PAUL AGGARWAL

Big Transfer

QUESTION 4:

Mixup alleviates undesirable behaviors such as memorization and sensitivity to adversarial examples in large deep neural networks. Mixup trains a neural network on convex combinations of pairs of examples and their labels. By doing so, mixup regularizes the neural network to favor simple linear behavior in-between training examples. It's a data augmentation method that consists of only two parts: random convex combination of raw inputs, and correspondingly, convex combination of one-hot label encodings. This data augmentation technique generates new examples as random convex combinations of data points and labels from the training set. This technique has been proven to substantially improve generalization on a broad range of tasks ranging from computer vision to natural language processing and semi-supervised learning. It does so by alleviating overfitting, improved calibration, robustness to input adversarial noise, and robustness to label corruption in training deep neural networks.

Mixup can exploit ingredients of label smoothing (leads to better calibration), dropout (improves generalization and robustness to label corruption) and Lipschitz regularization (helps stabilize the training of generative adversarial networks and leads to increased robustness to adversarial perturbations). It uniquely combines them to improve calibration and smooth the Jacobian of the model. Mixup strong regularization properties, which may explain its good empirical behavior in a variety of tasks.

Mixup transformations shrinks both the inputs and the outputs towards their mean creating a form of regularization by label smoothing where it translates into an increase in the entropy of the predictions.

*References for equations, ERM, perturbation, etc: arXiv:202006.06049.pdf and arXiv:201710.09412.pdf

Solving (1) with the empirical risk (2) is often called empirical risk minimization (ERM), and is typically performed in practice by first-order numerical optimization such as stochastic gradient descent [7].

Perturbing inputs with additive or multiplicative noise (e.g., dropout), and independently perturbing outputs (resulting, e.g., in label smoothing) Mixup perturbation (7) is unique in the sense that it is applied to both inputs and outputs simultaneously, and that the input and output perturbations are not independent from each other

Analyzing input/output perturbations with Taylor approximation used a quadratic approximation of the loss function dropout combined with independent label smoothing) Mixup is unique in the correlation it creates between input and output perturbation.

We can see how training without regularization (ERM) leads to overfitted estimators, while both Mixup and approximate Mixup have lower training accuracy but higher test accuracy than ERM, suggesting that approximate Mixup correctly mimics Mixup's ability to control overfitting.

On input modification, mixup implicitly shrinks inputs towards their mean. Bayes optimal classifier under the Mixup distribution matches the one under the empirical distribution of the modified data (up to regularization effects), and not of the original data. This is coherent with the toy problem experiment, using (15) at prediction time for the model trained with Mixup improves test accuracy, and gets closer to the performance of approximate Mixup.

In the multi-class setting, since the softmax is invariant to a constant in the logits, (15) becomes equivalent to a scaling of the logits, commonly referred to as temperature scaling. Mixup automatically sets this value, according to the distribution of θ.

For label smoothing, which is the transformation that modifies the original labels y onto y~ and acts as some form of label smoothing, a technique known to often improve accuracy and calibration.

The transformed labels y~ are indeed pulled towards the average label of y. So, it is easy to see that for $\varepsilon = (1 - \theta)$ and $u(i) = y$ the two formulations coincide. This implies that Mixup implicitly performs label smoothing, and can benefit from this technique in terms of accuracy or calibration. However, in the case of the cross entropy and linear models, label smoothing translates into an increase in the average entropy of the predictions, or, in other words, that predictions become less certain, as observed in practice.

Standard ERM produces very confident predictions, how label smoothing helps decreasing ERM confidence at test time, and how Mixup naturally produces even less confident (and often better calibrated) predictions. We also see that that approximate Mixup, like Mixup, produces less confident prediction, and that the similarity of the prediction confidence is very good when Mixup is “correctly” used at prediction time using (15). This supports our analysis that Mixup’s ability to produce well-calibrated scores stems from the label smoothing effect captured by our approximate Mixup formulation.

Mixup also regularizes the Jacobian of f but with a different and more informative implicit bias, namely, to mimic a good linear model in the input space this implicit bias results from the correlation between input and output noise. Similar to dropout, this implies that this regularization vanishes when the prediction p(f(x~)) is confidently near 0 or 1. In the Mixup case, the label smoothing effect tends to prevent over-confident predictions on the training point ensuring that the Jacobian regularization remains active even for “easy” points. This interaction between label smoothing (due to output Mixup) and Jacobian regularization (due to input Mixup) may explain why Mixup on inputs only performs poorly compared to Mixup on both inputs and outputs.

Mixup improves the generalization error of state-of-the-art models on ImageNet, CIFAR, speech, and tabular datasets. Furthermore, mixup helps to combat memorization of corrupt labels, sensitivity to adversarial examples, and instability in adversarial training. It is consistent that with increasingly large α, the training error on real data increases, while the generalization gap decreases.

Even though mixup implicitly controls model complexity, we still do not yet have a good theory for understanding the ‘sweet spot’ of this bias-variance trade-off.

CIFAR-10 classification we can get very low training error on real data even when $\alpha \rightarrow \infty$ (i.e., training only on averages of pairs of real examples), whereas in ImageNet classification, the training error on real data increases significantly with $\alpha \rightarrow \infty$

Based on our ImageNet and Google commands experiments with different model architectures, we conjecture that increasing the model capacity would make training error less sensitive to large α, hence giving mixup a more significant advantage.

The implementation of mixup training is straightforward, and introduces a minimal computation overhead.

Using a single data loader to obtain one minibatch, and then mixup is applied to the same minibatch after random shuffling. It was found that this strategy works equally well, while reducing I/O requirements.

When two neural network models trained on the CIFAR-10 dataset using ERM and mixup, the model trained with mixup is more stable in terms of model predictions and gradient norms in-between training samples.

Mixup on the ImageNet-2012: mixup, we find that α ∈ [0.1, 0.4] leads to improved performance over ERM, whereas for large α, mixup leads to underfitting. We also find that models with higher capacities and/or longer training runs are the ones to benefit the most from mixup. For example, when trained for 90 epochs, the mixup variants of ResNet-101 and ResNeXt-101 obtain a greater improvement (0.5% to 0.6%) over their ERM analogues than the gain of smaller models such as ResNet-50 (0.2%). When trained for 200 epochs, the top-1 error of the mixup variant of ResNet-50 is further reduced by 1.2% compared to the 90 epoch run, whereas its ERM analogue stays the same.

As the training progresses with a smaller learning rate (e.g. less than 0.01), the ERM model starts to overfit the corrupted labels. When using a large probability (e.g. 0.7 or 0.8), dropout can effectively reduce overfitting. Mixup with a large α (e.g. 8 or 32) outperforms dropout on both the best and last epoch test errors, and achieves lower training error on real labels while remaining resistant to noisy labels. Interestingly, mixup + dropout performs the best of all, showing that the two methods are compatible.

Finally, as it was explained before, mixup is a data augmentation method that consists of only two parts: random convex combination of raw inputs, and correspondingly, convex combination of one-hot label encodings. There is specific design parameter, weight decay to take into consideration. For each of the data augmentation methods, two weight decay settings (10^−4 which works well for mixup, and 5 × 10^−4 which works well for ERM) were tested by the respective authors. The following observations were made: First, mixup is the best data augmentation method they test, and is significantly better than the second best method (mix input + label smoothing). Second, the effect of regularization can be seen by comparing the test error with a small weight decay (10^−4) with a large one (5 × 10^−4). For example, for ERM a large weight decay works better, whereas for mixup a small weight decay is preferred, confirming its regularization effects.

In conclusion, the theoretical and experimental analysis that explains the multiple regularization effects of Mixup proves that training with Mixup is equivalent to learn on modified data with the injection of structured noise. Through a Taylor approximation, Mixup amounts to empirical risk minimization on modified points plus multiple regularization terms. Mixup induces varied and complex effects, e.g., calibration, Jacobian regularization, label noise and normalization, while being a simple and cheap data augmentation technique while applying Mixup to Deep Learning model capacity training and testing such as Big Transfer.