf{z}}\mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t}$ is a $K \times K$ matrix and $\nabla_{\mathbf{w}}\mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t}$ is a $K \times d$ matrix. The third line applies the definition of SGD, where $\nabla_{\mathbf{w}}L(\mathbf{x}_u)|_{\mathbf{w}^t}$ is a $1 \times d$ vector. The fourth line also follows the chain rule, where $\nabla_{\mathbf{z}}L(\mathbf{x}_u)|_{\mathbf{w}^t}$ is a $1 \times K$ vector and $\nabla_{\mathbf{w}}\mathbf{z}(\mathbf{x}_u)|_{\mathbf{w}^t}$ is a $K \times d$ matrix. The fifth and the sixth lines are rearrangments of different terms, where both $\mathcal{A}^t$ and $\mathcal{K}^t$ are $K \times K$ matrixes. In this expression,

$$\mathcal{A}^t(\mathbf{x}_o) = \begin{bmatrix} q_1(1-q_1) & -q_1q_2 & \cdots & -q_1q_K \\ -q_2q_1 & q_2(1-q_2) & & \\ \vdots & & \ddots & \\ -q_Kq_1 & & & q_K(1-q_K) \end{bmatrix}, \tag{16}$$

which is a symmetric positive semi-definite (PSD) matrix[9] with trace $tr(\mathcal{A}^t(\mathbf{x}_o)) = 1 - \sum_{i=1}^{K} q_i^2$. As we have $\sum_i q_i = 1$, it is easy to check the trace of this matrix is larger at the beginning of

---

[9]The matrix $\mathcal{A}$ can be observed to be the covariance matrix of a categorical distribution with item probabilities q, and hence PSD.

training (when $\mathbf{q}$ tend to be flat) than that at the end of training ($\mathbf{q}$ tend to be peaky), as illustrated by most panels in Figure 19. Given that the trace of a matrix equals the summation of its eigenvalues and $\mathcal{A}^t(\mathbf{x}_o)$ is PSD, smaller $tr(\mathcal{A}^t(\mathbf{x}_o)) = 1 - \sum_{i=1}^{K} q_i^2$ means this matrix will shrink the vector it multiplied more. Hence the step size of $\Delta^t(\mathbf{x}_o)$ tend to decrease when $\mathbf{q}^t$ becomes more peaky.

The second term in that expression, i.e., $\mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u)$ is the inner product of gradients at $\mathbf{x}_o$ and $\mathbf{x}_u$. Intuitively, if their gradients has similar directions, this term might be large, and vice versa. As this term also depends on t, it is hard to analyze its property when there are two influencing factors. So following the results about NTK, which claims that an infinitely wide NN can be approximated by a linear kernel model, we study $\mathcal{K}^0$ instead of $\mathcal{K}^t$ to focus more on the influence of $\mathbf{x}_u$.

Neural Tangent Kernel: researchers find that the weights of a NN tend to be 'lazy' during training (Jacot et al., 2018) when the NN is wide enough and the learning rate is sufficiently small. In other words, the offset of each dimension of $\mathbf{w}^0$ and $\mathbf{w}^T$ is small. Hence, it is reasonable to use first-order Taylor expansion on $\mathbf{w}^0$ to approximate the function indicated by the model:

$$f(\mathbf{w}, \mathbf{x}) \approx f(\mathbf{w}^0, \mathbf{x}) + \left(\nabla_{\mathbf{w}} f(\mathbf{w}^0, \mathbf{x})\right)^\top \cdot (\mathbf{w} - \mathbf{w}^0) \tag{17}$$

$$= \left(\nabla_{\mathbf{w}} f(\mathbf{w}^0, \mathbf{x})\right)^\top \cdot \mathbf{w} + f(\mathbf{w}^0, \mathbf{x}) - \left(\nabla_{\mathbf{w}} f(\mathbf{w}^0, \mathbf{x})\right)^\top \cdot \mathbf{w}^0 \tag{18}$$

$$= \phi(\mathbf{x})^\top \mathbf{w} + \mathbf{b}, \tag{19}$$

where we use $\phi(\mathbf{x})^\top$ and $\mathbf{b}$ to simplify the notation. Then, a 1-layer infinitely wide NN can be approximated by the aforementioned linear model, where $\phi(\mathbf{x})$ plays a similar role as the design matrix in a linear regression model. The NTK of that model is defined as $\mathcal{K}^0(\mathbf{x}_i, \mathbf{x}_j) \triangleq \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, which is useful when analyzing the learning dynamics of the model.

Learning Dynamics of $\mathbf{q}(\mathbf{x}_o)$: in proposition 1, we decompose $\Delta^t(\mathbf{x}_o)$ into three parts and analyze what the learning path of a training sample might be like in section 3.3. Here we will provide more detailed illustration of the two groups of force imposed on $\mathbf{q}(\mathbf{x}_o)$.

Specifically, $\Delta^t(\mathbf{x}_o)$ will be influenced by two variables, i.e., updating sample $\mathbf{x}_u$ and time t. We discuss their effects separately. For $\mathbf{x}_u$, only the last two terms, i.e., $\mathcal{K}^0(\mathbf{x}_o, \mathbf{x}_u)$ and $(\mathbf{p}_{\text{tar}}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u))$ depends on it. In section 3.3, we claim that if $\mathbf{x}_o$ and $\mathbf{x}_u$ are similar, the norm of $\mathcal{K}^0(\mathbf{x}_o, \mathbf{x}_u)$ might be large, and vice versa. Here we emperically verify this using a toy Gaussian dataset, as illustrated in Figure 19. The figure shows how the similarity between $\mathbf{x}_o$ and $\mathbf{x}_u$ correlates with $tr(\mathcal{K}^0(\mathbf{x}_o, \mathbf{x}_u))$. Each panel represents one $\mathbf{x}_o$, which is claimed in the title of the subfigures. The x-axis represents the rank of the cosine similarity between observed $\mathbf{x}_o$ and all N training samples (including itself). The y-axis is the trace of $\mathcal{K}^0(\mathbf{x}_o, \mathbf{x}_u)$. The color of each scatter point is the true label of $\mathbf{x}_u$. From the figure, we can observe a clear decreasing trend, which means smaller similarity leads to larger $tr(\mathcal{K}^0(\mathbf{x}_o, \mathbf{x}_u))$. Additionally, the term $(\mathbf{p}_{\text{tar}}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u))$ provides a direction that $\mathbf{q}^t(\mathbf{x}_o)$ should move towards. We claim in section 3.3 that at any point in the input space, the labels of these input samples might follow the ground truth distribution, i.e., $p^*(y|\mathbf{x})$. Hence most of the neighbouring $\mathbf{x}$ might pull $\mathbf{q}(\mathbf{x}_o)$ towards its ground truth $p^*(y|\mathbf{x}_o)$. We will discuss the norm of this term when discussing the influence of t. In short, at any time, the neighbouring $\mathbf{x}_u$ will impose stronger force on $\mathbf{x}_o$ and the direction of the force roughly points to the ground truth $p^*(y|\mathbf{x}_o)$.

Another factor to consider is t, which influence $\Delta^t(\mathbf{x}_o)$ through the term $\mathcal{A}^t(\mathbf{x}_o)$ and term $(\mathbf{p}_{\text{tar}}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u))$. However, these two terms do not have an analytical expression w.r.t t, as $\mathbf{q}^t$ is quite complicated. So we can only provide some intuitions through the behavior of $\mathbf{q}^t$ and then experimentally verify them. In Figure 19, we show how $tr(\mathcal{A}^t(\mathbf{x}_o))$ depends on $\mathbf{q}^t$: flat $\mathbf{q}^t$ leads to larger trace. A similar trend also exists in the norm of $(\mathbf{p}_{\text{tar}}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u))$. As the initialized $\mathbf{q}$ tend to be flat, updates of any samples will influence network's parameters a lot. When the training progress, those easy samples converge fast, so their $\|\mathbf{p}_{\text{tar}}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u)\|_2$ and $tr(\mathcal{A}^t)$ become small. However, as the $\mathbf{q}^t$ for the hard sample is still far away from its $\mathbf{e}_y$, the large $\|\mathbf{p}_{\text{tar}}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u)\|_2$ and $tr(\mathcal{A}^t)$ will finally drag $\mathbf{q}^t$ back towards the one-hot distribution, as illustrated in Figure 3.

H LOW PASS FILTER ON NETWORK PARAMETERS

In Section 4, we point out the high variance issue of the traditional KD methods after observing the learning path of those hard samples. We then propose a Filter-KD algorithm to smooth the output of each training sample during training. Such a low pass filtering method is quite similar to the

Figure 20: Behavior of the tracking model (with low pass filter on each parameters). ResNet18 trained for 150 epochs on CIFAR100.

momentum mechanism used in self-supervised learning, e.g., MOCO in He et al. (2020), which conduct low pass filtering on each parameter of the network. As mentioned before, the proposed Filter-KD algorithm might require more memory when the dataset becomes larger, because $\mathbf{q}_{smooth}$ would record every training sample's prediction.

Conducting low pass filtering on network parameters might be a good way to solve this. To verify whether this method works, we train a ResNet18 on CIFAR100, using one-hot supervision. At the beginning of training, we copy the training model and call it tracking model. At the end of each update (i.e., each batch), the parameters of training model is updated as usual while the tracking model's parameters are updated with momentum. Specifically, for the training model, $\mathbf{w}^{t+1}_{train} = \mathbf{w}^t_{train} + \eta \nabla L$, for the tracking model, $\mathbf{w}^{t+1}_{track} = (1 - \alpha)\mathbf{w}^t_{track} + \alpha \mathbf{w}^{t+1}_{train}$. We train the model for 150 epochs, and show the learning curve of test accuracy in the left panel of Figure 20. We see the performance of the tracking model converges faster than the training model, which is reasonable because filtering parameters is regularizing the training process. However, the converging accuracy of this two models are the same. At the same time, we find this tracking model is not as good as $\mathbf{q}_{smooth}$ when teaching a new model, as illustrated in the right panel of Figure 20.

As this paper mainly focuses on the behavior of the model's prediction, the discussion and experiments of filtering in parameter space is limited. However, as many exciting works demonstrates the effectiveness of this momentum mechanism, we would try to understand it in our future work.

	Test ACC					Test ECE				
	Run1	Run2	Run3	Run4	Run5	Run1	Run2	Run3	Run4	Run5
OHT	95.35	95.30	95.42	95.23	95.42	0.027	0.027	0.026	0.025	0.025
KD	95.30	95.38	95.42	95.44	95.42	0.027	0.027	0.027	0.026	0.026
ESKD	95.29	95.41	95.39	95.58	95.42	0.026	0.029	0.025	0.028	0.027
FilterKD	95.66	95.68	95.49	95.76	95.58	0.005	0.006	0.008	0.011	0.006
OHT	78.27	78.31	77.97	77.78	78.02	0.053	0.053	0.052	0.053	0.054
KD	78.64	78.55	78.03	78.18	78.49	0.060	0.057	0.062	0.066	0.059
ESKD	78.84	78.73	78.85	78.97	78.74	0.065	0.066	0.067	0.070	0.066
FilterKD	79.87	79.93	80.19	80.22	80.23	0.028	0.026	0.034	0.028	0.031

Table 6: Each run of results in Table. 3. In each run, the initialization of student networks for different methods are controlled to be the same.