Toward Student-oriented Teacher Network Training for Knowledge Distillation

Chengyu Dong
University of California, San Diego
cdong@eng.ucsd.edu

Liyuan Liu
University of Illinois at Urbana-Champaign
ll2@illinois.edu

Jingbo Shang
University of California, San Diego
jshang@eng.ucsd.edu

Abstract

How to conduct teacher training for knowledge distillation is still an open problem. It has been widely observed that a best-performing teacher does not necessarily yield the best-performing student, suggesting a fundamental discrepancy between the current teacher training practice and the ideal teacher training strategy. To fill this gap, we explore the feasibility of training a teacher that is oriented toward student performance with empirical risk minimization (ERM). Our analyses are inspired by the recent findings that the effectiveness of knowledge distillation hinges on the teacher's capability to approximate the true label distribution of training inputs. We theoretically establish that ERM minimizer can approximate the true label distribution of training data as long as the feature extractor of the learner network is Lipschitz continuous and is robust to feature transformations. In light of our theory, we propose a teacher training method *SoTeacher* which incorporates Lipschitz regularization and consistency regularization into ERM. Experiments on benchmark datasets using various knowledge distillation algorithms and teacher-student pairs confirm that *SoTeacher* can improve student accuracy consistently.

1 Introduction

Knowledge distillation aims to train a small yet effective *student* neural network following the guidance of a large *teacher* neural network (Hinton et al., 2015). It dates back to the pioneering idea of model compression (Buciluǎ et al., 2006) and has a wide spectrum of real-world applications, such as recommender systems (Tang & Wang, 2018; Zhang et al., 2020), question answering systems (Yang et al., 2020; Wang et al., 2020) and machine translation (Liu et al., 2020).

Despite the prosperous interests in knowledge distillation, one of its crucial components, teacher training, is largely neglected. The existing practice of teacher training is often directly targeted at maximizing the performance of the teacher, which does not necessarily transfer to the performance of the student. Empirical evidence shows that a teacher trained toward convergence will yield an inferior student (Cho & Hariharan, 2019) and regularization methods benefitting the teacher may contradictorily degrade student performance (Müller et al., 2019).

In this work, we explore the theoretical feasibility of training the teacher toward student performance. Our analyses are built upon the recent understanding of knowledge distillation from a statistical perspective. In specific, Menon et al. (2021) show that the soft prediction provided by the teacher is essentially an approximation to the true label distribution, and true label distribution as supervision for the student improves the generalization bound compared to one-hot labels. Dao et al. (2021) show that the accuracy of the student is directly bounded by the distance between teacher's prediction and the true label distribution through the Rademacher analysis.

37th Conference on Neural Information Processing Systems (NeurIPS 2023) M3L Workshop.

Based on the above understanding, a teacher benefitting the student should be able to learn the true label distribution of the *distillation data* [1]. Since practically the distillation data is often reused from the teacher's training data, the teacher will have to learn the true label distribution of its own training data. This might appear to be infeasible using standard empirical risk minimization, as the teacher network often has enough capacity to fit all one-hot training labels, in which case, distilling from teacher predictions should largely degrade to directly training with one-hot labels. Existing works tend to evade this dilemma by distilling from teacher predictions only on data that is not used in teacher training (Menon et al., 2021; Dao et al., 2021).

Instead, we directly prove the feasibility of training the teacher to learn the true label distribution of its training data. We show that the standard empirical risk minimizer can approach the true label distribution of training data under a mixed-feature data distribution, as long as the feature extractor of the learner network is Lipschitz continuous and is robust to feature transformations.

In light of our theory, we show that explicitly imposing the Lipschitz and consistency constraint in teacher training can facilitate the learning of the true label distribution and thus improve the student performance (See Section D). We conduct extensive experiments on benchmark datasets using various knowledge distillation algorithms and different teacher-student architecture pairs (See Section F), which demonstrate that our method can improve student performance consistently and significantly.

2 Preliminaries

We study knowledge distillation in the context of multi-class classification. Specifically, we are given a set of training samples $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i\in[N]}$, where $[N] \coloneqq \{1, 2, \cdots, N\}$. $\mathcal{D}$ is drawn from a probability distribution $p_{X,Y}$ that is defined jointly over input space $\mathcal{X}$ and label space $\mathcal{Y} = [K]$. For convenience, we denote $1(y) \in \mathbb{R}^K$ as the one-hot encoding of label y.

Learning Objective of Teacher in Practice. In common practice, the teacher network f is trained to minimize the empirical risk given a loss function ℓ, namely $\min_f \hat{R}(f)$, where $\hat{R}(f) = \mathbb{E}_{(x,y)\in\mathcal{D}}\ \ell(f(x), y)$, and $\mathbb{E}_{(x,y)\in\mathcal{D}}$ is the empirical expectation. We consider cross-entropy loss $\ell(f(x), y) = -1(y) \cdot \log f(x)$.

Ideal Learning Objective of Teacher. Recent advances in understanding knowledge distillation suggest a teacher should approximate the true label distribution of the distillation data, which is often reused from the teacher's training data. From this perspective, ideally the teacher should learn the true label distribution of its training data, namely $\min_f R(f)$, where $R(f) = \mathbb{E}_{(x,y)\in\mathcal{D}}\ \|f(x) - p^*(x)\|$. Here, $p^*(x) \coloneqq p_{Y|X}(\cdot|x)$ denotes the *true label distribution* of an input x, namely the (unknown) distribution that its label is sampled from, which is not necessarily one-hot. And $\|\cdot\|$ is a norm.

Our Research Question. One can find that there is a fundamental discrepancy between the learning objective of the teacher in the common practice and that in the ideal case. In particular, minimization of $\hat{R}(f)$ would lead to $f(x) = 1(y)$ for any input x, which significantly deviates from $p^*(x)$ and thus challenges the effectiveness of knowledge distillation. Therefore, in this work, we explore the following question.

- *Can a teacher network learn the true label distribution of the training data with the standard teacher training practice?*

Note that our focus in the rest of the paper is teacher training, thus by "data" we refer to the data used to train the teacher, and by "network" we refer to the teacher network.

3 Theoretical Feasibility to Learn True Label Distribution of Training Data

3.1 Notations and Problem Setup

Notation. We will use calligraphic typefaces to denote sets, *e.g.*, the dataset $\mathcal{D}$. We use $|\mathcal{D}|$ to denote the size of the set. We use $\circ$ to denote function composition. We use $\tilde{O}(\eta)$ to denote polynomial terms of η.

[1] For simplicity, we refer to the training data of the student model in knowledge distillation as the distillation data (Stanton et al., 2021)

Data Distribution. We consider a data distribution referred to as the *mixed-feature distribution*. Our distribution can be viewed as a simplified version of the "multi-view distribution" introduced in Allen-Zhu & Li (2020). Our distribution can also be viewed as a variation of Latent Dirichlet Allocation (LDA) (Blei et al., 2001), a generative data distribution widely used to model text data.

Definition 3.1 (Mixed-feature distribution)**.** *We first generate the input x following LDA. We define a feature vocabulary $\mathcal{Z}$ that denotes the names of all possible features in the data inputs. For example, in image classification we can have $\mathcal{Z} = \{$'eye', 'tail', 'wheel', ...$\}$. Now for each data example i,*

1. *Sample M feature names, namely $\{z_m\}_{m\in[M]} \sim p_Z$, where p_Z is a discrete distribution over $\mathcal{Z}$.*
2. *For each feature name z, we sample its input representation $x_z \in \mathbb{R}^b$, namely $x_z \sim p_X(\cdot|z)$, where $p_X(\cdot|z)$ is a continuous distribution with finite variance [2], namely Var$[X|z] \le \nu_z$.*
3. *For each feature name z, we transform its input representation by a function $\gamma : \mathbb{R}^b \to \mathbb{R}^b$, namely $x_z := \gamma(x_z)$, where γ is sampled from a set of possible transformation functions $\mathcal{T}$ [3].*
4. *We concatenate the transformed representations of these features to obtain the input, namely $x = (x_{z_1}, x_{z_2}, \cdots, x_{z_M})$, where $x \in \mathbb{R}^{b\times M}$. This views each data input as a concatenation of M patches and each patch is a b-dimensional vector [4], which follows Allen-Zhu & Li (2020).*

Now we generate the label y. We assume each feature name defines a label distribution $p_Y(\cdot|z)$ [5]. The label distribution of an input x is the geometric average of the label distributions of the feature names, namely $y \sim p_Y(\cdot|x) := (\prod_m p_Y(\cdot|z_m))^{1/M}$.

Network Architecture. We consider a multi-layer neural network that produces probabilistic outputs, namely $f : \mathbb{R}^{b\times M} \to [0,1]^K$. The network consists of a feature extractor and a classification head, namely $f := f_C \circ f_E$, defined as follows respectively.

- $f_E : \mathbb{R}^{b\times M} \to \mathbb{R}^{M\times d}$ denotes the feature extractor, whose architecture can be arbitrary as long as it processes each patch independently [6].
- $f_C : \mathbb{R}^{M\times d} \to \mathbb{R}^K$ denotes the probabilistic classification head, which consists of a 1x1 convolutional layer and a modified Softmax layer, namely $f_C(h) = \widetilde{\text{Softmax}}(w_C h)$, where $w_C \in \mathbb{R}^{1\times K\times d}$. The 1x1 convolutional layer is similar to the assumption in (Allen-Zhu & Li, 2020), while the modified Softmax is slightly different from the standard Softmax in terms of the denominator, namely $\widetilde{\text{Softmax}}(\hat{h}) = \frac{\exp(1/M\sum_m \hat{h}_m)}{(\prod_m \sum_k \exp(\hat{h}_{m,k}))^{1/M}}$, where $\hat{h} := w_c h$.

3.2 A Hypothetical Case: Invariant Feature Extractor

For starters, we investigate a hypothetical case where the feature extractor is *invariant*, which means that it always outputs the same feature map given the same feature patch, regardless of which input this feature patch is located at or which transformation is applied to this feature patch. Given such an assumption, showing the learning of true label distribution of the training data can be greatly simplified. We will thus briefly walk through the steps towards this goal to shed some intuitions.

Definition 3.2 (Invariant feature extractor)**.** *We call f_E an invariant feature extractor, if for any two inputs i and j, for any two transformations γ and γ', $f_E(\gamma(x_m^{(i)})) = f_E(\gamma'(x_{m'}^{(j)}))$, as long as $z_m = z_{m'}$, namely the feature names of the patches m and m' match.*

We first show that given an invariant feature extractor, the minimizer of the empirical risk has the property that its probabilistic prediction of each feature converges to the sample mean of the labels whose corresponding inputs contain this feature.

[2] The finite variance means that the representation of the same feature sampled in different inputs is similar.

[3] For example, in image classification γ can be rotation, mirroring, or resizing.

[4] Note that this is only to simplify the notations and can be generalized, as our theory is not dependent on it. The specific shape of each patch can be arbitrary, for example, can be a 3-rank tensor (height, width, channel) in image classification.

[5] For example, $p_Y(\cdot|\text{'eye'}) = (0.5, 0.5, 0, \cdots)$, assuming the label set is given as ('cat', 'dog', 'car', $\cdots$).

[6] With a little abuse of notation, we will also write $h_m := f_E(x_m)$

Lemma 3.3 (Convergence of the probabilistic predictions of features)**.** *Let* $\bar{y}_z := \frac{1}{N}\sum_{\{i|z\in\mathcal{Z}^{(i)}\}} 1(y^{(i)})$*, where* $\mathcal{Z}^{(i)}$ *denote the set of feature names in the* i*-th input, and thus* $\{i|z \in \mathcal{Z}^{(i)}\}$ *denotes the set of inputs that contain feature* z*. Let* $f^* = \arg\min_f \hat{R}(f)$ *and assume* f^*_E *is an invariant feature extractor. Let* $p_{f^*}(x_z) := \mathit{Softmax}(w_C f^*_E(x_z))$ *be the probabilistic prediction of feature* z*. We have* $p_{f^*}(x_z) = \bar{y}_z$*.*

The intuition here is that since the feature extractor is invariant and the number of possible features is limited, the feature maps fed to the classification head would be a concatenation of M vectors selected from a fixed set of $|\mathcal{Z}|$ candidate vectors, where each vector corresponds to one feature in the vocabulary. Therefore, the empirical risk can be regrouped as $-\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log f^*(x^{(i)}) = -\frac{1}{M}\sum_{z\in\mathcal{Z}} \bar{y}_z \cdot \log p_{f^*}(x_z)$. Then by Gibbs' inequality, the empirical risk can be minimized only when the probabilistic prediction of each feature $p_{f^*}(x_z)$ is equal to $\bar{y}_z$.

We now show that the sample mean of multiple labels $\bar{y}_z$ will converge to the average distribution of these labels, even though they are non-identically distributed. Since each label is a categorical random variable, this directly follows the property of multinomial distributions (Lin et al., 2022) [7].

Lemma 3.4 (Convergence of the sample mean of labels)**.** *Let* $\bar{p}(\cdot|z) = \frac{1}{N}\sum_{\{i|z\in\mathcal{Z}^{(i)}\}} p_Y(\cdot|x^{(i)})$*, then with probability at least* $1-\delta$*, we have* $\|\bar{y}_z - \bar{p}(\cdot|z)\| \leq \tilde{O}\left(\sqrt{KN^{-1}M|\mathcal{Z}|^{-1}\delta^{-1}}\right)$*.*

Next, we show that the average label distribution $\bar{p}(\cdot|z)$ approximates the true label distribution of the corresponding feature z.

Lemma 3.5 (Approximation of the true label distribution of each feature)**.** *Following notations in Lemma 3.4, we have* $\|\bar{p}(\cdot|z) - p_Y(\cdot|z)\| \leq \tilde{O}\left(M|\mathcal{Z}|^{-1}\right)$*.*

The intuition is that since the average distributions $\bar{p}(\cdot|z)$ here are all contributed by the true label distribution of the inputs that contain feature z, it will be dominated by the label distribution of feature z, given some minor assumption on the sampling process of features when generating each input.

Finally, combining Lemmas 3.3, 3.4, 3.5, one can see that the probabilistic prediction of each feature given by the empirical risk minimizer will approximate the true label distribution of that feature. Subsequently, we can show that the probabilistic prediction of each input approximates the true label distribution of that input, which leads to the following main theorem.

Theorem 3.6 (Approximation error under a hypothetical case)**.** *Given the setup introduced in Section 3.1, let* $f^* = \arg\min_f \hat{R}(f)$ *and assume* f^*_E *is an invariant feature extractor. Then for any input* $x \in \mathcal{D}$*, with probability at least* $1-\delta$*,*

$$\|f^*(x) - p^*(x)\| \leq \tilde{O}\left(\sqrt{KN^{-1}M|\mathcal{Z}|^{-1}\delta^{-1}}\right) + \tilde{O}\left(M|\mathcal{Z}|^{-1}\right). \tag{1}$$

3.3 The Realistic Case: Robust Feature Extractor

In a realistic case where the feature extractor is not exactly invariant, it is still possible to approximate the true label distribution, as long as the feature extractor is robust to the variation of features across inputs and also robust to feature transformations. In fact, we only need to modify Lemma 3.3 in the hypothetical case, where the probabilistic predictions of features $p_{f^*}(x_z)$ will now converge to $\bar{y}_z$ with some constant error. This in turn causes additional error terms in Theorem 3.6. We will defer the detailed discussion to Section A.1.

4 Conclusion and Future Work

In this work, we rigorously studied the feasibility to learn the true label distribution of the training data under a standard empirical risk minimization framework. We also explore possible improvements of current teacher training that facilitate such learning. We believe our work is among the first attempts to explore the theory and practice of training a *student-oriented* teacher in knowledge distillation. In the future, we plan to adapt our theory to other knowledge distillation scenarios such as transfer learning and mixture of experts, and explore more effective student-oriented teacher training methods.

[7] It is also feasible to achieve this lemma by applying Lindeberg's central limit theorem, with a weak assumption that the variance of each label is finite.

References

Tiny imagenet, 2017. URL `https://www.kaggle.com/c/tiny-imagenet`.

Ahn, S., Hu, S. X., Damianou, A. C., Lawrence, N. D., and Dai, Z. Variational information distillation for knowledge transfer. *CVPR*, pp. 9155–9163, 2019.

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *ArXiv*, abs/2012.09816, 2020.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. Spectrally-normalized margin bounds for neural networks. In *NIPS*, 2017.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *ArXiv*, abs/1911.09785, 2019.

Blei, D. M., Ng, A., and Jordan, M. I. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2001.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Chen, X. A new generalization of chebyshev inequality for random vectors. *ArXiv*, abs/0707.0805, 2007.

Cho, J. H. and Hariharan, B. On the efficacy of knowledge distillation. *ICCV*, pp. 4793–4801, 2019.

Dao, T., Kamath, G. M., Syrgkanis, V., and Mackey, L. W. Knowledge distillation as semiparametric inference. *ArXiv*, abs/2104.09732, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Dong, C., Liu, L., and Shang, J. Double descent in adversarial training: An implicit label noise perspective. *ArXiv*, abs/2110.03135, 2021.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. In *ICML*, 2018.

Geifman, Y., Uziel, G., and El-Yaniv, R. Bias-reduced uncertainty estimation for deep neural classifiers. In *ICLR*, 2019.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110:393–416, 2021.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *ArXiv*, abs/1706.04599, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, pp. 770–778, 2016.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. Augmix: A simple data processing method to improve robustness and uncertainty. *ArXiv*, abs/1912.02781, 2020.

Heo, B., Lee, M., Yun, S., and Choi, J. Y. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Huang, Z. and Wang, N. Like what you like: Knowledge distill via neuron selectivity transfer. *ArXiv*, abs/1707.01219, 2017.

Jin, X., Peng, B., Wu, Y., Liu, Y., Liu, J., Liang, D., Yan, J., and Hu, X. Knowledge distillation via route constrained optimization. *ICCV*, pp. 1345–1354, 2019.

Kim, J., Park, S., and Kwak, N. Paraphrasing complex network: Network compression via factor transfer. *ArXiv*, abs/1802.04977, 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *ArXiv*, abs/1610.02242, 2017.

Lin, Z., Wang, Y., and Hong, Y. The poisson multinomial distribution and its applications in voting theory, ecological inference, and machine learning. 2022.

Liu, J., Shou, L., Pei, J., Gong, M., Yang, M., and Jiang, D. Cross-lingual machine reading comprehension with language branch knowledge distillation. In *COLING*, 2020.

Matsubara, Y. torchdistill: A modular, configuration-driven framework for knowledge distillation. In *International Workshop on Reproducible Research in Pattern Recognition*, pp. 24–44. Springer, 2021.

Menon, A. K., Rawat, A. S., Reddi, S. J., Kim, S., and Kumar, S. A statistical perspective on distillation. In *ICML*, 2021.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018.

Mobahi, H., Farajtabar, M., and Bartlett, P. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 33:3351–3361, 2020.

Moon, J., Kim, J., Shin, Y., and Hwang, S. Confidence-aware learning for deep neural networks. In *ICML*, 2020.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *NeurIPS*, 2019.

Naeini, M. P., Cooper, G. F., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, 2015:2901–2907, 2015.

Park, D. Y., Cha, M., Jeong, C., Kim, D., and Han, B. Learning student-friendly teacher networks for knowledge distillation. *ArXiv*, abs/2102.07650, 2021.

Park, W., Kim, D., Lu, Y., and Cho, M. Relational knowledge distillation. *CVPR*, pp. 3962–3971, 2019.

Passalis, N. and Tefas, A. Learning deep representations with probabilistic knowledge transfer. In *ECCV*, 2018.

Peng, B., Jin, X., Liu, J., Zhou, S., Wu, Y., Liu, Y., Li, D., and Zhang, Z. Correlation congruence for knowledge distillation. *ICCV*, pp. 5006–5015, 2019.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2015.

Shah, H., Khare, A., Shah, N., and Siddiqui, K. Kd-lib: A pytorch library for knowledge distillation, pruning and quantization, 2020.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *ArXiv*, abs/2001.07685, 2020.

Stanton, S., Izmailov, P., Kirichenko, P., Alemi, A. A., and Wilson, A. G. Does knowledge distillation really work? *Advances in Neural Information Processing Systems*, 34:6906–6919, 2021.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. *CVPR*, pp. 2818–2826, 2016.

Tan, M., Chen, B., Pang, R., Vasudevan, V., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. *CVPR*, pp. 2815–2823, 2019.

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

Tang, J., Shivanna, R., Zhao, Z., Lin, D., Singh, A., Chi, E. H., and Jain, S. Understanding and improving knowledge distillation. *ArXiv*, abs/2002.03532, 2020.

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. E. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, 2019.

Tian, Y., Krishnan, D., and Isola, P. Contrastive representation distillation. *ArXiv*, abs/1910.10699, 2020.

Tung, F. and Mori, G. Similarity-preserving knowledge distillation. *ICCV*, pp. 1365–1374, 2019.

Wang, Q., Yang, L., Kanagal, B., Sanghai, S. K., Sivakumar, D., Shu, B., Yu, Z., and Elsas, J. L. Learning to extract attribute value from product via question answering: A multi-task approach. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Xie, Q., Dai, Z., Hovy, E. H., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation for consistency training. *arXiv: Learning*, 2020.

Xu, G., Liu, Z., Li, X., and Loy, C. C. Knowledge distillation meets self-supervision. In *ECCV*, 2020.

Yang, Z., Shou, L., Gong, M., Lin, W., and Jiang, D. Model compression with two-stage multi-teacher knowledge distillation for web question answering system. *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

Yoshida, Y. and Miyato, T. Spectral norm regularization for improving the generalizability of deep learning. *ArXiv*, abs/1705.10941, 2017.

Yuan, L., Tay, F. E. H., Li, G., Wang, T., and Feng, J. Revisiting knowledge distillation via label smoothing regularization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3902–3910, 2020.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *ArXiv*, abs/1605.07146, 2016.

Zagoruyko, S. and Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *ArXiv*, abs/1612.03928, 2017.

Zhang, H., Cissé, M., Dauphin, Y., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *ArXiv*, abs/1710.09412, 2018a.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CVPR*, pp. 6848–6856, 2018b.

Zhang, Y., Xu, X., Zhou, H., and Zhang, Y. Distilling structured knowledge into embeddings for explainable and accurate recommendation. *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

Zhang, Z. and Sabuncu, M. R. Self-distillation as instance-specific label smoothing. *ArXiv*, abs/2006.05065, 2020.

A Additional theory

A.1 Realistic Case

We now show that in a realistic case where the feature extractor is not exactly invariant, it is still possible to approximate the true label distribution, as long as the feature extractor is robust to the variation of features across inputs and also robust to feature transformations.

Definition A.1 (Lipschitz-continuous feature extractor)**.** *We call f_E a L_X-Lipschitz-continuous feature extractor, if for any two inputs i and j, $\|f_E(x_m^{(i)}) - f_E(x_{m'}^{(j)})\| \leq L_X \|x_m^{(i)} - x_{m'}^{(j)}\|$, as long as $z_m = z_{m'}$, namely the feature names of the patches m in i and patch m' in j match.*

Definition A.2 (Transformation-robust feature extractor)**.** *We call f_E a L_Γ-transformation-robust feature extractor, if for any patch m in any input i, and for any transformation γ, $\|f_E(\gamma(x_m^{(i)})) - f_E(x_m^{(i)})\| \leq L_\Gamma$.*

Similar to Lemma 3.3, given a Lipschitz-continuous and transformation-robust feature extractor, the probabilistic predictions of features will still converge to the sample mean of the labels where the corresponding inputs contain this feature, up to some constant error. This requires the assumption that the input representation of the same feature is similar across different inputs, which is intuitive and covered by Definition 3.1. All other lemmas introduced in the hypothetical case still hold trivially as they are not dependent on the network. Therefore, we can have the following result.

Theorem A.3 (Approximation error under a realistic case)**.** *Given the setup introduced in Section 3.1, let $f^* = \min_f \hat{R}(f)$ and assume f_E^* is a L_X-Lipschitz-continuous and L_Γ-transformation-robust feature extractor. Let $\nu = \max_z \nu_z$. Then for any input $x \in \mathcal{D}$, with probability at least $1 - \delta$,*

$$\|f^*(x) - p^*(x)\| \leq \tilde{O}\left(\sqrt{KN^{-1}M|\mathcal{Z}|^{-1}\delta^{-1}}\right) + \tilde{O}\left(M|\mathcal{Z}|^{-1}\right) + \tilde{O}(L_X \tilde{\delta}^{-0.5}\nu) + \tilde{O}(L_\Gamma). \quad (2)$$

B Proof

B.1 Modified Softmax

We provide more details for the modified Softmax we defined. Recall that

Definition B.1 (Modified Softmax)**.**

$$\widetilde{Softmax}(\hat{h}) = \frac{\exp(\frac{1}{M}\sum_m \hat{h}_m)}{(\prod_m \sum_k \exp(\hat{h}_{m,k}))^{1/M}}. \quad (3)$$

Our modified Softmax layer maps $\hat{h} \in \mathbb{R}^{M \times K}$ to the probabilistic output $f(x) \in \mathbb{R}^K$. It can be viewed as a combination of an average pooling layer and a Softmax layer, where the denominator of the softmax layer is a geometric average of the sum-exp of the probabilistic output of each batch $\hat{h}_m$.

B.2 Lemma 3.3

Proof. Given the labeled dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}$, the empirical risk of the network function f under cross-entropy loss can be written as

$$
\begin{aligned}
&-\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log f(x^{(i)}) \\
=&-\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log \widetilde{\text{Softmax}}(w_C f_E(x^{(i)})) \\
:=&-\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log \widetilde{\text{Softmax}}\left(\hat{h}^{(i)}\right) \\
=&-\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log \frac{\exp(\frac{1}{M}\sum_m \hat{h}_m^{(i)})}{(\prod_m \sum_k \exp(\hat{h}_{m,k}^{(i)}))^{1/M}} \\
=&-\frac{1}{N}\sum_i 1(y^{(i)}) \cdot \log \left(\frac{\prod_m \exp \hat{h}_m^{(i)})}{\prod_m \sum_k \exp(\hat{h}_{m,k}^{(i)})}\right)^{1/M} \qquad (4) \\
=&-\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \log \frac{\prod_m \exp(\hat{h}_m^{(i)})}{\prod_m \sum_k \exp(\hat{h}_{m,k}^{(i)})} \\
=&-\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log \frac{\exp(\hat{h}_m^{(i)})}{\sum_k \exp(\hat{h}_{m,k}^{(i)})} \\
=&-\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log \text{Softmax}(\hat{h}_m^{(i)}) \\
:=&-\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log p_m^{(i)},
\end{aligned}
$$

where we have defined $p_m^{(i)} = \text{Softmax}(\hat{h}_m^{(i)}) \equiv \text{Softmax}(w_c f_E(x_m^{(i)}))$. Here $x_m^{(i)}$ indicates the m-th patch of the input x.

Now recall our assumption that f_E is an invariant feature extractor, which means that for any i and j, any m and m', and any γ and γ', $f_E(\gamma(x_m^{(i)})) = f_E(\gamma'(x_m^{(j)}))$ as long as $z_m = z_{m'}$. Since, $p_m^{(i)} = \text{Softmax}(\hat{h}_m^{(i)}) = \text{Softmax}(w_C f_E(\gamma(x_m^{(i)})))$, this means that $p_m^{(i)} = p_{m'}^{(j)}$ as long as $z_m = z_{m'}$.

Therefore, the empirical risk can be regrouped as

$$
\begin{aligned}
&-\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_m \log p_m^{(i)} \\
=&-\frac{1}{MN}\sum_i 1(y^{(i)}) \cdot \sum_{z \in \mathcal{Z}^i} \log p_z \\
=&-\frac{1}{M}\sum_{z \in \mathcal{Z}} \left(\frac{1}{N}\sum_{\{i | z \in \mathcal{Z}^{(i)}\}} 1(y^{(i)})\right) \cdot \log p_z,
\end{aligned}
$$

where we have let $p_z := p_m^{(i)}$ where $z_m = z$.

Now we use the KKT condition to approach the empirical risk minimization problem since the only variable is p_z. The first-order stationarity condition gives

$$
\nabla_{p_z} \cdot = -\frac{1}{MN}\sum_i 1(y^{(i)}) \odot \frac{1}{p_z} + \lambda \mathbf{1} = 0, \qquad (5)
$$

where $\odot$ indicates the Hadamard product. Now Hadamard product both sides by p_z we have

$$
-\frac{1}{MN}\sum_i 1(y^{(i)}) + \lambda p_z = 0,
$$

and thus

$$p_z = \frac{1}{MN\lambda} \sum_i 1(y^{(i)}).$$

Now consider the condition $\mathbf{1} \cdot p_z = 1$, we have

$$\frac{1}{MN\lambda} \sum_i 1(y^{(i)}) \cdot \mathbf{1} = 1.$$

Since $\sum_i 1(y^{(i)}) \cdot \mathbf{1} = |\{i|z \in \mathcal{Z}^{(i)}\}|$,

$$\lambda = \frac{|\{i|z \in \mathcal{Z}^{(i)}\}|}{MN}.$$

This leads to

$$p_z = \frac{1}{N_z} \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} 1(y^{(i)}),$$

where $N_z := |\{i|z \in \mathcal{Z}^{(i)}\}|$.

□

B.3 Lemma 3.4

Proof. The sum of independently but non-identically distributed multinomial random variables is known as the Possion multinomial distribution (PMD). We utilize a known result of PMD to prove this lemma.

Proposition B.2 (Lin et al. (2022))**.** *Let $\boldsymbol{I}_i = (I_{i1,\cdots,I_{im}}), i = 1, \cdots, n$ be n independent random indicators where $I_{ij} \in \{0, 1\}$ and $\sum_{j=1}^{m} I_{ij} = 1$ for each i. Let $\boldsymbol{p}_i = (p_{i1}, \cdots, p_{im})$ be the probability vector that $\boldsymbol{I}_i$ is sampled from, where $\sum_{j=1}^{m} p_{ij} = 1$. Let $\boldsymbol{X}$ be the sum of these n random indicators, namely $\boldsymbol{X} = (X_1, \cdots, X_m) = \sum_{i=1}^{n} \boldsymbol{I}_i$. Then we have*

$$\mathbb{E}[\boldsymbol{X}] = (p_{\cdot 1}, \cdots, p_{\cdot m}), \tag{6}$$

where $p_{\cdot j} = \sum_{i=1}^{n} p_{ij}$.

And

$$\Sigma_{jk} = \begin{cases} \sum_{i=1}^{n} p_{ij}(1 - p_{ij}), \text{ if } j = k, \\ -\sum_{i=1}^{n} p_{ij} p_{ik}, \text{ if } j \neq k, \end{cases} \tag{7}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of $\boldsymbol{X}$.

Following this result, since $1(y^{(i)})$ is a random indicator and $y^{(i)} \sim p_Y(\cdot|x^{(i)})$, we have

$$\mathbb{E}[\bar{y}_z] = \bar{p}(\cdot|z) := \frac{1}{N} \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} p_Y(\cdot|x^{(i)}). \tag{8}$$

And let the covariance matrix of $\bar{y}_z$ to be $\Sigma_{\bar{y}_z}$. Let $N_z := |\{i|z \in \mathcal{Z}^{(i)}\}|$ be the number of inputs that contain feature z, we have

$$\begin{aligned} \mathrm{tr}[\Sigma_{\bar{y}_z}] &= \frac{1}{N^2} \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} \sum_{k=1}^{K} p_Y(k|x^{(i)})(1 - p_Y(k|x^{(i)})) \\ &= \frac{1}{N^2} \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} \sum_{k=1}^{K} \left(1 - p_Y^2(k|x^{(i)})\right) \\ &\leq \frac{(K-1)N_z}{N^2}, \end{aligned} \tag{9}$$

where we utilized the fact that $\sum_k p_Y^2(k|x^{(i)}) \geq \frac{1}{K}$ by the Cauchy-Schwarz inequality.

Then by a multivariate Chebyshev inequality (see, *e.g.*, Chen (2007)), we have with probability $1-\delta$,

$$\|\bar{y}_z - \bar{p}(\cdot|z)\|_2^2 \le \frac{\mathrm{tr}(\Sigma_{\bar{y}_z})}{\delta} \le \frac{(K-1)N_z}{N^2} \cdot \frac{1}{\delta}. \tag{10}$$

Now finally, we derive an estimation of N_z. Recall that each input contains M feature names sampled from the feature vocabulary $|\mathcal{Z}|$ subject to the distribution p_Z. Therefore,

$$\mathbb{E}\left[\frac{N_z}{N}\right] = P(z \in \mathcal{Z}^{(i)}) = 1 - (1 - p_Z(z))^M \approx M p_Z(z). \tag{11}$$

For simplicity, we assume p_Z is a uniform distribution, which means $p_Z(z) \approx 1/|\mathcal{Z}|$, for any z. Note that this is not a necessary assumption since N_z can also be trivially bounded by N. Therefore we have

$$\|\bar{y}_z - \bar{p}(\cdot|z)\|_2^2 \le \frac{(K-1)M}{N|\mathcal{Z}|} \cdot \frac{1}{\delta}. \tag{12}$$

Recall that M is the number of features (patches) in each input, and $|\mathcal{Z}|$ is the total number of features in the vocabulary. This result thus indicates a larger feature vocabulary can improve the approximation while a more complicated input can hurt the approximation.

□

B.4 Lemma 3.5

Proof. To prove this lemma we need an additional weak assumption on the true label distribution of features that are "similar".

Assumption B.3. *Let $I(z, z')$ define the pointwise mutual information (PMI) between features z and z'. Let ψ be a concave and monotonically* ***decreasing*** *function. We assume*

$$D_{KL}\left(p_Y(\cdot|z) \,\|p_Y(\cdot|z')\right) = \psi\left(I(z, z')\right). \tag{13}$$

Eq. (13) indicates that when sampling the features in an input, if two features are more likely to be sampled together, their true label distribution should be more similar.

Given Eq. (13) and recall that

$$p_Y(\cdot|x^{(i)}) = \left(\prod_m p_Y(\cdot|z_m)\right)^{1/M} = \left(\prod_{z\in\mathcal{Z}^{(i)}} p_Y(\cdot|z)\right)^{1/M},$$

it can be shown that the difference between the true label distribution of the input that contains feature z and the true label distribution of feature z can be bounded. In specific, for $i \in \{i' | z \in \mathcal{Z}^{(i')}\}$, we have

$$\begin{aligned}
D_{\mathrm{KL}}\left(p_Y(\cdot|z) \middle\| p_Y(\cdot|x^{(i)})\right) &= -\sum_k p_Y(k|z) \log\left(\frac{p_Y(k|x^{(i)})}{p_Y(k|z)}\right) \\
&= -\frac{1}{M}\sum_k p_Y(k|z) \log\left(\prod_{z'\in\mathcal{Z}^{(i)}} \frac{p_Y(k|z')}{p_Y(k|z)}\right) \\
&= -\frac{1}{M}\sum_k p_Y(k|z) \sum_{z'\in\mathcal{Z}^{(i)}} \log\left(\frac{p_Y(k|z')}{p_Y(k|z)}\right) \\
&= -\frac{1}{M}\sum_{z'\in\mathcal{Z}^{(i)}} \sum_k p_Y(k|z) \log\left(\frac{p_Y(k|z')}{p_Y(k|z)}\right) \\
&= \frac{1}{M}\sum_{z'\in\mathcal{Z}^{(i)}} D_{\mathrm{KL}}\left(p_Y(\cdot|z) \,\| p_Y(\cdot|z')\right) \\
&= \frac{1}{M}\sum_{z'\in\mathcal{Z}^{(i)}} \psi\left(I(z, z')\right), \\
&\le \psi(\bar{I}(i, z)).
\end{aligned} \tag{14}$$

where $\bar{I}(i,z) = \frac{1}{M}\sum_{z' \in \mathcal{Z}^{(i)}} I(z, z')$ is the average PMI between feature z and other features z' in an input that contains z. Here we utilize Jensen's inequality on concave functions.

Now we can show that the average true label distribution of inputs that contain feature z approximates the true label distribution of feature z. Recall the definition of average true label distribution of inputs that contain feature z is $\bar{p}(\cdot|z) \coloneqq \frac{1}{N}\sum_{\{i|z\in\mathcal{Z}^{(i)}\}} p_Y(\cdot|x^{(i)})$. And we have

$$\begin{aligned}
\|\bar{p}(\cdot|z) - p_Y(\cdot|z)\|_1 &= \left\| \left(\frac{1}{N} \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} p_Y(\cdot|x^{(i)}) \right) - p_Y(\cdot|z) \right\|_1 \\
&\leq \frac{1}{N} \left\| \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} \left(p_Y(\cdot|x^{(i)}) - p_Y(\cdot|z) \right) \right\|_1 \\
&\leq \frac{1}{N} \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} \left\| p_Y(\cdot|x^{(i)}) - p_Y(\cdot|z) \right\|_1 \\
&\leq \frac{2^{1/2}}{N} \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} D_{\text{KL}}^{1/2} \left(p_Y(\cdot|z) \| \, p_Y(\cdot|x^{(i)}) \right) \\
&\leq \frac{2^{1/2}}{N} \sum_{\{i|z\in\mathcal{Z}^{(i)}\}} \psi^{1/2}(\bar{I}(i,z)) \\
&\leq \frac{2^{1/2} N_z}{N} \psi^{1/2} \left(\bar{I}(z) \right) \\
&\approx \frac{2^{1/2} M}{|\mathcal{Z}|} \psi^{1/2} \left(\bar{I}(z) \right),
\end{aligned} \tag{15}$$

where $\bar{I}(z) = \frac{1}{N_z} \min_{\{i|z\in\mathcal{Z}^{(i)}\}} \bar{I}(i,z)$ is the average PMI over all inputs that contain feature z.

□

B.5 Theorem 3.6

Proof. Recall the probabilistic output of the minimizer of the empirical risk is

$$f^*(x^{(i)}) = \left(\prod_m p_m^{(i)} \right)^{1/M} = \left(\prod_{z\in\mathcal{Z}^{(i)}} p_z \right)^{1/M}, \tag{16}$$

where p_z defined in Section B.2 is the network's probabilistic output of each feature z.

Note that for all k, $f(x^{(i)})(k) \leq 1$ and $p_Y(k|x^{(i)}) \leq 1$. Therefore,

$$\begin{aligned}
\left\| f^*(x^{(i)}) - p_Y(\cdot|x^{(i)}) \right\|_2 &\leq \exp\left(\left\| \log(f^*(x^{(i)})) - \log(p_Y(\cdot|x^{(i)})) \right\|_2 \right) \\
&= \exp\left(\frac{1}{M} \left\| \sum_{z \in \mathcal{Z}^{(i)}} (\log p_z - \log p_Y(\cdot|z)) \right\|_2 \right) \\
&\leq \exp\left(\frac{1}{M} \sum_{z \in \mathcal{Z}^{(i)}} \| \log p_z - \log p_Y(\cdot|z) \|_2 \right) \\
&\leq \exp\left(\frac{1}{M} \sum_{z \in \mathcal{Z}^{(i)}} \beta \log \| p_z - p_Y(\cdot|z) \|_2 \right) \\
&\leq \left(\max_{z \in \mathcal{Z}^{(i)}} \| p_z - p_Y(\cdot|z) \|_2 \right)^\beta \\
&= \left(\max_{z \in \mathcal{Z}^{(i)}} \| \bar{y}_z - p_Y(\cdot|z) \|_2 \right)^\beta \\
&= \left(\max_{z \in \mathcal{Z}^{(i)}} \left(\| \bar{y}_z - \bar{p}(\cdot|z) \|_2 + \| \bar{p}(\cdot|z) - p_Y(\cdot|z) \|_2 \right) \right)^\beta \\
&\leq \left(\max_{z \in \mathcal{Z}^{(i)}} \left(\| \bar{y}_z - \bar{p}(\cdot|z) \|_2 + \| \bar{p}(\cdot|z) - p_Y(\cdot|z) \|_1 \right) \right)^\beta \\
&= \left(\sqrt{\frac{(K-1)M}{N|\mathcal{Z}|} \frac{1}{\delta}} + \frac{2^{1/2} M}{|\mathcal{Z}|} \psi^{1/2}(\bar{I}_{\min}) \right)^\beta ,
\end{aligned} \tag{17}$$

where $\beta \coloneqq \min(\min_k f(x^{(i)})(k), \min_k p_Y(k|x^{(i)}))$ and $\bar{I}_{\min} = \min_{z \in \mathcal{Z}^{(i)}} \bar{I}(z)$.

□

B.6 Theorem A.3

Proof. We first present a Lemma similar to Lemma 3.3, which shows that the probabilistic predictions of features will still converge to the sample mean of the labels where the corresponding inputs contain this feature, up to some constant error.

Lemma B.4 (Convergence of the probabilistic predictions of features with Lipschitz-continuous and transformation-robust feature extractor)**.** *Let $\bar{y}_z \coloneqq \frac{1}{N} \sum_{\{i|z \in \mathcal{Z}^{(i)}\}} 1(y^{(i)})$, where $\mathcal{Z}^{(i)}$ denote the set of feature names in the i-th input, and thus $\{i|z \in \mathcal{Z}^{(i)}\}$ denotes the set of inputs that contain feature z. Let $f^* = \arg\min_f \hat{R}(f)$ and assume f_E^* is a L_X-Lipschitz-continuous and L_Γ-transformation-robust feature extractor. Let $p_{f^*}(x_z) \coloneqq$ Softmax$(w_C f_E^*(x_z))$. We have with probability $1-\delta$,*

$$\| p_{f^*}(x_z) - \bar{y}_z \| \leq L_z, \tag{18}$$

where $L_z \coloneqq 2L_\Gamma + L_X \tilde{O}(\frac{\nu_z}{\delta})$.

Proof. Since f_E^* is a L_X-Lipschitz-continuous and L_Γ-transformation-robust feature extractor, for any i, j, for any m, m' and any γ, γ', as long as $z_m = z_{m'} \equiv z$, we will have ,

$$\begin{aligned}
&\| f_E(x_m^{(i)}) - f_E(x_{m'}^{(j)}) \| \\
=&\| f_E(\gamma(x_z^{(i)})) - f_E(\gamma'(x_z^{(j)})) \| \\
\leq&\| f_E(\gamma(x_z^{(i)})) - f_E(x_z^{(i)}) \| + \| f_E(\gamma'(x_z^{(j)})) - f_E(x_z^{(j)}) \| + \| f_E(x_z^{(i)}) - f_E(x_z^{(j)}) \| \\
\leq&2L_\Gamma + L_X \| x_z^{(i)} - x_z^{(j)} \|
\end{aligned} \tag{19}$$

Now since the feature representation in the input space is always concentrated, by Chebyshev's inequality we will have with probability $1 - \delta$

$$\|x_z^{(i)} - x_z^{(j)}\| \leq \tilde{O}\left(\frac{\nu_z}{\delta}\right).$$

Therefore, we have with probability $1 - \delta$,

$$\|f_E(x_m^{(i)}) - f_E(x_{m'}^{(j)})\| \leq L_z,$$

where $L_z \coloneqq 2L_\Gamma + L_X \tilde{O}\left(\frac{\nu_z}{\delta}\right)$.

The empirical loss minimization now becomes

$$\begin{aligned} &\min_f -\frac{1}{MN} \sum_i 1(y^{(i)}) \cdot \sum_m \log p_m^{(i)} \\ &s.t. \quad \|p_m^{(i)} - p_{m'}^{(j)}\| \leq L_z, \text{ if } z_m = z_{m'}. \end{aligned} \tag{20}$$

Let $\mathcal{S}_z \coloneqq \{i | z \in \mathcal{Z}^{(i)}\}$ for simplicity, we can have

$$\begin{aligned} -\frac{1}{MN} \sum_i 1(y^{(i)}) \cdot \sum_m \log p_m^{(i)} &= -\frac{1}{MN} \sum_i 1(y^{(i)}) \cdot \sum_{z \in \mathcal{Z}^{(i)}} \log p_z^{(i)} \\ &= -\frac{1}{MN} \sum_{z \in \mathcal{Z}} \sum_{\{i | z \in \mathcal{Z}^{(i)}\}} 1(y^{(i)}) \cdot \log p_z^{(i)}, \\ &= -\frac{1}{MN} \sum_{z \in \mathcal{Z}} \sum_{i \in \mathcal{S}_z} 1(y^{(i)}) \cdot \log p_z^{(i)}, \end{aligned} \tag{21}$$

Note that this basically means that we assign the label of an example $y^{(i)}$ to each feature z contained in this example's input.

Therefore, (20) will be equivalent to the following problem.

$$\begin{aligned} &\min_{\{p_z^{(i)}\}} -\frac{1}{MN} \sum_{z \in \mathcal{Z}} \sum_{i \in \mathcal{S}_z} 1(y^{(i)}) \cdot \log p_z^{(i)} \\ &s.t. \quad \|p_z^{(i)} - p_z^{(j)}\| \leq L_z, \mathbf{1} \cdot p_z^{(i)} = 1, \ \forall \ z, \ \forall \ i, j \in \mathcal{S}_z, i \neq j. \end{aligned}$$

Note that the constraint $\|\cdot\| \leq L_z$ is only imposed in each subset $\mathcal{S}_z$.

Using the KKT condition, the above problem can be further formulated as follows.

$$\begin{aligned} &\min_{\{p_z^{(i)}\}} -\frac{1}{MN} \sum_{z \in \mathcal{Z}} \sum_{i \in \mathcal{S}_z} 1(y^{(i)}) \cdot \log p_z^{(i)} \\ &\quad + \sum_z \sum_{j,k \in \mathcal{S}_z, j \neq k} \mu_{z;jk} \left(\|p_z^{(j)} - p_z^{(k)}\| - L_z\right) \\ &\quad + \sum_z \sum_{l \in \mathcal{S}_z} \lambda_{z;l} (\mathbf{1} \cdot p_z^{(l)} - 1), \\ &s.t. \quad \mu_{z;jk}(\|p_z^{(j)} - p_z^{(k)}\| - L_z) = 0, \ \ \mu_{z;jk} \geq 0, \ \forall \ z, \ \forall \ j, k \in \mathcal{S}_z, j \neq k, \\ &\qquad \|p_z^{(j)} - p_z^{(k)}\| \leq L_z, \ \ \mathbf{1} \cdot p_z^{(l)} = 1, \ \forall \ z, \ \forall \ j, k, l \in \mathcal{S}_z, j \neq k. \end{aligned} \tag{22}$$

Using the first-order stationarity condition we have, $\forall \ z, \ \forall \ i \in \mathcal{S}_z$,

$$\nabla_{p_z^{(i)}} \cdot = -\frac{1}{MN} 1(y^{(i)}) \odot \frac{1}{p_z^{(i)}} + 2 \sum_{k \neq i} \mu_{ik} (p_z^{(i)} - p_z^{(k)}) + \lambda_i \mathbf{1} = 0, \tag{23}$$

where for simplicity we neglected the subscript z since the condition is the same for all z.

Sum (23) over i we have

$$-\frac{1}{MN} \sum_i 1(y^{(i)}) \odot \frac{1}{p_z^{(i)}} + 2 \sum_i \sum_{k \neq i} \mu_{ik} (p_z^{(i)} - p_z^{(k)}) + \sum_i \lambda_i \mathbf{1} = 0. \tag{24}$$

Note that $\sum_i \sum_{k\neq i} \mu_{ik}(p_z^{(i)} - p_z^{(k)}) = 0$ since each pair of i, k appears twice in the sum, where μ_{ik} is the same but the sign of $p_z^{(i)} - p_z^{(k)}$ is different. Therefore

$$-\frac{1}{MN}\sum_i 1(y^{(i)}) \odot \frac{1}{p_z^{(i)}} + \sum_i \lambda_i \mathbf{1} = 0. \tag{25}$$

Notice that to minimize the loss, if $y^{(i)} = y^{(j)}$, it is necessary that $p_z^{(i)} = p_z^{(j)}$.

Now we dot product both sides of (25) with $1(k)$, we have

$$-\frac{1}{MN}\frac{|\{i|y^{(i)} = k\}|}{p_z^{(i)}[k]} + \sum_i \lambda_i = 0,\ \forall\, i \in \{i|y^{(i)} = k\},$$

which means that

$$p_z^{(i)}[k] = \frac{|\{i|y^{(i)} = k\}|}{MN\sum_i \lambda_i},\ \text{if}\ y^{(i)} = k.$$

Now recall the constraint that when $i \neq j$,

$$\|p_z^{(i)} - p_z^{(j)}\| \leq L_z.$$

This at least indicates that

$$\|p_z^{(i)} - p_z^{(j)}\|_\infty \leq L_z.$$

Therefore,

$$p_z^{(i)}[k'] - \frac{|\{i|y^{(i)} = k'\}|}{MN\sum_i \lambda_i} \leq L_z,\ \text{if}\ k' \neq y^{(i)}.$$

This implies that

$$\left\| p_z^{(i)} - \frac{1}{MN\sum_i \lambda_i}\sum_i 1(y^{(i)}) \right\| \leq L_z$$

Now consider the condition $\mathbf{1} \cdot p_z^{(i)} = 1$, we will have

$$\sum_i \lambda_i = \frac{N_z}{MN}.$$

Thus

$$\left\| p_z^{(i)} - \bar{y}_z \right\| \leq L_z,$$

where we recall $\bar{y}_z = \frac{1}{N_z}\sum_i 1(y^{(i)})$. □

Now to prove Theorem A.3, we only need to combine Lemma B.4 and Lemmas 3.4, 3.5, where the reasoning is exactly same as the proof of Theorem 3.6.

□

C Limitations

In this paper we focus on the theoretical feasibility of learning the true label distribution of training examples with empirical risk minimization. Therefore we only analyze the existence of such a desired minimizer, but neglect the optimization process to achieve it. By explore the optimization towards true label distribution, potentially more dynamics can be found to inspire new regularization techniques.

Also, our proposed method for training a student-oriented teacher may not be able to advance the state-of-the-art significantly, as the regularization techniques based inspired by our analyses (e.g., Lipschitz regularization and consistency regularization) may more or less be leveraged by existing training practice of deep neural networks, either implicitly or explicitly.

D *SoTeacher*

Based on our theoretical findings, we investigate practical techniques for training the teacher model to more accurately approximate the true label distribution of the training data.

Lipschitz regularization. Theorem A.3 suggests that it is necessary to enforce the feature extractor to be Lipschitz continuous. This may also be achieved by current teacher training practice, as neural networks are often implicitly Lipschitz bounded (Bartlett et al., 2017). Nevertheless, we observe that explicit Lipschitz regularization (LR) can still help in multiple experiments (See Sect. F). Therefore, we propose to incorporate a global Lipschitz constraint into teacher's training. For the implementation details, we follow the existing practice of Lipschitz regularization (Yoshida & Miyato, 2017; Miyato et al., 2018) and defer them to Appendix E.

Consistency regularization. Theorem A.3 also shows that to learn the true label distribution, it is necessary to ensure the feature extractor is robust to feature transformations. This is aligned with the standard practice which employs data augmentation as regularization. However, when data is scarce, it is better to explicitly enforce the model to be robust to transformations. This is known as consistency regularization (CR) (Laine & Aila, 2017; Xie et al., 2020; Berthelot et al., 2019; Sohn et al., 2020) that is widely used in semi-supervised learning.

Considering the training efficiency of consistency regularization, we utilize temporal ensembling (Laine & Aila, 2017), which penalizes the difference between the current prediction and the aggregation of previous predictions for each training input. In this way, the consistency under data augmentation is implicitly regularized since the augmentation is randomly sampled in each epoch. And it is also efficient as no extra model evaluation is required for an input.

To scale the consistency loss, a loss weight is often adjusted based on a Gaussian ramp-up curve in previous works (Laine & Aila, 2017). However, the specific parameterization of such a Gaussian curve varies greatly across different implementations, where more than one additional hyperparameters have to be set up and tuned heuristically. Here to avoid tedious hyperparameter tuning we simply linearly interpolate the weight from 0 to its maximum value, namely $\lambda_{\text{CR}}(t) = \frac{t}{T}\lambda_{\text{CR}}^{\max}$, where T is the total number of epochs.

Summary. To recap, our teacher training method introduces two additional regularization terms. The loss function can thus be defined as $\ell = \ell_{\text{Stand.}} + \lambda_{\text{LR}}\ell_{\text{LR}} + \lambda_{\text{CR}}\ell_{\text{CR}}$, where $\ell_{\text{Stand.}}$ is the standard empirical risk $\hat{R}(f)$ and λ_{LR} is the weight for Lipschitz regularization. Our method is simple to implement and incurs only minimal computation overhead (see Section F.2).

E Implementation details

Lipschitz regularization. Following previous practice using Lipschitz regularization for generalization on unseen data (Yoshida & Miyato, 2017) or stabilizing generative model (Miyato et al., 2018), we regularize the Lipschitz constant of a network by constraining the Lipschitz constant of each trainable component. The regularization term is thus defined as $\ell_{\text{LR}} = \sum_f \text{Lip}(f)$, where f denotes a trainable component in the network $\boldsymbol{f}$. The Lipschitz constant of a network component $\text{Lip}(f)$ induced by a norm $\|\cdot\|$ is the smallest value L such that for any input features h, h', $\|f(h) - f(h')\| \leq L\|h - h'\|$. Here we adopt the Lipschitz constant induced by 1-norm, since its calculation is accurate, simple and efficient. For calculating the Lipschitz constants of common trainable components in deep neural networks, we refer to (Gouk et al., 2021) for a comprehensive study.

Consistency regularization. We design our consistency regularization term as $\ell_{\text{CR}} = \frac{1}{N}\sum_i \left\|\boldsymbol{f}(x_i) - \overline{\boldsymbol{f}(x_i)}\right\|_2^2$, where we follow previous work (Laine & Aila, 2017) and employ MSE to penalize the difference. Here $\overline{\boldsymbol{f}(x)}$ is the aggregated prediction of an input x, which we calculate as the simple average of previous predictions $\overline{\boldsymbol{f}(x)}_t = \frac{1}{t}\sum_{t'=0}^{t-1} \boldsymbol{f}(x)_{t'}$, where we omit the data augmentation operator for simplicity. At epoch 0 we simply skip the consistency regularization. Note that such a prediction average can be implemented in an online manner thus there is no need to store every previous prediction of an input.

F Experiments

In this section, we evaluate the effectiveness of our teacher training method in knowledge distillation. We focus on compressing a large network to a smaller one where the student is trained on the same data set as the teacher.

F.1 Experiment setup

We denote our method as student-oriented teacher training (*SoTeacher*), since it aims to learn the true label distribution to improve student performance, rather than to maximize teacher performance. We compare our method with the standard practice for teacher training in knowledge distillation (*Standard*). We conduct experiments on benchmark datasets including CIFAR-100 (Krizhevsky, 2009), Tiny-ImageNet (Tin, 2017), and ImageNet (Deng et al., 2009). We experiment with various backbone networks including ResNet (He et al., 2016), Wide ResNet (Zagoruyko & Komodakis, 2016) and ShuffleNet (Zhang et al., 2018b; Tan et al., 2019). We test the applicability of *SoTeacher* from different aspects of model compression in knowledge distillation including reduction of the width or depth, and distillation between heterogeneous neural architectures.

For knowledge distillation algorithms, we experiment with the original knowledge distillation method (KD) (Hinton et al., 2015), and a wide variety of other sophisticated knowledge distillation algorithms (see Sect. F.2). We report the classification accuracies on the test set of both teacher and the student distilled from it. All results except ImageNet are presented with mean and standard deviation based on 3 independent runs. For Tiny-ImageNet, we also report the top-5 accuracy. For hyperparameters, we set $\lambda_{\text{CR}}^{\max} = 1$ for both datasets, $\lambda_{\text{LR}} = 10^{-5}$ for CIFAR-100 and $\lambda_{\text{LR}} = 10^{-6}$ for Tiny-ImageNet/ImageNet. More detailed hyperparameter settings for neural network training and knowledge distillation can be found in Appendix G.

Table 1: Test accuracy of the teacher and student with knowledge distillation conducted on CIFAR-100. *SoTeacher* achieves better student accuracy than Standard for various architectures, depsite a lower teacher accuracy.

	WRN40-2/WRN40-1		WRN40-2/WRN16-2		ResNet32x4/ShuffleNetV2	
	Student	Teacher	**Student**	Teacher	**Student**	Teacher
Standard	73.73 ± 0.13	76.38 ± 0.13	74.87 ± 0.45	76.38 ± 0.13	74.86 ± 0.18	79.22 ± 0.03
SoTeacher	$\mathbf{74.35} \pm 0.23$	74.95 ± 0.28	$\mathbf{75.39} \pm 0.23$	74.95 ± 0.28	$\mathbf{77.24} \pm 0.09$	78.49 ± 0.09
no-CR	74.34 ± 0.11	74.30 ± 0.12	75.20 ± 0.24	74.30 ± 0.12	76.52 ± 0.52	77.73 ± 0.17
no-LR	73.81 ± 0.15	76.71 ± 0.16	75.21 ± 0.13	76.71 ± 0.16	76.23 ± 0.18	80.01 ± 0.18

Table 2: Test accuracy of the teacher and student with knowledge distillation conducted on Tiny-ImageNet and ImageNet. The student network is ResNet18, while the teacher network is ResNet34 for Tiny-ImageNet and ResNet152 for ImageNet. Due to computation constraints, we are not able to perform ablation experiments on ImageNet.

	Tiny-ImageNet				ImageNet	
	Student (Top-1)	**Student** (Top-5)	Teacher (Top-1)	Teacher (Top-5)	**Student** (Top-1)	Teacher (Top-1)
Standard	66.19 ± 0.17	85.74 ± 0.21	64.94 ± 0.32	84.33 ± 0.40	71.30	77.87
SoTeacher	$\mathbf{66.83} \pm 0.20$	86.19 ± 0.22	64.88 ± 0.48	84.91 ± 0.41	71.45	77.11
no-CR	66.39 ± 0.27	86.05 ± 0.17	64.36 ± 0.43	84.10 ± 0.27	-	-
no-LR	66.48 ± 0.43	$\mathbf{86.20} \pm 0.40$	64.26 ± 1.54	84.48 ± 0.84	-	-

F.2 Results

End-to-end Knowledge Distillation Performance. Tables 1 and 2 show the evaluation results on CIFAR-100 and Tiny-ImageNet/ImageNet, respectively. Our teacher training method SoTeacher can improve the student's test accuracy consistently across different datasets and teacher/student architecture pairs. Note that the success of our teacher training method is not due to the high accuracy of the teacher. In Tables 1 and 2, one may already notice that our regularization method will hurt the accuracy of the teacher, despite that it can improve the accuracy of the student distilled from it.

On ImageNet particularly, the gain in student's accuracy by *SoTeacher* is minor, potentially because the large size of the dataset can already facilitate the learning of true label distribution as suggested by our theory. Nevertheless, one may observe that the teacher's accuracy is significantly lower when using *SoTeacher*, which implicitly suggests our teacher regularization method is tailored towards the student's accuracy.

Table 3: Estimation of the uncertainty quality of the teacher network trained by Standard and *SoTeacher*. The uncertainty quality is estimated by the ECE and NLL both before and after temperature scaling (TS).

Dataset	Teacher	Method	ECE	NLL	ECE (w/ TS)	NLL (w/ TS)
CIFAR-100	WRN40-2	Standard	0.113 ± 0.003	1.047 ± 0.007	0.028 ± 0.004	0.905 ± 0.008
		SoTeacher	0.057 ± 0.002	0.911 ± 0.013	0.016 ± 0.003	0.876 ± 0.012
CIFAR-100	ResNet32x4	Standard	0.083 ± 0.003	0.871 ± 0.010	0.036 ± 0.001	0.815 ± 0.008
		SoTeacher	0.037 ± 0.001	0.777 ± 0.006	0.021 ± 0.001	0.764 ± 0.003
Tiny-ImageNet	ResNet34	Standard	0.107 ± 0.007	1.601 ± 0.037	0.043 ± 0.002	1.574 ± 0.015
		SoTeacher	0.070 ± 0.007	1.496 ± 0.031	0.028 ± 0.002	1.505 ± 0.033

Ablation Study. We toggle off the Lipschitz regularization (LR) or consistency regularization (CR) in *SoTeacher* (denoted as *no-LR* and *no-CR*, respectively) to explore their individual effects. As shown in Tables 1 and 2, LR and CR can both improve the performance individually. But on average, *SoTeacher* achieves the best performance when combining both LR and CR, as also demonstrated in our theoretical analyses. Note that in Table 2, using Lipschitz regularization is not particularly effective because the regularization weight might not be properly tuned (see Figure 1).

Quality of True Distribution Approximation. To further interpret the success of our teacher training method, we show that our regularization can indeed improve the approximation of the true label distribution thus benefiting the student generalization. Directly measuring the quality of the true distribution approximation is infeasible as the true distribution is unknown for realistic datasets. Follow previous works (Menon et al., 2021), we instead *estimate* the approximation quality by reporting the Expected Calibration Error (ECE) (Guo et al., 2017) and NLL loss of the teacher on a holdout set with one-hot labels. Since scaling the teacher predictions in knowledge distillation can improve the uncertainty quality (Menon et al., 2021), we also report ECE and NLL after temperature scaling, where the optimal temperature is located on an additional holdout set (Guo et al., 2017). As shown in Table 3, our teacher training method can consistently improve the approximation quality for different datasets and teacher architectures.

Effect of Hyperparameters. We conduct additional experiments on Tiny-ImageNet as an example to study the effect of two regularization terms introduced by our teacher training method. For Lipschitz regularization, we modulate the regularization weight λ_{LR}. For consistency regularization, we try different maximum regularization weights $\lambda_{\text{CR}}^{\max}$ and different weight schedulers including linear, cosine, cyclic, and piecewise curves. Detailed descriptions of these schedulers can be found in Appendix G. As shown in Figure 1, both Lipschitz and consistency regularizations can benefit the teacher training in terms of the student generalization consistently for different hyperparameter settings. This demonstrates that our regularizations are not sensitive to hyperparameter selection. Note that the hyperparameter chosen to report the results in Tables 1 and 2 might not be optimal since we didn't perform extensive hyperparameter search in fear of overfitting small datasets. It is thus possible to further boost the performance by careful hyperparameter tuning.

In particular, Figure 1(a) shows that, as Lipschitz regularization becomes stronger, the teacher accuracy constantly decreases while the student accuracy increases and converges. This demonstrates that excessively strong Lipschitz regularization hurts the performance of neural network training, but it can help student generalization in the knowledge distillation context.

Other Knowledge Distillation Algorithms. Besides the original knowledge distillation algorithm, we experiment with various feature distillation algorithms including FitNets (Romero et al., 2015), AT (Zagoruyko & Komodakis, 2017), SP (Tung & Mori, 2019), CC (Peng et al., 2019), VID (Ahn et al., 2019), RKD (Park et al., 2019), PKT (Passalis & Tefas, 2018), AB (Heo et al., 2019), FT (Kim et al., 2018), NST (Huang & Wang, 2017), CRD (Tian et al., 2020) and SSKD (Xu et al., 2020). For the implementation of these algorithms, we refer to existing repositories for knowledge distillation (Tian et al., 2020; Shah et al., 2020; Matsubara, 2021) and author-provided codes. Although these distillation

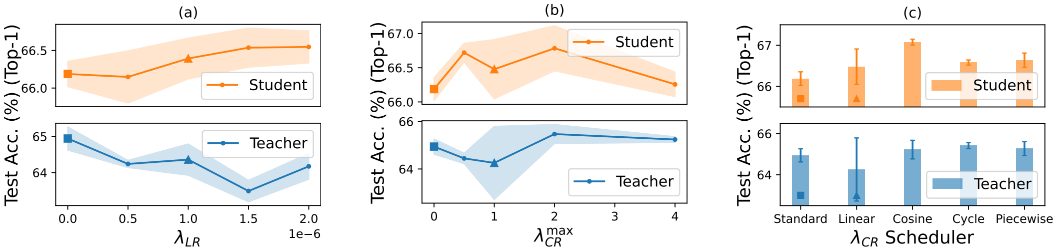

Figure 1: Effect of varying the hyperparameters in our teacher training method, including the weight for Lipschitz regularization λ_{LR}, the weight for consistency regularization λ_{CR}, and its scheduler. The settings of our method used to report the results (e.g. Table 2) are denoted as "▲". The standard teacher training practice is denoted as "■" for comparison.

algorithms match all types of features instead of predictions between teacher and student, they will achieve the best distillation performance when combined with the prediction distillation (*i.e.* original KD). Therefore, our teacher training method should still benefit the effectiveness of these distillation algorithms. We also experiment with a curriculum distillation algorithm RCO (Jin et al., 2019) which distills from multiple checkpoints in teacher's training trajectory. Our teacher training method should also benefit RCO as the later teacher checkpoints become more student-oriented. As shown in Table 6, our *SoTeacher* can boost the distillation performance of almost all these distillation algorithms, demonstrating its wide applicability.

Student fidelity. Recent works have underlined the importance of student fidelity in knowledge distillation, namely the ability of the student to match teacher predictions (Stanton et al., 2021). Student fidelity can be viewed as a measure of knowledge distillation effectiveness that is orthogonal to student generalization, as the student is often unable to match the teacher predictions although its accuracy on unseen data improves (Furlanello et al., 2018; Mobahi et al., 2020). Here we measure the student fidelity by the average agreement between the student and teacher's top-1 predicted labels on the test set. As shown in Table 4, our teacher training method can consistently and significantly improve the student fidelity for different datasets and teacher-student pairs, which aligns with the improvement of student generalization shown by Table 1 and 2. This demonstrates that the teacher can better transfer its "knowledge" to the student with our training method.

Table 4: Average agreement (%) between the student and teacher's top-1 predictions on the test set.

	CIFAR-100			Tiny-ImageNet
	WRN-40-2/WRN-40-1	WRN-40-2/WRN-16-2	ResNet32x4/ShuffleNetV2	ResNet34/ResNet18
Standard	76.16 ± 0.14	76.92 ± 0.29	76.63 ± 0.25	71.33 ± 0.07
SoTeacher	$\mathbf{77.92} \pm 0.27$	$\mathbf{79.41} \pm 0.11$	$\mathbf{80.36} \pm 0.13$	$\mathbf{73.36} \pm 0.25$

G Details of experiment setting

G.1 Hyperparameter setting for teacher network training

For all the experiments on CIFAR-100, we employ SGD as the optimizer and train for 240 epochs with a batch size of 64. The learning rate is initialized at 0.05 and decayed by a factor of 10 at the epochs 150, 180 and 210, with an exception for ShuffleNet where the learning rate is initialized at 0.01 following existing practice (Tian et al., 2020; Park et al., 2021). The weight decay and momentum are fixed as 0.0005 and 0.9 respectively. The training images are augmented with random cropping and random horizontal flipping with a probability of 0.5.

For Tiny-ImageNet experiments, we employ SGD as the optimizer and conduct the teacher training for 90 epochs with a batch size of 128. The learning rate starts at 0.1 and is decayed by a factor of 10 at epochs 30 and 60. The weight decay and momentum are fixed as 0.0005 and 0.9 respectively. The training images are augmented with random rotation with a maximum degree of 20, random cropping and random horizontal flipping with a probability of 0.5. For student training the only difference is

Table 5: β for different feature distillation algorithms

	β			β			β	
	Standard	SoTeacher		Standard	SoTeacher		Standard	SoTeacher
FitNets	100	50	AT	1000	500	SP	3000	1500
CC	0.02	0.01	VID	1.0	0.5	RKD	1.0	0.5
PKT	30000	15000	AB	1.0	0.5	FT	200	100
NST	50	25	CRD	0.8	0.5			

that we train for additional 10 epochs, with one more learning rate decay at epoch 90, aligned with previous settings (Tian et al., 2020).

For consistency regularization in our teacher training method, we experiment with various weight schedules besides the linear schedule mentioned in the main paper. We list the formulas for these schedules in the following. Here t denotes the epoch number, T denotes the total number of epochs, and $\lambda_{CR}^{\max}$ denotes the maximum weight.

- Cosine schedule:
$$\lambda_{CR}(t) = \cos\left[\left(1 - \frac{t}{T}\right)\frac{\pi}{2}\right]\lambda_{CR}^{\max}$$
- Cyclic schedule:
$$\lambda_{CR}(t) = \sqrt{1 - \left(1 - \frac{t}{T}\right)^2}\lambda_{CR}^{\max}$$
- Piecewise schedule:
$$\lambda_{CR}(t) = \begin{cases} 0, & 0 < t \leq T/3, \\ \lambda_{CR}^{\max}/2, & T/3 < t \leq 2T/3, \\ \lambda_{CR}^{\max}, & 2T/3 < t \leq T. \end{cases}$$

G.2 Hyperparameter setting for knowledge distillation algorithms

For knowledge distillation algorithms we refer to the setting in RepDistiller [8]. Specifically, for original KD, the loss function used for student training is defined as

$$\ell = \alpha\ell_{\text{Cross-Entropy}} + (1-\alpha)\ell_{KD}.$$

We grid search the best hyper-parameters that achieve the optimal performance, namely the loss scaling ratio α is set as 0.5 and the temperature is set as 4 for both CIFAR-100 and Tiny-ImageNet. For all feature distillation methods combined with KD the loss function can be summarized as (Tian et al., 2020)

$$\ell = \gamma\ell_{\text{Cross-Entropy}} + \alpha\ell_{KD} + \beta\ell_{\text{Distill}},$$

where we grid search the optimal γ and α to be 1.0 and 1.0 respectively. When using our teacher training method, all these hyperparameters are kept same except that for all feature distillation algorithms the scaling weights corresponding to the feature distillation losses β are cut by half, as we wish to rely more on the original KD that is well supported by our theoretical understanding. Table 5 list β used in our experiments for all feature distillation algorithms. For SSKD (Xu et al., 2020) the hyperparameters are set as $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 2.7$, $\lambda_4 = 10.0$ for standard training and $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$, $\lambda_4 = 10.0$ for our methods. For the curriculum distillation algorithm RCO we experiment based on one-stage EEI (equal epoch interval). We select 24 anchor points (or equivalently every 10 epochs) from the teacher's saved checkpoints.

H Additional experiment results

Training overhead. Compared to standard teacher training, the computation overhead of *SoTeacher* is mainly due to the calculation of the Lipschitz constant, which is efficient as it only requires simple arithmetic calculations of the trainable weights of a neural network (see Section D). Empirically

[8] `https://github.com/HobbitLong/RepDistiller`

Table 6: *SoTeacher* consistently outperforms Standard on CIFAR-100 with various KD algorithms.

	WRN40-2/WRN40-1		WRN40-2/WRN16-2		ResNet32x4/ShuffleV2	
	Standard	*SoTeacher*	Standard	*SoTeacher*	Standard	*SoTeacher*
FitNet	74.06 ± 0.20	$\mathbf{74.88} \pm 0.15$	75.42 ± 0.38	$\mathbf{75.64} \pm 0.20$	76.56 ± 0.15	$\mathbf{77.91} \pm 0.21$
AT	73.78 ± 0.40	$\mathbf{75.12} \pm 0.17$	75.45 ± 0.28	$\mathbf{75.88} \pm 0.09$	76.20 ± 0.16	$\mathbf{77.93} \pm 0.15$
SP	73.54 ± 0.20	$\mathbf{74.71} \pm 0.19$	74.67 ± 0.37	$\mathbf{75.94} \pm 0.20$	75.94 ± 0.16	$\mathbf{78.06} \pm 0.34$
CC	73.46 ± 0.12	$\mathbf{74.76} \pm 0.16$	75.08 ± 0.07	$\mathbf{75.67} \pm 0.39$	75.43 ± 0.19	$\mathbf{77.68} \pm 0.28$
VID	73.88 ± 0.30	$\mathbf{74.89} \pm 0.19$	75.11 ± 0.07	$\mathbf{75.71} \pm 0.19$	75.95 ± 0.11	$\mathbf{77.57} \pm 0.16$
RKD	73.41 ± 0.47	$\mathbf{74.66} \pm 0.08$	75.16 ± 0.21	$\mathbf{75.59} \pm 0.18$	75.28 ± 0.11	$\mathbf{77.46} \pm 0.10$
PKT	74.14 ± 0.43	$\mathbf{74.89} \pm 0.16$	75.45 ± 0.09	$\mathbf{75.53} \pm 0.09$	75.72 ± 0.18	$\mathbf{77.84} \pm 0.03$
AB	73.93 ± 0.35	$\mathbf{74.86} \pm 0.10$	70.09 ± 0.66	$\mathbf{70.38} \pm 0.87$	76.27 ± 0.26	$\mathbf{78.05} \pm 0.21$
FT	73.80 ± 0.15	$\mathbf{74.75} \pm 0.13$	75.19 ± 0.15	$\mathbf{75.68} \pm 0.28$	76.42 ± 0.17	$\mathbf{77.56} \pm 0.15$
NST	73.95 ± 0.41	$\mathbf{74.74} \pm 0.14$	74.95 ± 0.23	$\mathbf{75.68} \pm 0.16$	76.07 ± 0.08	$\mathbf{77.71} \pm 0.10$
CRD	74.44 ± 0.11	$\mathbf{75.06} \pm 0.37$	75.52 ± 0.12	$\mathbf{75.95} \pm 0.02$	76.28 ± 0.13	$\mathbf{78.09} \pm 0.13$
SSKD	75.82 ± 0.22	$\mathbf{75.94} \pm 0.18$	76.31 ± 0.07	$\mathbf{76.32} \pm 0.09$	78.49 ± 0.10	$\mathbf{79.37} \pm 0.11$
RCO	74.50 ± 0.32	$\mathbf{74.81} \pm 0.04$	75.24 ± 0.34	$\mathbf{75.50} \pm 0.12$	76.75 ± 0.13	$\mathbf{77.59} \pm 0.31$

Table 7: Performance of the knowledge distillation when training the teacher using existing regularization methods for learning quality uncertainty on unseen data.

	WRN40-2/WRN40-1	
	Student	Teacher
Standard	73.73 ± 0.13	76.38 ± 0.13
SoTeacher	$\mathbf{74.35} \pm 0.23$	74.95 ± 0.28
ℓ_2 (5×10^{-4})	73.73 ± 0.13	76.38 ± 0.13
ℓ_1 (10^{-5})	73.60 ± 0.15	73.52 ± 0.05
Mixup ($\alpha = 0.2$)	73.19 ± 0.21	77.30 ± 0.20
Cutmix ($\alpha = 0.2$)	73.61 ± 0.26	78.42 ± 0.07
Augmix ($\alpha = 1, k = 3$)	73.83 ± 0.09	77.80 ± 0.30
CRL ($\lambda = 1$)	74.13 ± 0.29	76.69 ± 0.16

we observe that training with *SoTeacher* is only slightly longer than the standard training for about 5%. The memory overhead of *SoTeacher* is incurred by buffering an average prediction for each input. However, since such prediction requires no gradient calculation we can simply store it in a memory-mapped file.

I Experiments with uncertainty regularization methods on unseen data

Uncertainty learning on unseen data. Since the objective of our student-oriented teacher training is to learn label distributions of the training data, it is related to those methods aiming to learn quality uncertainty on the unseen data. We consider those methods that are feasible for large teacher network training, including (1) classic approaches to overcome overfitting such as ℓ_1 and ℓ_2 regularization, (2) modern regularizations such as label smoothing (Szegedy et al., 2016) and data augmentations such as mixup (Zhang et al., 2018a) and Augmix (Hendrycks et al., 2020), (3) Post-training methods such as temperature scaling (Guo et al., 2017), as well as (4) methods that incorporate uncertainty as a learning ojective such as confidence-aware learning (CRL) (Moon et al., 2020).

We have conducted experiments on CIFAR-100 using all these methods and the results can be found in Appendix I. Unfortunately, the performance of these regularization methods is unsatisfactory in knowledge distillation — only CRL can slightly outperform the standard training. We believe the reasons might be two-folds. First, most existing criteria for uncertainty quality on the unseen data such as calibration error (Naeini et al., 2015) or ranking error (Geifman et al., 2019), only require the model to output an uncertainty estimate that is correlated with the probability of prediction errors. Such criteria may not be translated into the approximation error to the true label distribution. Second, even if a model learns true label distribution on unseen data, it does not necessarily have to learn true label distribution on the training data, as deep neural networks tend to memorize the training data.

Experiment setup. We conduct experiments on CIFAR-100 with teacher-student pair WRN40-2/WRN40-1. We employ the original KD as the distillation algorithm. The hyperparameter settings are the same as those mentioned in the main results (see Appendix G). For each regularization method we grid search the hyperparameter that yields the best student performance. The results are summarized in Table 7.

Classic regularization. We observe that with stronger ℓ_2 or ℓ_1 regularization the student performance will not deteriorate significantly as teacher converges. However, it also greatly reduces the performance of the teacher. Subsequently the performance of the student is not improved as shown in Table 7.

Label smoothing. Label smoothing is shown to not only improve the performance but also the uncertainty estimates of deep neural networks (Müller et al., 2019). However, existing works have already shown that label smoothing can hurt the effectiveness of knowledge distillation (Müller et al., 2019), thus we neglect the results here. An intuitive explanation is that label smoothing encourages the representations of samples to lie in equally separated clusters, thus "erasing" the information encoding possible secondary classes in a sample (Müller et al., 2019).

Data augmentation. Previous works have demonstrated that mixup-like data augmentation techniques can greatly improve the uncertainty estimation on unseen data (Thulasidasan et al., 2019; Hendrycks et al., 2020). For example, Mixup augmented the training samples as $x := \alpha x + (1-\alpha)x'$, and $y := \alpha y + (1-\alpha)y'$, where (x', y') is a randomly drawn pair not necessarily belonging to the same class as x.

As shown in Table 7, stronger mixup can improve the performance of the teacher, whereas it can barely improve or even hurt the performance of the student. Based on our theoretical understanding of knowledge distillation, we conjecture the reason might be that mixup distorts the true label distribution of an input stochastically throughout the training, thus hampering the learning of true label distribution.

Temperature scaling. Previous works have suggested using the uncertainty on a validation set to tune the temperature for knowledge distillation either in standard learning (Menon et al., 2021) or robust learning (Dong et al., 2021). However, the optimal temperature may not be well aligned with that selected based on uncertainty (Menon et al., 2021). We neglect the experiment results here as the distillation temperature in our experiments is already fine-tuned.

Uncertainty learning. CRL designs the loss function as $\ell = \ell_{\text{CE}} + \lambda \ell_{\text{CRL}}$, where ℓ_{CE} is the cross-entropy loss and ℓ_{CRL} is an additional regularization term bearing the form of

$$\ell_{\text{CRL}} = \max\left(0, -g(c(x_i), c(x_j))(p(x_i) - p(x_j)) + |c(x_i) - c(x_j)|\right), \tag{26}$$

where $p(x) = \max_k \mathbf{f}(x)^k$ is the maximum probability of model's prediction on a training sample x and

$$c(x) = \frac{1}{t-1} \sum_{t'=1}^{t-1} 1(\arg\max_k \mathbf{f}(x)_{t'}^k = y)$$

is the frequency of correct predictions through the training up to the current epoch. Here $g(c_i, c_j) = 1$ if $c_i > c_j$ and $g(c_i, c_j) = -1$ otherwise. Although originally proposed to improve the uncertainty quality of deep neural networks in terms of ranking, we found that CRL with a proper hyperparameter can improve the distillation performance, as shown in Table 7.

We note that the effectiveness of CRL on distillation can be interpreted by our theoretical understanding, as its regularization term (26) is essentially a special form of consistency regularization. To see this we first notice (26) is a margin loss penalizing the difference between $p(x)$ and $c(x)$ in terms of ranking. We then rewrite $c(x)$ as

$$c(x) = \mathbf{1}_y \cdot \frac{1}{t-1} \sum_{t'=1}^{t-1} \text{Onehot}[\mathbf{f}(x)_{t'}], \tag{27}$$

which is similar to our consistency regularization target, except the prediction is first converted into one-hot encoding. Such variation may not be the best for knowledge distillation as we wish those secondary class probabilities in the prediction be aligned as well.

J Related Work

Understand knowledge distillation. There exist multiple perspectives in understanding the effectiveness of knowledge distillation. Besides the statistical perspective which views the soft prediction of the teacher as an approximation of the true label distribution (Menon et al., 2021; Dao et al., 2021), another line of work understands knowledge distillation from a regularization perspective, which views the teacher's soft predictions as instance-wise label smoothing (Yuan et al., 2020; Zhang & Sabuncu, 2020; Tang et al., 2020). More recently, (Allen-Zhu & Li, 2020) understands knowledge distillation from a feature learning perspective by focusing on the data that possesses a "multi-view" structure, that multiple features co-exist in the input and can be used to make the correct classification. Knowledge distillation is effective because the teacher can learn different features and transfer them to the student. Our theory is built on a similar assumption on the data structure, albeit we require the teacher to learn the true label distribution of the feature. Our theory thus can also be viewed as a bridge between the statistical perspective and feature learning perspective.

Alleviate "teacher overfitting". Since in knowledge distillation, the distillation data is often reused from the teacher's training data, a teacher trained toward convergence is very likely to overfit its soft predictions on the distillation data. Intuitively, it is possible to tackle this problem by early stopping the teacher training (Cho & Hariharan, 2019). However, a meticulous hyperparameter search may be required since the epoch number to find the best checkpoint is often sensitive to the specific training setting such as the learning rate schedule. It is also possible to save multiple early teacher checkpoints for the student to be distilled from sequentially (Jin et al., 2019). Additionally, one can utilize a "cross-fitting" procedure to prevent the teacher from memorizing the training data. Namely, the training data is first partitioned into several folds, where the teacher predictions on each fold are generated by the teacher trained only on out-of-fold data (Dao et al., 2021). One can also train the teacher network jointly with student's network blocks, which imposes a regularization toward the student performance (Park et al., 2021). Different from these attempts, we train the teacher to directly learn the true label distribution of its training data, leading to a simple and practical student-oriented teacher training framework with minimum computation overhead.