BETTER SUPERVISORY SIGNALS BY OBSERVING LEARNING PATHS

ABSTRACT

Better supervision might lead to better generalization 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 (KD), in terms of our proposed criterion. To further answer why and how better supervision emerges, we look deeper into the learning path of the network's predicted distribution for training samples with different difficulty. A "zig-zag pattern" for hard samples is the crux of KD's success. Observing the learning path not only provides a new perspective for understanding KD, overfitting, and learning dynamics, but also points out the high variance issue of KD on real tasks. Inspired by this, we propose Filter-KD to further enhance classification performance, as verified by experiments in various settings.

1 INTRODUCTION

In multi-class classification problems, we usually supervise our learning problems by providing "one-hot" labels ($y_i = 1$ 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. One approach is to summarize a distribution of human annotations, as was done on the CIFAR-10 dataset by Peterson et al. (2019). An alternative approach is label smoothing (henceforth LS; Szegedy et al., 2016), mixing between a one-hot label and the uniform distribution. Knowledge distillation (KD) was originally proposed for model compression (Hinton et al., 2015), and involves first training a teacher network on the training set and then a student network on the teacher's output probabilities; this can also be thought of as refining the supervision signal, providing "soft" outputs from a teacher 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), many based on the idea that the teacher provides useful "dark knowledge" to the student through its labels.

Inspired by these works, 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 KD's success owes to the behavior of hard samples, where $p^*(y \mid \mathbf{x})$ is far from the one-hot vector from the training set: during one-hot training, the model predictions first move towards $p^*(y \mid \mathbf{x})$, and then slowly converges to the one-hot label, following a zig-zag pattern. A well-trained teacher, one that is 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 neural network (NN) training, including in an analytical case in the neural tangent kernel (NTK) regime (Jacot et al., 2018). Based on our explanations, we point out that this signal can be better achieved if we take a moving average of the teacher network, an algorithm we term Filter-KD, which often provides better supervision, especially under label noise.

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 $f : \mathcal{X} \to \mathbb{R}^K$ 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[1] 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

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

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 vector 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 one image $\mathbf{x}$ 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 $\mathbf{e}_{y_n}$. 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 achieving this, using multiple human annotators to estimate $\mathbf{p}_{\mathrm{tar}}$. Szegedy et al. (2016) use label smoothing, setting $\mathbf{p}_{\mathrm{tar}}$ to a convex combination of $\mathbf{e}_y$ and the constant vector $\frac{1}{K}\mathbf{1}$. In knowledge distillation[2] (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}.

2.2 MEASURING THE QUALITY OF SUPERVISION

Choosing a different $\mathbf{p}_{\mathrm{tar}}$, then, can lead to a better final model. Can we characterize which $\mathbf{p}_{\mathrm{tar}}$ 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 $\mathbb{E}_{\mathbf{x} \sim \mathcal{D}'}[\|\mathbf{p}_{tar}(\mathbf{x}) - \mathbf{p}^*(\mathbf{x})\|_2]$, will in general lead to better generalization performance.*

---

[1] 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.

[2] In this paper, we only consider self-distillation, when the structure of teacher and student are identical.

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

Figure 1: Experiments on a toy Gaussian 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 (in terms of ACC or ECE). 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.

This hypothesis is suggested by Proposition 3 of Menon et al. (2021), which shows that for any given predictor f,

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

where $c > 0$ is a constant. When N is large, the second term will dominate the right hand side, implying Hypothesis 1. In other words, 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 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 $\mathbf{p}^*$). From Figure 1 (b, c), it is clear that in this regime smaller $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ leads to better generalization performance, as measured either by accuracy (ACC) or expected calibration error (ECE)[3] on a held-out test set.

3 INSIGHTS FROM THE LEARNING PATH

In the toy example of Section 2, we see that early-stopped knowledge distillation (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 label. This section will answer this by observing the learning path of training samples with different difficulty.

3.1 PAY MORE ATTENTION TO HARDER SAMPLES

For a finer-grained understanding of early stopping in knowledge distillation, we would like to better understand how the teacher's representations evolve during 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 also 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 will be high for ambiguous points, but especially points where the sampled y had low probability under $\mathbf{p}^*$.

Figure 2 shows these two quantities at three points in training: initialization, the point where ESKD stops, and convergence. At initialization, most points have large $\|\mathbf{q}(\mathbf{x}) - \mathbf{p}^*(\mathbf{x})\|_2$.[4] By the point of

---

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

[4]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.

Figure 2: Distance gap for different samples. The samples are ordered by their base difficulty (see Section 3.1). The figures show the output distribution, i.e., **q**, during the one-hot training in different stages. In these figures, $\|\mathbf{q}-\mathbf{p}^*\|_2$ can be considered as the average height of each point. We provide more results about the NNs trained under different supervisions using this fashion in Appendix C.

Figure 3: Learning path of three samples (from left to right: easy, medium, and hard). Corners correspond to one-hot vectors for each class. Colors represent training time: transparent at initialization (though that point is also marked with a green diamond), fully blue at the end of training. Also marked: the training label y (red cross), the unobserved true **p*** (gray cross), and the point chosen by ESKD's early stopping rule (red plus). For an explanation of points on the right, see Section 3.3.

early stopping, most **q**(**x**) values are roughly near **p***. At convergence, however, **q**(**x**) $\approx$ **e**$_y$, as the classifier has nearly memorized the training inputs, leading to a diagonal line in the plot.

From these figures, it is clear that 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, then, these are the points we should focus on the most.

3.2 LEARNING PATH OF DIFFERENT SAMPLES

To better understand how **q**(**x**$_n$) changes, we randomly select 100 hard samples (top 5% in terms of base difficulty), 100 medium samples (40%~60%) and 100 easy samples (bottom 5%) as the seed samples. We track the model's outputs on these samples, **q**^{t} where t indexes time, at each training step. As this problem has 3 classes, we directly plot these three-dimensional vectors for three examples, using a barycentric coordinate system (details in Appendix B), in Figure 3.

We only show three points, because they are quite representative of all samples. The easy sample very quickly moves to the correct location near **e**$_y$ (as indicated by the light color until it reaches the corner). The medium sample takes a less direct route, drifting off slightly towards **p***, but still directly approaches **e**$_y$. The hard sample, however, does something very different: it first approaches **p***, but then veers off towards **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 **p***; in both cases, the early stopping point is not quite perfect, but is noticeably closer to **p*** than the final converged point near **e**$_y$. Under Hypothesis 1, this also explains the superior performance of ESKD to KD on the converged model.

3.3 EXPLANATION OF PATTERNS IN THE LEARNING PATH

Where does this "unknown force" pushing **q** towards **p*** come from? The following decomposition, proved in Appendix E, 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 $\mathbf{q}^t(\mathbf{x}) = \mathrm{Softmax}(\mathbf{z}^t(\mathbf{x}))$ 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 on the data point $(\mathbf{x}_u, \mathbf{p}_{tar}(\mathbf{x}_u))$ and learning rate η to $\mathbf{w}^t$. Then*

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

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 each $K \times K$.

The matrix $\mathcal{K}^t$ is known as the *empirical neural tangent kernel*; in general, it changes during training in potentially complex ways, but in very wide networks with appropriate random initialization, it becomes nearly invariant throughout training and nearly independent of the initialization (Jacot et al., 2018; Arora et al., 2019). Thus, we can imagine $\mathcal{K}^t$ as roughly constant. $\mathcal{A}^t$, meanwhile, can be written in terms of $\mathbf{q}(\mathbf{x}_o)$ as $\mathrm{diag}(\mathbf{q}) - \mathbf{q}\mathbf{q}^\mathsf{T}$; its eigenvalues are bounded between 0 and 1. For more details, see Appendix E.

Consider 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 $\mathbf{x}_u \neq \mathbf{x}_o$, and one update based on $\mathbf{x}_o$. These are represented on the right of Figure 3: blue lines for $\mathbf{x}_u$ and red lines for $\mathbf{x}_o$, with a purple dot for the start of the epoch and a purple plus for the end. 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}.

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, but the learning path deviates slightly towards $\mathbf{p}^*$ en route to its converged point.

4   LEARNING PATHS ON REAL TASKS

Our analysis in Sections 2 and 3 explains the success of ESKD: it simply stops when the zig-zag begins to "turn around." But do these patterns still occur on real data, for more complex network architectures and problems? 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, 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. To solve this, we instead use the **integral difficulty**, defined as the total probability the model gives to labels other than the training label y over time: $\frac{1}{T} \sum_{t=1}^{T} (1 - \mathbf{e}_y^\mathsf{T} \mathbf{q}^t(\mathbf{x}))$. Unlike base difficulty, the integral difficulty depends on the network structure, hyper-parameters, the particular training run, and so on. However, we empirically show in Appendix C that the integral difficulty of a particular sample is quite consistent across these choices. We also compare to other potential notions of difficulty, such as those based on the converged confidence $\max(\mathbf{q}^T(\mathbf{x}))$.

Another issue with replicating our earlier analysis is that most classification problems have the number of classes K larger than 3, making visualization harder. We thus choose two class labels to highlight – the ones with the two highest overall prediction probability through training, the two k with largest $\frac{1}{T} \sum_{t=1}^{T} \frac{1}{N} \sum_{n=1}^{N} \mathbf{e}_k^\mathsf{T} \mathbf{q}^t(\mathbf{x}_n)$ – and sum the total probability for all other classes.

Figure 4: Learning path on CIFAR-10. In the first two panels we record $\mathbf{q}^t$ per batch while in the last panel we record it per epoch.

With these changes, we will now visualize the learning path of data points while training a ResNet-18 (He et al., 2016b) on CIFAR-10 for 100 epochs as an example. Results on some other network structures and datasets and more discussion are in Appendix F.

The first panel of Figure 4 shows the learning path of an easy sample. We can see that $\mathbf{q}^t$ converges quickly towards the left corner, a 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 (57 in this experiment), $\mathbf{q}^t$ has already converged to $\mathbf{e}_y$.

however, for a harder sample (middle), 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 rates 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, then plot the low-frequency components only via a low-pass filter, taking a moving average of the $\mathbf{q}^t$ (red points). This makes it clear that, overall, the pattern is roughly a zig-zag as before, first moving towards the "plane" class before eventually turning to memorize the training "ship" class.

The early-stopping epoch is again near the "corner." We can hope that, across the training dataset, the early-stopped $\mathbf{q}^t$ will on average be near the "corner" and hence hopefully also near $\mathbf{p}^*$. This might not happen, however, either because the early stopping occurs at the wrong point along the smoothed path or, even if the smoothed position is correct, the random particular value of $\mathbf{q}^t$ when we stop might be noisy and far away.

The image in Figure 4 (middle) is difficult for humans to classify, but highlights another possibility for "zig-zagging": it can also occur when an input looks like one label to a simplistic model, but a different one to a more sophisticated model capable of recognizing subtler clues. In the analysis of Proposition 1, this could correspond to changes in $\mathcal{K}^t$ leading to different sets of similar points over time. In such cases, the best label to use is actually *after* the zig-zag, but looking only at the learning path, these two cases are difficult or impossible to distinguish.

5 SMOOTHING AS A METHOD FOR KNOWLEDGE DISTILLATION

The previous analysis gives us some explanation for ESKD's success, but also highlights one particular drawback: the extremely high variance of $\mathbf{q}^t$ through training. Moreover, unlike Figure 4 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, further exacerbating the variance issue for $\mathbf{q}^t$, as shown in the rightmost panel of Figure 4. This makes it easy for the teacher to select a bad $\mathbf{p}_{\text{tar}}$ for many data points.

Inspired by our visualization of the smoothed learning path, i.e., running a low-pass filter on each dimension of $\mathbf{q}_t$, we propose a knowledge distillation variant called Filter-KD in Algorithm 1. Specifically, we maintain a look-up table $\mathbf{q}_{\text{smooth}} \in \mathbb{R}^{N \times K}$ to store a moving average of $\mathbf{q}_t$ for each training sample. Note that in one epoch, each $\mathbf{q}_{\text{smooth}}(\mathbf{x}_n)$ will be updated only once. We check the early stopping criterion with the help of a validation set. Afterwards, the teaching supervision $\mathbf{q}_{\text{smooth}}$ is ready, and we can train a new 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.

We will show the effectiveness of this algorithm in the next subsection, but first that, 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 an extra softmax operation and a few extra summations for each batch. Furthermore, compared to the teaching model in ESKD, the Filter-KD requires a teaching table $\mathbf{q}_{\text{smooth}}$, 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 applied in MOCO (He et al., 2020). We empirically find that, although this helps the model converge faster, it does not lead to a better overall teacher network; see Appendix G. 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.

Setting: Dataset $\{(\mathbf{x}_n, y_n, Id_n)\}_{n=1}^N$, where $Id_n = \{1, 2, ..., N\}$ is index for each pair.
=========================== Train the teacher ===========================
Initialize a network model, initialize a $N \times K$ matrix called $\mathbf{q}_{\text{smooth}}$.
Go through the entire dataset, calculate $\mathbf{q}_{\text{smooth}} = \text{Softmax}(\text{model}(\mathbf{X}))$.
for $t = 1, 2, ..., T$ **do**
    for $\mathbf{x}_{\text{batch}}, y_{\text{batch}}, Id_{\text{batch}}$ **do**
        $hid = \text{model}(\mathbf{x}_{\text{batch}})$
        $prob = \text{Softmax}(hid)$
        $\mathbf{q}_{\text{smooth}}[Id_{\text{batch}}] = (1 - \alpha) \cdot \mathbf{q}_{\text{smooth}}[Id_{\text{batch}}] + \alpha \cdot prob$
        $loss = \text{CrossEntropy}(hid, y_{\text{batch}})$
        Backpropagate and update the parameters
    end for
    Check the early stopping criterion and stop training if satisfied.
end for
=========================== Train the student ===========================
For any input $\mathbf{x}_n$, set $\mathbf{p}_{\text{tar}} = \mathbf{q}_{\text{smooth}}[Id_n]$ and train the network under the supervision of $\mathbf{p}_{\text{tar}}$.

Algorithm 1: Filter-KD. α controls the cut-off frequency of low-pass filter (0.1 here).

5.1 COMPARISON OF FILTER-KD AS A TEACHER MODEL

In the experiments, we choose 5 different random seeds. For each seed, we first train a network under the supervision of one-hot label for maximum 200 epochs, during which we update $\mathbf{q}_{\text{smooth}}$ according Algorithm 1. At the end of each epoch, we validate the network on a validation set and check the early stopping criterion. When the training stops, we evaluate the network on a held-out test set and report the test accuracy and ECE as the generalization performance for OHT. We then initialize two student networks, one for ESKD and the other for Filter-KD, and train them separately under the supervision of the output of the one-hot trained teacher and $\mathbf{q}_{\text{smooth}}$, respectively. Each student is also early-stopped and evaluated on the same test set. To avoid influence from other decisions, we use identical network structures[5], learning rate, scheduler, data augmentation, batch size, temperature, etc. for both teacher and students in all cases. The student models in ESKD and Filter-KD do not receive y_n, only the teacher supervision.

Quantitative results are listed in Table 1, in which we can see small but reliable improvements in accuracy for Filter-KD. Note that in all the settings, for each random seed, Filter-KD strictly outperforms ESKD and OHT. ECE is improved more than accuracy, sometimes quite dramatically; it is natural to expect that a better $\mathbf{p}^*$ will have a more direct effect on ECE than on accuracy.

5.2 ROBUSTNESS TO NOISY LABELS

The zig-zag pattern is particularly relevant when $\mathbf{p}^*$ is far from $\mathbf{e}_y$; this can happen for inherently ambiguous data points, but it especially happens on data points where the observed label is unlikely under $\mathbf{p}^*$. Thus we can expect that Filter-KD, which we expect to generally pick points at the inflection of the zig-zag, should do particularly well on datasets where e.g. some labels have been

---

[5]ResNet-18 for CIFAR-10/100 and Bi-LSTM for AG-News.

	Accuracy			ECE		
	OHT	ESKD	FilterKD	OHT	ESKD	FilterKD
CIFAR10	95.41±0.093	95.47±0.077	95.60±0.103	0.027±0.0011	0.026±0.0010	0.007±0.0021
CIFAR100	78.76±0.135	78.97±0.284	80.05±0.191	0.053±0.0005	0.068±0.0026	0.029±0.0030
AG News	90.51±0.13	90.52±0.25	90.55±0.15	0.671±0.0451	0.454±0.051	0.477±0.1120

Table 1: Quantitative comparison of generalization performance.

Figure 5: Test accuracy of different supervisions under different noisy ratio ρ. In the training and validation set, each label y would be replaced to a random one with probability ρ.

flipped at random, where the learning path should approach the correct label before veering away to the observed label.

To verify this, we compare the test accuracy of one-hot training, ESKD and Filter-KD on different datasets with different portions of the training labels randomly flipped. The experimental setting is otherwise identical to before. In Figure 5, it is clear that Filter-KD outperforms one-hot training and ESKD by a large margin when the noise ratio is significant. As too many points become noisy, though, the labels of similar points become less reliable, and the learning path will no longer head towards $\mathbf{p}^*$ as reliably. Thus, for very high noise ratios, the performance of Filter-KD decays back towards that of OHT.

Figure 6 shows an example of the zig-zag behavior when for one particular sample, in different runs where the image of a car has been assigned different noisy labels and in different training settings (Table 2). Figure 16, in the appendix, shows a random sample of more such images.

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.

Label smoothing. Another popular method is label smoothing (Szegedy et al., 2016), which already discussed and analyzed in the previous sections. Besides our explanation, Lukasik et al. (2020) mathematically show that LS can be considered as a weight shrinking regularizer. Müller et al. (2019) show that LS can enhance the calibration of the model, but is not compatible with KD method. From our perspective, we believe that suitable LS can decrease $\|\mathbf{p}_{\text{tar}} - \mathbf{p}^*\|_2$ to some extent, but the lower bound of this gap should be large, because LS treats each class and each sample in the same way: in e.g. Figure 3, it would simply move the OHT label towards the center.

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,

Figure 6: Learning paths on a CIFAR-10 image with a noisy label, under four different runs with varying hyperparameters (Table 2).

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. Our work also focuses on self-distillation, but These surprising findings sparked studies of self-distillation, and our work also focuses on this direction. However, our discussions and results can directly apply to the larger-teacher case as well.

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.

There are also alternative attractive enhancements and explanations of KD (e.g. Gao et al., 2018; Shen et al., 2021; Allen-Zhu & Li, 2020; Müller et al., 2020), which do not conflict with our results. We can also observe the learning path under their settings to get more insights.

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, e.g., Menon et al. (2019) use gradient clipping, Patrini et al. (2017); Reed et al. (2014) use loss correction, Huang et al. (2020) change the supervision during the training, and Zhang et al. (2020) take advantage of extra information to achieve good performance. From our discussion, denoising is a by-product of Filter-KD, which means it is possible to combine KD-based methods with traditional ones to further enhance the performance. The idea of observing learning path of samples with different difficulty also provides a new perspective of understanding anti-noise task.

7 DISCUSSION

In this paper, we first claim that better supervision signal, i.e., $\mathbf{p}_{tar}$ 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 $\mathbf{p}_{tar}$. The experimental results on various settings (i.e., 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.

Stanislav Fort, Paweł Krzysztof Nowak, Stanislaw Jastrzebski, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 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.

Hangfeng He and Weijie Su. The local elasticity of neural networks. In *International Conference on Learning Representations*, 2019.

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

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, 2016b.

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. 2018.

Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *Proceedings of International Conference on Machine Learning*, pp. 6448–6458. PMLR, 2020.

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.

Rafael Müller, Simon Kornblith, and Geoffrey Hinton. Subclass distillation. *arXiv preprint arXiv:2002.03936*, 2020.

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.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

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.

Zhiqiang Shen, Zechun Liu, Dejia Xu, Zitian Chen, Kwang-Ting Cheng, and Marios Savvides. Is label smoothing truly incompatible with knowledge distillation: An empirical study. *Proceedings of International Conference on Learning Representations*, 2021.

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.

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.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2403–2412, 2018.

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.

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.

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

In section 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: in this dataset, we have N samples and each sample is a 3-tuple $(\mathbf{x}, y, p^*)$. 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 $\{-1, 0, 1\}$. 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:

$$p^*(y = k|\mathbf{x}) = \frac{e^{s_k}}{\sum_{j \neq k} e^{s_j}}; \quad s_i = -\frac{1}{2\sigma^2} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2 \tag{4}$$

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, which are generated by setting $\hat{\mathbf{p}}_{\text{tar}} = \eta\mathbf{p}^* + (1 - \eta)\mathbf{e}_y, \eta \in [0, 1]$.

B SOME BACKGROUND

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:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|, \tag{5}$$

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 7. 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

Figure 7: How to project a probability vector on to the plane of Barycentric coordinate system.

Figure 8: Left panel: comparison between integral difficulty and converged confidence. Right two panels: counts of hardest %10 and easiest %10 CIFAR-10 training samples in 24 different settings.

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 7.

C DEFINITION OF DIFFICULTY MEASUREMENTS

For samples in toy Gaussian dataset, we define base difficulty as $\|\mathbf{e}_{y_n} - \mathbf{p}^*(\mathbf{x}_n)\|_2$, because $\mathbf{p}^*$ can be directly calculated. For real dataset like CIFAR-10, we instead use integral difficulty, i.e., $\frac{1}{T}\sum_{t=1}^{T}(1 - \mathbf{q}^t(y_n = k|\mathbf{x}_n))$. Another popular metric for model's difficulty is the converged confidence, i.e., $\mathbf{q}^T(y_n = k|\mathbf{x}_n)$. We compare these two metrics by showing their correlation with the base difficulty on the toy Gaussian dataset. From the left panel in Figure 8, we see the integral difficulty captures base difficulty better than the converged one, which is reasonable because the model is trained using cross-entropy loss. If the model is sufficiently complex and we train it long enough, the confidence of every sample might converge to 1.

Another flaw of the integral difficulty is that it depends on the network structure, learning rate, batch size, etc., while the base difficulty only depends on the data distribution. However, we experimentally show that the integral difficulty is relatively consistent across different settings. To verify this, we choose two different initializations (different random seeds), three different batch size (64, 128, 256), and four different network structures, i.e., ResNet18, Preact-ResNet18 (He et al., 2016a), DLA (Yu et al., 2018) and mobilenet (Sandler et al., 2018). By combining these settings, we conduct 24 different experiments. For each experiment, we order the training samples by their corresponding integral difficulty. After that, we count how many times one training sample occur in the top-%10 and last-%10 among these experiments and provide the histgram in the right two panels in Figure 8. In CIFAR-10 there are 50000 training samples and its %10 is 5000. In these experiments, 4704 training samples are considered as the top-%10 hardest (and 4584 samples for the last-% hardest) in more than 12 different settings[6], which means the difficulty of the data samples can be well captured by the integral difficulty.

---

[6]These quantities can be calculated by the summation of the height of bars in the figure.

Figure 9: Distance gap of each sample. The blue dots are $\mathbf{q}(\mathbf{x}_n)$ at specific time, the red dots (or line) are $\mathbf{p}_{\text{tar}}$ during training. The first row is label smoothing case, the second row is KD case, the third row is ESKD case, and the last row it trained using $\mathbf{p}^*$

.

D   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.

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.

E   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):

$$\Delta^t(\mathbf{x}_o) = \nabla_{\mathbf{w}}\mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \left(\mathbf{w}^{t+1} - \mathbf{w}^t\right) + \mathcal{O}\left(\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2\right) \tag{6}$$

$$= \left(\nabla_{\mathbf{z}}\mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \nabla_{\mathbf{w}}\mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t}\right) \cdot \left(\mathbf{w}^{t+1} - \mathbf{w}^t\right) + \mathcal{O}(\epsilon^2) \tag{7}$$

$$= \left(\nabla_{\mathbf{z}}\mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \nabla_{\mathbf{w}}\mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t}\right) \cdot \left(-\eta\nabla_{\mathbf{w}}L(\mathbf{x}_u)|_{\mathbf{w}^t}\right)^{\mathsf{T}} + \mathcal{O}(\eta^2) \tag{8}$$

$$= \left(\nabla_{\mathbf{z}}\mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \nabla_{\mathbf{w}}\mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t}\right) \cdot \left(-\eta\nabla_{\mathbf{z}}L(\mathbf{x}_u)|_{\mathbf{w}^t} \cdot \nabla_{\mathbf{w}}\mathbf{z}(\mathbf{x}_u)|_{\mathbf{w}^t}\right)^{\mathsf{T}} + \mathcal{O}(\eta^2) \tag{9}$$

$$= -\eta\nabla_{\mathbf{z}}\mathbf{q}^t(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \left[\nabla_{\mathbf{w}}\mathbf{z}(\mathbf{x}_o)|_{\mathbf{w}^t} \cdot \left(\nabla_{\mathbf{w}}\mathbf{z}(\mathbf{x}_u)|_{\mathbf{w}^t}\right)^{\mathsf{T}}\right] \cdot \left(\nabla_{\mathbf{z}}L(\mathbf{x}_u)|_{\mathbf{w}^t}\right)^{\mathsf{T}} + \mathcal{O}(\eta^2) \tag{10}$$

$$= \eta \cdot \mathcal{A}^t(\mathbf{x}_o) \cdot \mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u) \cdot \left(\mathbf{p}_{\text{tar}}(\mathbf{x}_u) - \mathbf{q}^t(\mathbf{x}_u)\right) + \mathcal{O}(\eta^2), \tag{11}$$

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.

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,

$$\mathcal{A}^t(\mathbf{x}_o) = \begin{bmatrix} q_1(1-q_1) & -q_1q_2 & \cdots & -q_1q_K \\ -q_2q_1 & q_2(1-q_2) & & \\ \vdots & & \ddots & \\ -q_Kq_1 & & & q_K(1-q_K) \end{bmatrix}, \tag{12}$$

which is a symmetric positive semi-definite (PSD) matrix[7] 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 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 10. 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:

$$f(\mathbf{w}, \mathbf{x}) \approx f(\mathbf{w}^0, \mathbf{x}) + \left(\nabla_{\mathbf{w}} f(\mathbf{w}^0, \mathbf{x})\right)^\mathsf{T} \cdot (\mathbf{w} - \mathbf{w}^0) \tag{13}$$

$$= \left(\nabla_{\mathbf{w}} f(\mathbf{w}^0, \mathbf{x})\right)^\mathsf{T} \cdot \mathbf{w} + f(\mathbf{w}^0, \mathbf{x}) - \left(\nabla_{\mathbf{w}} f(\mathbf{w}^0, \mathbf{x})\right)^\mathsf{T} \cdot \mathbf{w}^0 \tag{14}$$

$$= \phi(\mathbf{x})^\mathsf{T} \mathbf{w} + \mathbf{b}, \tag{15}$$

where we use $\phi(\mathbf{x})^\mathsf{T}$ 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 $\mathcal{K}^0(\mathbf{x}_i, \mathbf{x}_j) \triangleq \phi(\mathbf{x}_i)^\mathsf{T} \phi(\mathbf{x}_j)$, 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 11. 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 10, 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.

---

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

Figure 10: 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 $tr(\mathcal{A}^t(\mathbf{x}_n))$.

F MORE ABOUT LEARNING PATHS

In this appendix, we provide the learning path of training samples with different difficulties in different datasets. Note the zig-zag pattern might disappear for some extremely hard samples which are not correctly classified during training phase: the path will not converge to any corner and just hang around in the middle of the triangle plane. From these typical hard samples we provided, we might expect such zig-zag pattern. In Figure 12, a lawn mover also has four wheels, which is quite similar to a truck or a car; in Figure 13, the hard sample contains several country's names, which might let the NN classify it as "world" class; in Figure 14, the nuance differences between different dog species is responsible for the zig-zag pattern.

So from our intuition, the zig-zag pattern in real tasks might also highly correlated with the semantic distance between different classes. For example, an initialized network might first learn to distinguish samples from different "super-classes" (like a dog v.s. a vehicle). After training long enough, if the model's capacity is sufficient, it may then learn the nuance differences between a truck and a car. As the semantic information of different classes cannot be captured by one-hot label (or even LS), the KD-based method can hence improve the performance, as discussed in Allen-Zhu & Li (2020). Furthermore, there are also works studying how to directly use the semantic distance

Figure 11: 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).

Figure 12: The learning path of an easy sample (first panel) and a hard sample (last two panels) in CIFAR100. The middle panel is recorded per-update and the last panel is recorded per-epoch. We see the smoothing method can indeed help to reduce the variance.

between classes on other tasks, like clustering (He & Su, 2019; Fort et al., 2019). The analysis and observing tools provided by this paper might provide a new perspective of understanding this phenomenon. We left these questions for our future work.

Figure 13: The learning path of an easy sample (first panel) and a hard sample (last two panels) in AG News.

Figure 14: Three examples in ImageNet. As this dataset is quite complex and there are too many classes, we only select some typical samples and observe them. The ambiguous class in the medium and hard samples are likely to be in the same super-class. We left more detailed analysis for our future work.

G 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 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 $\mathbf{q}_{\text{smooth}}$ 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, $\mathbf{w}_{\text{train}}^{t+1} = \mathbf{w}_{\text{train}}^{t} + \eta\nabla L$, for the tracking model, $\mathbf{w}_{\text{track}}^{t+1} = (1 - \alpha)\mathbf{w}_{\text{track}}^{t} + \alpha\mathbf{w}_{\text{train}}^{t+1}$. We train the model for 150 epochs, and show the learning curve of test accuracy in the left panel of Figure 15. 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 $\mathbf{q}_{\text{smooth}}$ when teaching a new model, as illustrated in the right panel of Figure 15.

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.

Figure 15: Behavior of the tracking model (with low pass filter on each parameters). ResNet18 trained for 150 epochs on CIFAR100.

	batch size	learning rate	data augmentation
1st col.	256	1e-4	No
2nd col.	32	1e-4	No
3rd col.	32	Cosine 1e-2 to 1e-5	No
4th col.	32	Cosine 1e-2 to 1e-5	Random Crop, flip

Table 2: Hyperparameter settings for Figure 16.

H MORE ON THE NOISY LABEL TASK

In this appendix, we provide the learning path of several manually noisy labeled samples in CIFAR-10 to further support our analysis. In this experiment, we randomly select 1000 training samples and flip their labels (the resulting label is different from their true label). As these examples are very likely to be easy samples before label flipping (remember large amount samples in a dataset should be easy), their noisy version might be hard samples: the ground truth class and label class is different. After training 100 epochs with this noisy dataset under different settings (see Table 2), we see all these models achieve perfect training accuracy, which means the learning path of these noisy samples also converge to their one-hot label. In Figure 16, we randomly select 7 samples and show their learning path under different settings. It is clear that the learning path would first move towards the right corner, which represent their ground truth labels, and then converge back to the left corner (their labels). Comparing the sub-figures in each row, we can further conclude that smaller batch size, higher initial learning rate and random data augmentation might increase the variance of the original learning path and make the zig-zag pattern more remarkable. These trends also supports our analysis about why zig-zag occurs and how it help.

Figure 16: Track the learning path of some noisy label inputs under different training settings.