When Does Label Smoothing Help?

Rafael Müller,[*] Simon Kornblith, Geoffrey Hinton
Google Brain
Toronto
rafaelmuller@google.com

Abstract

The generalization and learning speed of a multi-class neural network can often be significantly improved by using soft targets that are a weighted average of the hard targets and the uniform distribution over labels. Smoothing the labels in this way prevents the network from becoming over-confident and label smoothing has been used in many state-of-the-art models, including image classification, language translation and speech recognition. Despite its widespread use, label smoothing is still poorly understood. Here we show empirically that in addition to improving generalization, label smoothing improves model calibration which can significantly improve beam-search. However, we also observe that if a teacher network is trained with label smoothing, knowledge distillation into a student network is much less effective. To explain these observations, we visualize how label smoothing changes the representations learned by the penultimate layer of the network. We show that label smoothing encourages the representations of training examples from the same class to group in tight clusters. This results in loss of information in the logits about resemblances between instances of different classes, which is necessary for distillation, but does not hurt generalization or calibration of the model's predictions.

1 Introduction

It is widely known that neural network training is sensitive to the loss that is minimized. Shortly after Rumelhart et al. [1] derived backpropagation for the quadratic loss function, several researchers noted that better classification performance and faster convergence could be attained by performing gradient descent to minimize cross entropy [2, 3]. However, even in these early days of neural network research, there were indications that other, more exotic objectives could outperform the standard cross entropy loss [4, 5]. More recently, Szegedy et al. [6] introduced label smoothing, which improves accuracy by computing cross entropy not with the "hard" targets from the dataset, but with a weighted mixture of these targets with the uniform distribution.

Label smoothing has been used successfully to improve the accuracy of deep learning models across a range of tasks, including image classification, speech recognition, and machine translation (Table 1). Szegedy et al. [6] originally proposed label smoothing as a strategy that improved the performance of the Inception architecture on the ImageNet dataset, and many state-of-the-art image classification models have incorporated label smoothing into training procedures ever since [7, 8, 9]. In speech recognition, Chorowski and Jaitly [10] used label smoothing to reduce the word error rate on the WSJ dataset. In machine translation, Vaswani et al. [11] attained a small but important improvement in BLEU score, despite a reduction in perplexity.

Although label smoothing is a widely used "trick" to improve network performance, not much is known about why and when label smoothing should work. This paper tries to shed light upon behavior

[*] This work was done as part of the Google AI Residency.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Table 1: Survey of literature label smoothing results on three supervised learning tasks.

DATA SET	ARCHITECTURE	METRIC	VALUE W/O LS	VALUE W/ LS
IMAGENET	INCEPTION-V2 [6]	TOP-1 ERROR	23.1	**22.8**
		TOP-5 ERROR	6.3	**6.1**
EN-DE	TRANSFORMER [11]	BLEU	25.3	**25.8**
		PERPLEXITY	**4.67**	4.92
WSJ	BILSTM+ATT.[10]	WER	8.9	7.0/**6.7**

of neural networks trained with label smoothing, and we describe several intriguing properties of these networks. Our contributions are as follows:

- We introduce a novel visualization method based on linear projections of the penultimate layer activations. This visualization provides intuition regarding how representations differ between penultimate layers of networks trained with and without label smoothing.
- We demonstrate that label smoothing implicitly calibrates learned models so that the confidences of their predictions are more aligned with the accuracies of their predictions.
- We show that label smoothing impairs distillation, *i.e.*, when teacher models are trained with label smoothing, student models perform worse. We further show that this adverse effect results from loss of information in the logits.

1.1 Preliminaries

Before describing our findings, we provide a mathematical description of label smoothing. Suppose we write the prediction of a neural network as a function of the activations in the penultimate layer as $p_k = \frac{e^{\boldsymbol{x}^T \boldsymbol{w}_k}}{\sum_{l=1}^{L} e^{\boldsymbol{x}^T \boldsymbol{w}_l}}$, where p_k is the likelihood the model assigns to the k-th class, $\boldsymbol{w}_k$ represents the weights and biases of the last layer, $\boldsymbol{x}$ is the vector containing the activations of the penultimate layer of a neural network concatenated with "1" to account for the bias. For a network trained with hard targets, we minimize the expected value of the cross-entropy between the true targets y_k and the network's outputs p_k as in $H(\boldsymbol{y}, \boldsymbol{p}) = \sum_{k=1}^{K} -y_k \log(p_k)$, where y_k is "1" for the correct class and "0" for the rest. For a network trained with a label smoothing of parameter α, we minimize instead the cross-entropy between the modified targets y_k^{LS} and the networks' outputs p_k, where $y_k^{LS} = y_k(1 - \alpha) + \alpha/K$.

2 Penultimate layer representations

Training a network with label smoothing encourages the differences between the logit of the correct class and the logits of the incorrect classes to be a constant dependent on α. By contrast, training a network with hard targets typically results in the correct logit being much larger than the any of the incorrect logits and also allows the incorrect logits to be very different from one another. Intuitively, the logit $\boldsymbol{x}^T \boldsymbol{w}_k$ of the k-th class can be thought of as a measure of the squared Euclidean distance between the activations of the penultimate layer $\boldsymbol{x}$ and a template $\boldsymbol{w}_k$, as $||\boldsymbol{x} - \boldsymbol{w}_k||^2 = \boldsymbol{x}^T \boldsymbol{x} - 2\boldsymbol{x}^T \boldsymbol{w}_k + \boldsymbol{w}_k^T \boldsymbol{w}_k$. Here, each class has a template $\boldsymbol{w}_k$, $\boldsymbol{x}^T \boldsymbol{x}$ is factored out when calculating the softmax outputs and $\boldsymbol{w}_k^T \boldsymbol{w}_k$ is usually constant across classes. Therefore, *label smoothing encourages the activations of the penultimate layer to be close to the template of the correct class and equally distant to the templates of the incorrect classes.* To observe this property of label smoothing, we propose a new visualization scheme based on the following steps: (1) Pick three classes, (2) Find an orthonormal basis of the plane crossing the templates of these three classes, (3) Project the penultimate layer activations of examples from these three classes onto this plane. This visualization shows in 2-D how the activations cluster around the templates and how label smoothing enforces a structure on the distance between the examples and the clusters from the other classes.

In Fig. 1, we show results of visualizing penultimate layer representations of image classifiers trained on the datasets CIFAR-10, CIFAR-100 and ImageNet with the architectures AlexNet [12], ResNet-56 [13] and Inception-v4 [14], respectively. Table 2 shows the effect of label smoothing on the accuracy of these models. We start by describing visualization results for CIFAR-10 (first row of Fig. 1) for the classes “airplane," “automobile" and “bird." The first two columns represent examples from the training and validation set for a network trained without label smoothing (w/o LS). We observe that

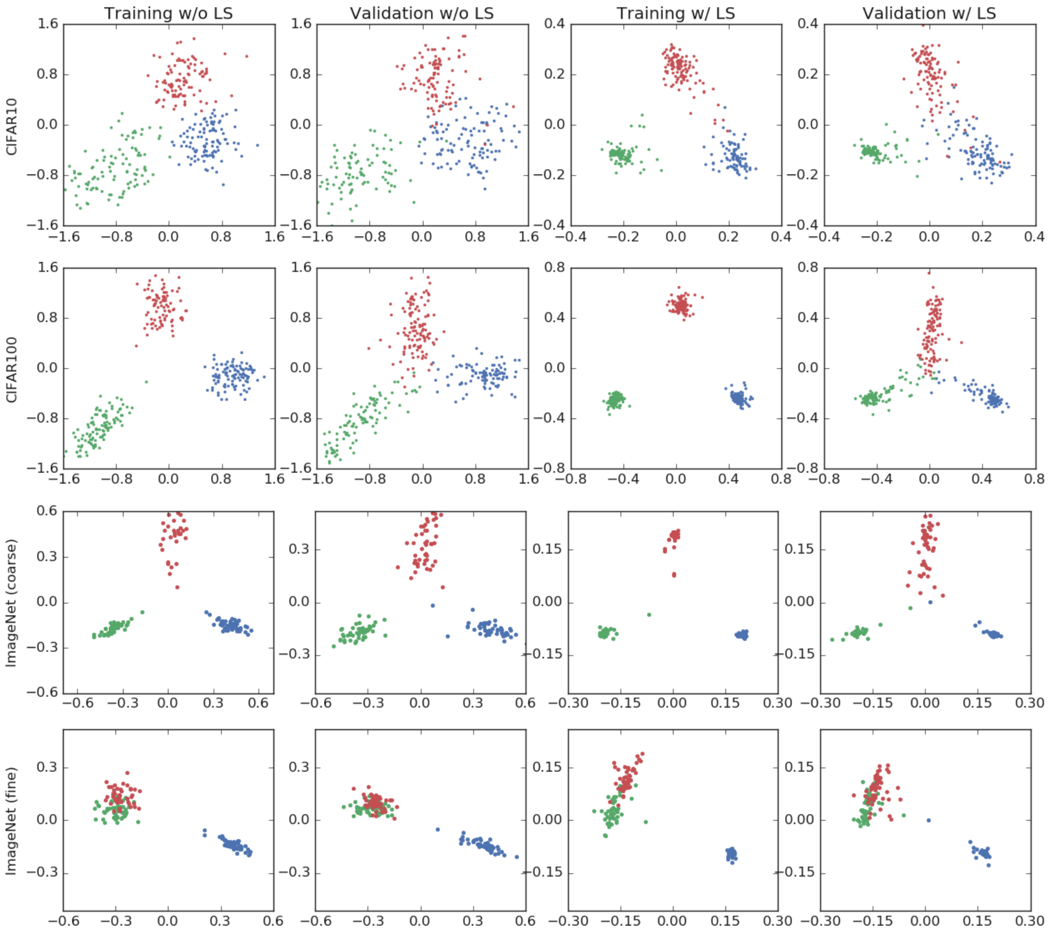

Figure 1: Visualization of penultimate layer's activations of: AlexNet/CIFAR-10 (first row), CIFAR-100/ResNet-56 (second row) and ImageNet/Inception-v4 with three semantically different classes (third row) and two semantically similar classes plus a third one (fourth row).

Table 2: Top-1 classification accuracies of networks trained with and without label smoothing used in visualizations.

Data set	Architecture	Accuracy ($\alpha = 0.0$)	Accuracy ($\alpha = 0.1$)
CIFAR-10	AlexNet	86.8 ± 0.2	86.7 ± 0.3
CIFAR-100	ResNet-56	72.1 ± 0.3	72.7 ± 0.3
ImageNet	Inception-v4	80.9	80.9

the projections are spread into defined but broad clusters. The last two columns show a network trained with a label smoothing factor of 0.1. We observe that now the clusters are much tighter, because label smoothing encourages that each example in training set to be equidistant from all the other class's templates. Therefore, when looking at the projections, the clusters organize in regular triangles when training with label smoothing, whereas the regular triangle structure is less discernible in the case of training with hard-targets (no label smoothing). Note that these networks have similar accuracies despite qualitatively different clustering of activations.

In the second row, we investigate the activation's geometry for a different pair of dataset/architecture (CIFAR-100/ResNet-56). Again, we observe the same behavior for classes "beaver," "dolphin," "otter." In contrast to the previous example, now the networks trained with label smoothing have better accuracy. Additionally, we observe the different scale of the projections between the network trained with and without label smoothing. With label smoothing, the difference between logits of two classes has to be limited in absolute value to get the desired soft target for the correct and incorrect classes. Without label smoothing, however, the projection can take much higher absolute values, which represent over-confident predictions.

Finally, we test our visualization scheme in an Inception-v4/ImageNet experiment and observe the effect of label smoothing for semantically similar classes, since ImageNet has many fine-grained classes (e.g. different breeds of dogs). The third row represents projections for semantically different classes (tench, meerkat and cleaver) with the behavior similar to previous experiments. The fourth row is more interesting, since we pick two semantically similar classes (toy poodle and miniature poodle) and observe the projection with the presence of a third semantically different one (tench, in blue). With hard targets, the semantically similar classes cluster close to each other with an isotropic spread. On the contrary, with label smoothing these similar classes lie in an arc. In both cases, the semantically similar classes are harder to separate even on the training set, but label smoothing enforces that each example be equidistant to all remaining class's templates, which gives rise to arc shape behavior with respect to other classes. We also observe that when training without label smoothing there is continuous degree of change between the "tench" cluster and the "poodles" cluster. We can potentially measure "how much a poodle is a particular tench". However, when training with label smoothing this information is virtually erased. This erasure of information is shown in the Section 4. Finally, the figure shows that the effect of label smoothing on representations is independent of architecture, dataset and accuracy.

3 Implicit model calibration

By artificially softening the targets, label smoothing prevents the network from becoming over-confident. But does it improve the calibration of the model by making the confidence of its predictions more accurately represent their accuracy? In this section, we seek to answer this question. Guo et al. [15] have shown that modern neural networks are poorly calibrated and over-confident despite having better performance than better calibrated networks from the past. To measure calibration, the authors computed the estimated expected calibration error (ECE). They demonstrated that a simple post-processing step, temperature scaling, can reduce ECE and calibrate the network. Temperature scaling consists in multiplying the logits by a scalar before applying the softmax operator. Here, we show that label smoothing also reduces ECE and can be used to calibrate a network without the need for temperature scaling.

Image classification. We start by investigating the calibration of image classification models. Fig. 2 (left) shows the 15-bin reliability diagram of a ResNet-56 trained on CIFAR-100. The dashed line represent perfect calibration where the output likelihood (confidence) predicts perfectly the accuracy. Without temperature scaling, the model trained with hard targets (blue line without markers) is clearly over-confident, since in expectation the accuracy is always below the confidence. To calibrate the model, one can tune the softmax temperature a posteriori (blue line with crosses) to a temperature of 1.9. We observe that the reliability diagram slope is now much closer to a slope of 1 and the model is better calibrated. We also show that, in terms of calibration, label smoothing has a similar effect. By training the same model with $\alpha = 0.05$ (green line), we obtain a model that is similarly calibrated compared to temperature scaling. In Table 3, we observe how varying label smoothing and temperature scaling affects ECE. Both methods can be used to reduce ECE to a similar and smaller value than an uncalibrated network trained with hard targets.

We also performed experiments on ImageNet (Fig. 2 right). Again, the network trained with hard targets (blue curve without markers) is over-confident and achieves a high ECE of 0.071. Using temperature scaling (T= 1.4), ECE is reduced to 0.022 (blue curve with crosses). Although we did not tune α extensively, we found that label smoothing of $\alpha = 0.1$ improves ECE to 0.035, resulting in better calibration compared to the unscaled network trained with hard targets.

These results are somewhat surprising in light of the penultimate layer visualizations of these networks shown in the previous section. Despite trying to collapse the training examples to tiny clusters, these networks generalize and are calibrated. Looking at the label smoothing representations for CIFAR-100 in Fig. 1 (second row, two last columns), we clearly observe this behavior. The red cluster is very tight in the training set, but in the validation set it spreads towards the center representing the full range of confidences for each prediction.

Machine translation. We also investigate the calibration of a model trained on the English-to-German translation task using the Transformer architecture. This setup is interesting for two reasons. First, Vaswani et al. [11] noted that label smoothing with $\alpha = 0.1$ improved the BLEU score of

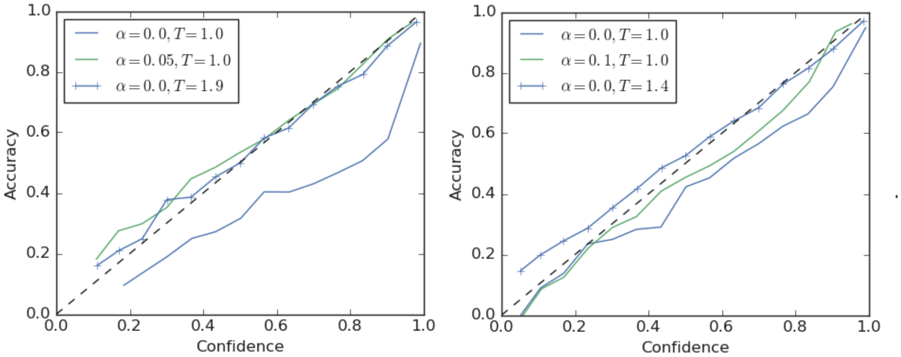

Figure 2: Reliability diagram of ResNet-56/CIFAR-100 (left) and Inception-v4/ImageNet (right).

Table 3: Expected calibration error (ECE) on different architectures/datasets.

Data set	Architecture	Baseline ECE (T=1.0, $\alpha = 0.0$)	Temp. scaling ECE / T ($\alpha = 0.0$)	Label smoothing ECE / α (T=1.0)
CIFAR-100	ResNet-56	0.150	0.021 / 1.9	0.024 / 0.05
ImageNet	Inception-v4	0.071	0.022 / 1.4	0.035 / 0.1
EN-DE	Transformer	0.056	0.018 / 1.13	0.019 / 0.1

their final model despite attaining worse perplexity compared to a model trained with hard targets ($\alpha = 0.0$). So we compare both setups in terms of calibration to verify that label smoothing also improves calibration in this task. Second, compared to image classification, where calibration does not directly affect the metric we care about (accuracy), in language translation, the network's soft outputs are inputs to a second algorithm (beam-search) which is affected by calibration. Since beam-search approximates a maximum likelihood sequence detection algorithm (Viterbi algorithm), we would intuitively expect better performance for a better calibrated model, since the model's confidence predicts better the accuracy of the next token.

We start by looking at the reliability diagram (Fig. 3) for a Transformer network trained with hard targets (with and without temperature scaling) and a network trained with label smoothing ($\alpha = 0.1$). We plot calibration of the next-token predictions assuming a correct prefix on the validation set. The results are in agreement with the previous experiments on CIFAR-100 and ImageNet, and indeed the Transformer network [11] with label smoothing is better calibrated than the hard targets alternative.

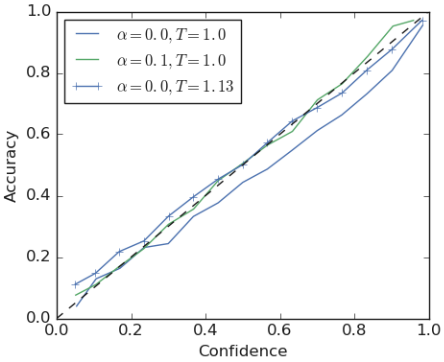

Figure 3: Reliability diagram of Transformer trained on EN-DE dataset.

Despite being better calibrated and achieving better BLEU scores, label smoothing results in worse negative log-likelihoods (NLL) than hard targets. Moreover, temperature scaling with hard targets is insufficient to recover the BLEU score improvement obtained with label smoothing. In Fig. 4, we artificially vary calibration of both networks, using temperature scaling, and analyze the effect upon BLEU and NLL. The left panel shows results for a network trained with hard targets. By increasing the temperature we can both reduce ECE (red, right y-axis) and slightly improve BLEU score (blue left y-axis), but the BLEU score improvement is not enough to match the BLEU score of the network trained with label smoothing (center panel). The network trained with label smoothing is "automatically calibrated" and changing the temperature degrades both calibration and BLEU score. Finally, in the right panel, we plot the NLL for both networks, where markers represent the network with label smoothing. The model trained with hard targets achieves better NLL at all temperature scaling settings. Thus, label smoothing improves translation quality measured by BLEU score despite worse NLL, and the difference of BLEU score performance is only partly explained by calibration. Note that the minimum of ECE for this experiment predicts top BLEU score slightly better than NLL.

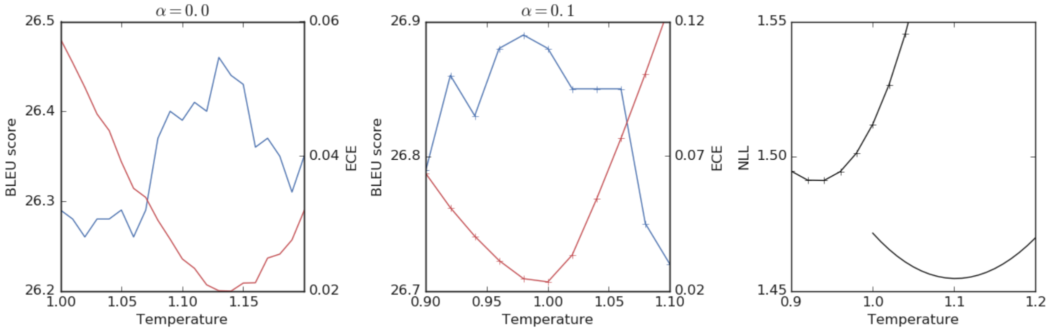

Figure 4: Effect of calibration of Transformer upon BLEU score (blue lines) and NLL (red lines). Curves without markers reflect networks trained without label smoothing while curves with markers represent networks with label smoothing.

4 Knowledge distillation

In this section, we study how the use of label smoothing to train a teacher network affects the ability to distill the teacher's knowledge into a student network. We show that, even when label smoothing improves the accuracy of the teacher network, teachers trained with label smoothing produce inferior student networks compared to teachers trained with hard targets. We first noticed this effect when trying to replicate a result in [16]. A non-convolutional teacher is trained on randomly translated MNIST digits with hard targets and dropout and gets 0.67% test error. Using distillation, this teacher can be used to train a narrower, unregularized student on untranslated digits to get 0.74% test error. If we use label smoothing instead of dropout, the teacher trains much faster and does slightly better (0.59%) but distillation produces a much worse student with 0.91% test error. Something goes seriously wrong with distillation when the teacher is trained with label smoothing.

In knowledge distillation, we replace the cross-entropy term $H(\boldsymbol{y}, \boldsymbol{p})$ by the weighted sum $(1-\beta)H(\boldsymbol{y}, \boldsymbol{p}) + \beta H(\boldsymbol{p}^t(T), \boldsymbol{p}(T))$, where $p_k(T)$ and $p_k^t(T)$ are the outputs of the student and teacher after temperature scaling with temperature T, respectively. β controls the balance between two tasks: fitting the hard-targets and approximating the softened teacher. The temperature can be viewed as a way to exaggerate the differences between the probabilities of incorrect answers.

Both label smoothing and knowledge distillation involve fitting a model using soft targets. Knowledge distillation is only useful if it provides an additional gain to the student compared to training the student with label smoothing, which is simpler to implement since it does not require training a teacher network. We quantify this gain experimentally. To demonstrate these ideas, we perform an experiment on the CIFAR-10 dataset. We train a ResNet-56 teacher and we distill to an AlexNet student. We are interested in four results:

1. the teacher's accuracy as a function of the label smoothing factor,
2. the student's baseline accuracy as a function of the label smoothing factor without distillation,
3. the student's accuracy after distillation with temperature scaling to control the smoothness of the teacher's provided targets (teacher trained with hard targets)
4. the student's accuracy after distillation with fixed temperature ($T = 1.0$ and teacher trained with label smoothing to control the smoothness of the teacher's provided targets)

To compare all solutions using a single smoothness index, we define the equivalent label smoothing factor γ which for scenarios 1 and 2 are equal to α. For scenarios 3 and 4, the smoothness index is $\gamma = \mathbb{E}\big[\sum_{k=1}^{K}(1-y_k)p_k^t(T)K/(K-1)\big]$, which calculates the mass allocated by the teacher to incorrect examples over the training set. Since the training accuracy is nearly perfect, for all distillation experiments, we consider only the case where $\beta = 1$, *i.e.*, when the targets are the teacher output and the true labels are ignored.

Fig. 5 shows the results of this distillation experiment. We first compare the performance of the teacher network (blue solid curve, top) and student network (light blue solid, bottom) trained without distillation. For this particular setup, increasing α improves the teacher's accuracy up to values of $\alpha = 0.6$, while label smoothing slightly degrades the baseline performance of the student networks.

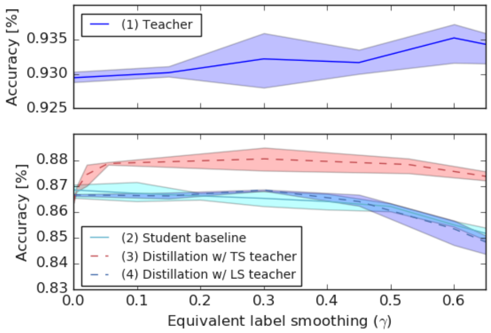

Figure 5: Performance of distillation from ResNet-56 to AlexNet on CIFAR-10.

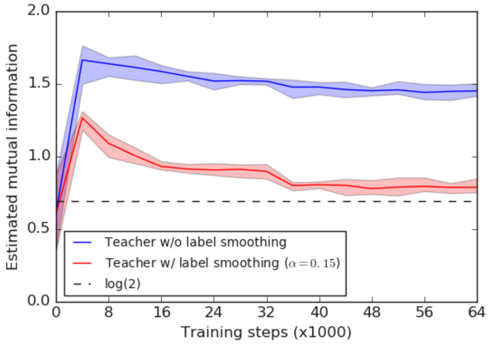

Figure 6: Estimated mutual information evolution during teacher training.

Next, we distill the teacher trained with hard targets to students using different temperatures, and calculate the corresponding γ for each temperature (red dashed curve). We observe that all distilled models outperform the baseline student with label smoothing. Finally, we distill information from teachers trained with label smoothing $\alpha > 0$, which have better accuracy (blue dashed curve). The figure shows that using these better performing teachers is no better, and sometimes worse, than training the student directly with label smoothing, as the relative information between logits is "erased" when the teacher is trained with label smoothing.

To observe how label smoothing “erases" the information contained in the different similarities that an individual example has to different classes we revisit the visualizations in Fig. 1. Note that we are interested in the visualization of the examples from the training set, since those are the ones used for distillation. For a teacher trained with hard targets ($\alpha = 0.0$), we observe that examples are distributed in broad clusters, which means that different examples from the same class can have very different similarities to other classes. For a teacher trained with label smoothing, we observe the opposite behavior. Label smoothing encourages examples to lie in tight equally separated clusters, so every example of one class has very similar proximities to examples of the other classes. Therefore, *a teacher with better accuracy is not necessarily the one that distills better*.

One way to directly quantify information erasure is to calculate the mutual information between the input and the logits. Computing mutual information in high dimensions for unknown distributions is challenging, but here we simplify the problem in several ways. We measure the mutual information between X and Y, where X is a discrete variable representing the index of the training example and Y is continuous representing the difference between two logits (out of K classes). The source of randomness comes from data augmentation and we approximate the distribution of Y as a Gaussian and we estimate the mean and variance from the examples using Monte Carlo samples. The difference of the logits y can be written as $y = f(d(\boldsymbol{z}_x))$, where $\boldsymbol{z}_x$ is the flattened input image indexed by x, $d(\cdot)$ is a random data augmentation function (random shifts for example), and $f(\cdot)$ is a trained neural network with an image as an input and the difference between two logits as output (resulting in a single dimension real-valued output). The mutual information and its respective approximation are $I(X;Y) = E_{X,Y}[\log(p(y|x)) - \log(\sum_x p(y|x)p(x))]$ and

$$\hat{I}(X;Y) = \frac{1}{N}\sum_{x=1}^{N}\big[-(f(d(\boldsymbol{z}_x)) - \mu_x)^2/(2\sigma^2) - \log(\frac{1}{N}\sum_{x=1}^{N} e^{-(f(d(\boldsymbol{z}_x))-\mu_x)^2/(2\sigma^2)})\big],$$

where $\mu_x = \sum_{l=1}^{L} f(d(\boldsymbol{z}_x))/L$, $\sigma^2 = \sum_{x=1}^{N}(f(d(\boldsymbol{z}_x)) - \mu_x)^2/N$, L is the number of Monte Carlo samples used to calculate the empirical mean and N is the number of training examples used for mutual information estimation. Here the mutual information is between 0 and $\log(N)$.

Fig. 6 shows the estimated mutual information between a subset (N = 600 from two classes) of the training examples and the difference of the logits corresponding to these two classes. After initialization, the mutual information is very small, but as the network is trained, first it rapidly increases then it slowly decreases specially for the network trained with label smoothing. This result confirms the intuitions from the last sections. As the representations collapse to small clusters of points, much of the information that could have helped distinguish examples is lost. This results in lower estimated mutual information and poor distillation for teachers trained with label smoothing.

For later stages of training, mutual information stays slightly above $\log(2)$, which corresponds to the extreme case where all training examples collapse to two separate clusters. In this case, all the information of the input is discarded except a single bit representing which class the example belongs to, resulting in no extra information in the teacher's logits compared to the information in the labels.

5 Related work

Pereyra et al. [17] showed that label smoothing provides consistent gains across many tasks. That work also proposed a new regularizer based on penalizing low entropy predictions, which the authors term "confidence penalty." They show that label smoothing is equivalent to confidence penalty if the order of the KL divergence between uniform distributions and model's outputs is reversed. They also propose to use distributions other than uniform, resulting in unigram label smoothing (see Table 1) which is advantageous when the output labels' distribution is not balanced. Label smoothing also relates to DisturbLabel [18], which can be seen as label dropout, whereas label smoothing is the marginalized version of label dropout.

Calibration of modern neural networks [15] has been widely investigated for image classification, but calibration of sequence models has been investigated only recently. Ott et al. [19] investigate the sequence level calibration of machine translation models and conclude they are remarkably well calibrated. Kumar and Sarawagi [20] investigate calibration of next-token prediction in language translation. They find that calibration of state-of-the-art models can be improved by a parametric model, resulting in a small increase in BLEU score. However, neither work invesigates the relation between label smoothing during training and calibration. For speech recognition, Chorowski and Jaitly [10] investigate the effect of softmax temperature and label smoothing on decoding accuracy. The authors conclude that both temperature scaling and label smoothing improve word error rates after beam-search (label smoothing performs best), but the relation between calibration and label smoothing/temperature scaling is not described.

Although we are unaware of any previous work that shows the adverse effect of label smoothing upon distillation, Kornblith et al. [21] previously demonstrated that label smoothing impairs the accuracy of transfer learning, which similarly depends on the presence of non-class-relevant information in the final layers of the network. Additionally, Chelombiev et al. [22] propose an improved mutual information estimator based on binning and show correlation between compression of softmax layer representations and generalization, which may explain why networks trained with label smoothing generalize so well. This relates to the information bottleneck theory [23, 24, 25] that explains generalization in terms of compression.

6 Conclusion and future work

Many state-of-the-art models are trained with label smoothing, but the inductive bias provided by this technique is not well understood. In this work, we have summarized and explained several behaviors observed while training deep neural networks with label smoothing. We focused on how label smoothing encourages representations in the penultimate layer to group in tight equally distant clusters. This emergent property can be visualized in low dimension thanks to a new visualization scheme that we proposed. Despite having a positive effect on generalization and calibration, label smoothing can hurt distillation. We explain this effect in terms of erasure of information. With label smoothing, the model is encouraged to treat each incorrect class as equally probable. With hard targets, less structure is enforced in later representations, enabling more logit variation across predicted class and/or across examples. This can be quantified by estimating mutual information between input example and output logit and, as we have shown, label smoothing reduces mutual information. This finding suggests a new research direction, focusing on the relationship between label smoothing and the information bottleneck principle, with implications for compression, generalization and information transfer. Finally, we performed extensive experiments on how label smoothing can implicitly calibrate model's predictions. This has big impact on model interpretability, but also, as we have shown, can be critical for downstream tasks that depend on calibrated likelihoods such as beam-search.

7 Acknowledgements

We would like to thank Mohammad Norouzi, William Chan, Kevin Swersky, Danijar Hafner and Rishabh Agrawal for the discussions and suggestions.

References

[1] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Nature*, 323:19, 1986.

[2] Eric B Baum and Frank Wilczek. Supervised learning of probability distributions by neural networks. In *Neural information processing systems*, pages 52–61, 1988.

[3] Sara Solla, Esther Levin, and Michael Fleisher. Accelerated learning in layered neural networks. *Complex systems*, 2:625–640, 1988.

[4] Scott E Fahlman. An empirical study of learning speed in back-propagation networks. Technical report, Carnegie Mellon University, Computer Science Department, 1988.

[5] John A Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley, 2018.

[6] 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*, pages 2818–2826, 2016.

[7] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[8] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

[9] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, and Zhifeng Chen. GPipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*, 2018.

[10] Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. *Proc. Interspeech 2017*, pages 523–527, 2017.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] 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*, pages 770–778, 2016.

[14] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[15] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

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

[17] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[18] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762, 2016.

[19] Myle Ott, Michael Auli, David Grangier, et al. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*, pages 3953–3962, 2018.

[20] Aviral Kumar and Sunita Sarawagi. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*, 2019.

[21] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

[22] Ivan Chelombiev, Conor Houghton, and Cian O'Donnell. Adaptive estimators show information compression in deep neural networks. *arXiv preprint arXiv:1902.09037*, 2019.

[23] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[24] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

[25] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Appendix

A Experimental details

A.1 AlexNet

The AlexNet architecture was used in the visualization section and in the distillation section. In both cases, it was trained on the CIFAR-10 dataset. We split the training set into training and validation (40k/10k) and results are shown in the validation set. Our implementation is based on the public available code in [2] and [3] without exponential weight averaging. The mini-batch size is 128 and we train for 390k iterations with stochastic gradient descent, starting with learning rate 0.1 and droping by a factor of 10 every 130k steps. We multiply the cross-entropy term by 3 and use weight decay of 0.04 in the last two dense layers. Five different seeds were used for each point shown in the figure and we show min/average/max. For visualization of representations of CIFAR-10 a total of 300 examples were used (100 per class). For the distillation results, β is set to 1 so distillation is done fully with teacher targets. Note that we use the same training procedure for: training the student without distillation (variable label smoothing), training the student to mimic the teacher trained with hard targets and temperature scaling, and training the student to mimic the teacher trained with variable label smoothing. In the end, the only difference is the targets used. Additionally, when $\beta = 1$, the temperature of the teacher is important to soften the targets, but the temperature of the student ends up to be just a scalar multiplying the logits and can be learned, therefore we set the student soft-max temperature to 1 for all experiments. For distillation with teacher trained with hard targets, the temperatures tested were 1.0, 2.0, 3.0, 4.0, 8.0, 12.0 and 16.0. The label smoothing values used were 0.0 to 0.75 with steps of 0.15.

A.2 ResNet

The ResNet architecture was used in the visualization, calibration and distillation sections. In all cases, it was trained on the CIFAR-10 and CIFAR-100 datasets[4]. We split the training set into training and validation (40k/10k) and results are shown in the validation set. Our implementation uses the public available model in [5]. The mini-batch size is 128 and we train for 64k iterations with stochastic gradient descent with Nesterov momentum of 0.9, starting with learning rate 0.1 and dropping by a factor of 10 at 32k and 48k steps. We multiply the cross-entropy term by 3 and use weight decay of 0.0001. Compared to the implementation in tensor2tensor library, we include gradient clipping of 1.0 and to create a deep network of 56 layers, we set the number of blocks per layer to 9 for the three blocks. We also set the kernel size to 3 instead of 7 to resemble the original ResNet architecture for CIFAR-10 dataset. For visualization of representations of CIFAR-100 a total of 300 examples were used (100 per class). For the calibration experiment, all 10k examples on the validation set were used to calculate the ECE and reliability diagram. The label smoothing values used when training the teacher in the distillation section were 0.0 to 0.75 with steps of 0.15. For the mutual information results we pick the classes "airplane" and "dog", and calculate the mutual information using 300 examples from the training set of each class. To calculate the average logit per example, we use 5 Monte Carlo samples and estimated mutual information is calculated every 4k training steps. For data augmentation we used random crops and left-right flips as in [6](cifar_image_augmentation).

A.3 Transformer

The Transformer architecture was used in the calibration section. We reuse the public available implementation by the original authors described in [7]. The dataset can be found in [8](TranslateEndeWmt32k for around 32k tokens). We used the hyper-parameters provided by the authors of original paper for single GPU training (transformer_base for label smoothing of 0.1, and transformer_ls0 for label smoothing of 0.0). Compared to the original paper, we train for 300k steps instead of 100k and we do not use weight averaging of checkpoints. For the calibration experiment, around 10k tokens of the

[2] https://www.tensorflow.org/tutorials/images/deep_cnn

[3] https://github.com/tensorflow/models/blob/master/tutorials/image/cifar10/cifar10.py

[4] https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/data_generators/cifar.py

[5] https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/models/resnet.py

[6] https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/data_generators/image_utils.py

[7] https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/models/transformer.py

[8] https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/data_generators/translate_ende.py

validation set were used to calculate the ECE and reliability diagram. For calculation of BLEU score (uncased), we use the full validation set [9].

A.4 Inception-v4

The Inception-v4 architecture was used in the visualization and calibration sections. We reuse a public implementation of the model which be can found at [10]. We modified this code to use a scale parameter for the batch normalization layer, as we found it improved performance. We used a batch size of 4096 and trained the model using stochastic gradient descent with Nesterov momentum of 0.9 and weight decay of 8×10^{-5}. We took an exponential average of the weights with decay factor 0.9999, and selected the checkpoint that achieved the maximum accuracy during a separate training run where approximately 50,000 images from the ImageNet training set were used for validation.

A.5 Fully-connected

For the distillation results on MNIST we use fully connected networks with 2 layers. 1200 neurons per layer is used on the teacher and 800 layers in the student. For training the teacher, we use $\alpha = 0.1$, random image shifts of plus or minus 2 pixels in both x and y axis (i.e. 25 equiprobable centerings for each case). The initialization distribution is Gaussian with variance 0.03. Learning rate is set 1, except for last layer which is set to 0.1. We used gradient smoothing corresponding to 0.9 times previous gradient plus 0.1 the current one. Finally the learning rate drops linearly to 0 after 100 epochs and no weight decay is used. For training the student during distillation, no data augmentation is used. We used $\beta = 0.6$, so the original cross-entropy with hard-targets is multiplied by 0.4 and we match the teacher logits with half the squared loss multiplied by 0.6. Note that optimizing for the squared loss is equivalent to picking a high temperature.

B Penultimate layer representation for translation

Below, we show visualizations for the English-to-German translation task. The results are similar to the previous image classification visualization results. Next-token prediction is equivalent to classification. In image classification we maximize the likelihood $p(y|x)$ of the correct class given an image, whereas in translation, we maximize the likelihood $p(y_t|x, y_{0:t-1})$ of next token given source sentence and preceding target sequence. However, there are differences between the tasks that affect visualization and distillation.

- The image classification datasets we examine have balanced class distributions, whereas token distributions in translation are highly imbalanced. (This could potentially be addressed with unigram label smoothing.)
- For image classification, we can get near-perfect training set accuracy and still generalize (interpolation regime); in this case, label smoothing erases information, whereas hard-targets preserves it. In the translation task, next-token accuracy on the training set is around 80%. Therefore, visualizations show errors in both the training and validation sets (tending to tight clusters as expected). For distillation, this means a student may learn from the teacher's errors. Since teachers trained with and without label smoothing will both have errors that the student can learn from, it is unclear which will perform best.
- For image classification, the penultimate layer dimension is usually higher than the number of classes, so templates can lie on a regular simplex (i.e. equidistant to each other). In translation, we have a ~30k token alphabet in 512 dimensions, so a simplex is not possible.

In this work, for visualization and distillation, we concentrate on the image classification case to simplify intuition and experimental design. The visualization results below demonstrate that some features of the image classification visualization also hold for translation. We leave analysis of distillation for translation for future work.

[9] `https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/bin/t2t_bleu.py`

[10] `https://github.com/tensorflow/models/blob/master/research/slim/nets/inception_v4.py`

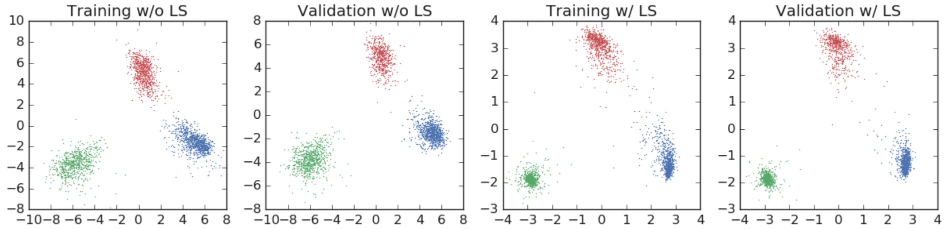

Figure 7: Visualizations of penultimate representations of Transformer trained to perform English-to-German translation.