Regularizing Neural Networks by Label Smoothing

Elyas Bayati - Mehrdad Hessar
Statistical Learning Winter 2017 | University of Washington

Abstract

In this project, we want to answer this question. Can we improve overfitting problem by using regularization approaches on the output? We consider three recent approaches for this problem; **Label Smoothing**, **Knowledge Distillation** and **Penalizing Output Confidence**. In this work, we implement these three methods and evaluate their effect on the test accuracy of two well-known datasets; MNIST [1] dataset and German Traffic Sign Benchmarks (GTSRB) [2] dataset. Our results show that all methods can improve the accuracy on the test data, but using Label Smoothing which is a less complicated method has better accuracy.

Introduction

Large neural networks with millions of parameters achieve strong performance on image classification, machine translation, language modeling, and speech recognition. However, despite using large datasets, neural networks are still prone to overfitting, and also there is no optimal background theory for it. Numerous techniques have been proposed to prevent overfitting, including early stopping, L1/L2 regularization, dropout, and batch normalization. These techniques, along with most other forms of regularization, act on the hidden activations or weights of a neural network. Alternatively, regularizing the output distribution of large, deep neural networks has largely been unexplored.

Recently, it is shown that output regularization methods can improve state of-the-art models across benchmarks without modifying existing hyperparameters. Here we are exploring ways to scale up networks in ways that aim to utilize additional computation as efficiently as possible by suitably using different regularization. We will cover each approach independently by theory and after that, we will the results of our implementation.

Output Regularization Approaches

1. Label Smoothing

Here we propose a mechanism to regularize the classifier layer by estimating the marginalized effect of label-dropout during training [3]. For each training example *x*, our model computes the probability of each label *k* (which is between 1 to K):

$$\frac{\exp(z_k)}{\sum_{i=1}^{K}\exp(z_i)}$$

Here, z_i are the logits or unnormalized log probabilities. Consider the ground-truth distribution over labels q(k|x) for this training example, normalized so that their sum in all distribution became 1. For brevity, let us omit the dependence of *p* and *q* on example *x*. We define the loss for the example cross entropy:

$$\ell = -\sum_{k=1}^{K} \log(p(k))q(k).$$

Minimizing this equation, is equivalent to maximizing the expected log-likelihood of a label, where the label is selected according to its ground-truth distribution *q(k)*. Cross-entropy loss is differentiable with respect to the logits z_k and thus can be used for gradient training of deep models.

Consider the case of a single ground-truth label "y", The log-likelihood is maximized for q(k) = $\boldsymbol{\delta(k,y)}$, where $\boldsymbol{\delta(k,y)}$ is Dirac delta for k = y and 0 otherwise. This, however, can cause two problems. First, it may result in overfitting, Second, it encourages the differences between the largest logit and all others to become large, and reduces the ability of the model to adapt. Intuitively, this happens because the model has too much confidence about its predictions.

We propose a mechanism for encouraging the model to be less confident. Consider a distribution over labels independent of the training example x, and a smoothing parameter $\boldsymbol{p}$ for a training example with ground-truth label y, we replace the label distribution $\boldsymbol{q(k|x) = \delta(k,y)}$ with:

$$\boldsymbol{q'(k) = (1-p)\delta_{k,y} + \frac{p}{K}}$$

We refer to this change in ground-truth label distribution as label-smoothing regularization, or LSR. Note that LSR achieves the desired goal of preventing the largest logit from becoming much larger than all others. Indeed, if this were to happen, then a single q(k) would approach 1 while all others would approach 0. This would result in a large cross-entropy with q(k) because, unlike q(k) = δ(k,y), all q'(k) have a positive lower bound.

2. Penalizing the Output Confidence

Instead of smoothing the labels with a uniform distribution, as in label smoothing, we can smooth the labels with a teacher model or the model's own distribution [4]. Confident predictions correspond to output distributions that have low entropy. A network is overconfident when it places all probability on a single class in the training set, which is often a symptom of overfitting. The confidence penalty constitutes a regularization term that prevents these peaked distributions, leading to better generalization.

A neural network produces a conditional distribution $p_\theta(y|x)$ over classes y given an input x through a SoftMax function. The entropy of this conditional distribution is given by:

$$\boldsymbol{H(p_\theta(y|x)) = -\sum_i p_\theta(y_i|x) log(p_\theta(y_i|x))}$$

To penalize confident output distributions, we add the negative entropy to the negative log-likelihood during training :

$$\boldsymbol{\mathcal{L}(\theta) = -\sum log p_\theta(y|x) - \beta H(p_\theta(y|x))}$$

In reinforcement learning, penalizing low entropy distributions prevents a policy network from converging early and encourages exploration. However, in supervised learning, we typically

want quick convergence, while preventing overfitting near the end of training, suggesting a confidence penalty that is weak at the beginning of training and strong near convergence. A simple way to achieve this is to anneal the confidence penalty.

3. Knowledge Distillation

A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and then to average their predictions. Unfortunately, making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural nets [5]. It is shown that it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy.

For tasks like MNIST in which the cumbersome model almost always produces the correct answer with very high confidence, much of the information about the learned function resides in the ratios of very small probabilities in the soft targets. For example, one version of a 2 may be given a probability of 10^{-6} of being a "3" and 10^{-9} of being a "7" whereas for another version it may be the other way around. This is valuable information that defines a rich similarity structure over the data (i. e. it says which 2's look like 3's and which look like 7's) but it has very little influence on the cross-entropy cost function during the transfer stage because the probabilities are so close to zero.

More general solution, called "distillation", is to raise the temperature of the final softmax until the cumbersome model produces a suitably soft set of targets. We then use the same high temperature when training the small model to match these soft targets. It is shown that matching the logits of the cumbersome model is actually a special case of distillation.

Neural networks typically produce class probabilities by using a "Softmax" output layer that converts the logit, z_i, computed for each class into a probability, q_i, by comparing z_j with the other logits.

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

where T is a temperature that is normally set to 1. Using a higher value for T produces a softer probability distribution over classes.

In the simplest form of distillation, knowledge is transferred to the distilled model by training it on a transfer set and using a soft target distribution for each case in the transfer set that is produced by using the cumbersome model with a high temperature in its softmax. The same high temperature is used when training the distilled model, but after it has been trained it uses a temperature of 1.

When the correct labels are known for all or some of the transfer set, this method can be significantly improved by also training the distilled model to produce the correct labels. One way to do this is to use the correct labels to modify the soft targets, but we found that a better way is to simply use a weighted average of two different objective functions. The first objective function is the cross entropy with the soft targets and this cross entropy is computed using the same

high temperature in the softmax of the distilled model as was used for generating the soft targets from the cumbersome model. The second objective function is the cross entropy with the correct labels. This is computed using exactly the same logits in softmax of the distilled model but at a temperature of 1. We found that the best results were generally obtained by using a considerably lower weight on the second objective function. In Fig.1, we use a simple model to show how distillation model work.

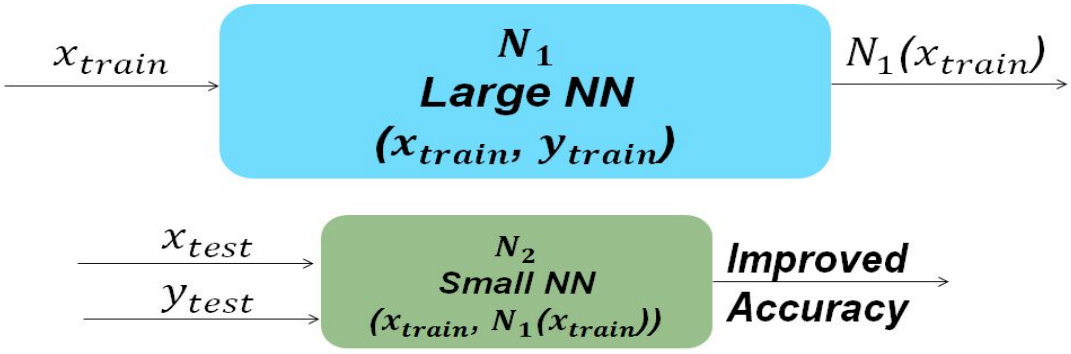


Figure 2. **Simple model of knowledge distillation method**

Experimental Results

Datasets and Setting

As a preliminary experiment, we evaluated the approaches on two different datasets: the standard MNIST digit recognition task and German Traffic Sign Benchmarks dataset. We used the standard split into 60k training images and 10k testing images. We use the first 10k images of the training set as a held-out validation set for hyper-parameter tuning and then retrained the models on the entire dataset with the best configuration.

For German Traffic Sign Benchmarks dataset, We used validation data with size of 5k, 32k training samples and 12k testing samples. We trained fully-connected, ReLu activation neural networks with 1200 hidden nodes for MNIST dataset and 1000 hidden nodes for GTSRB dataset. Models were optimized with stochastic gradient descent with a constant learning rate 0.05. In figure 2 you can see our some of our datasets for this project.

(a) MNIST

(b) GTSRB

Figure 2. Standard datasets for our experiments

Unregularized Neural Network

In this section, at first we tried to design different one layer neural networks with different hidden nodes, to find out which neural network works better for each dataset. Our evaluation methods here is test accuracy. Based on figure 3, we can see neural network with 1200 hidden nodes is best for MNIST dataset and neural network with 1000 hidden nodes works better for GTSRB. For following approaches, we used these optimized neural network to find out the best hyperparameters that work for them.

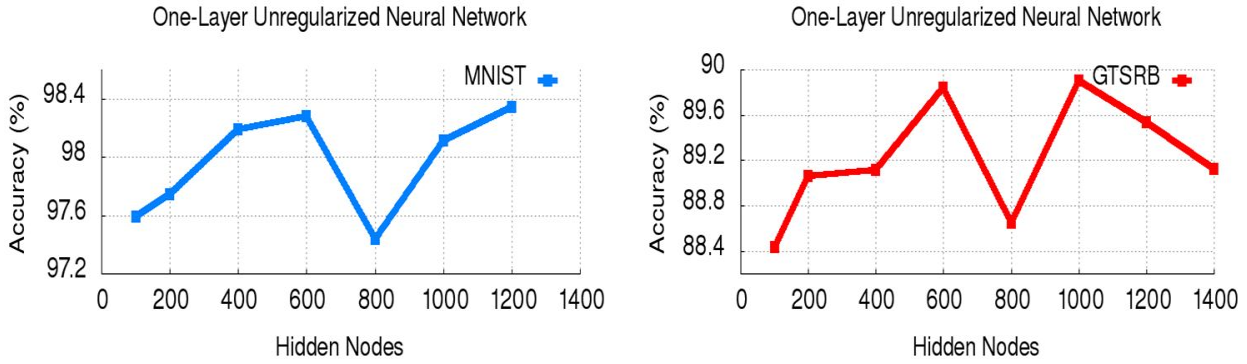


Figure 3. Accuracy of one-layer NN with different hidden nodes.

Regularized Neural Network

For label smoothing method, we changed smoothing parameter P in the range between 0.05 and 0.9, and we find out that P=0.05 has best test accuracy for MNIST dataset and 0.5 for GTSRB dataset. Figure 4 shows results for this experiment.

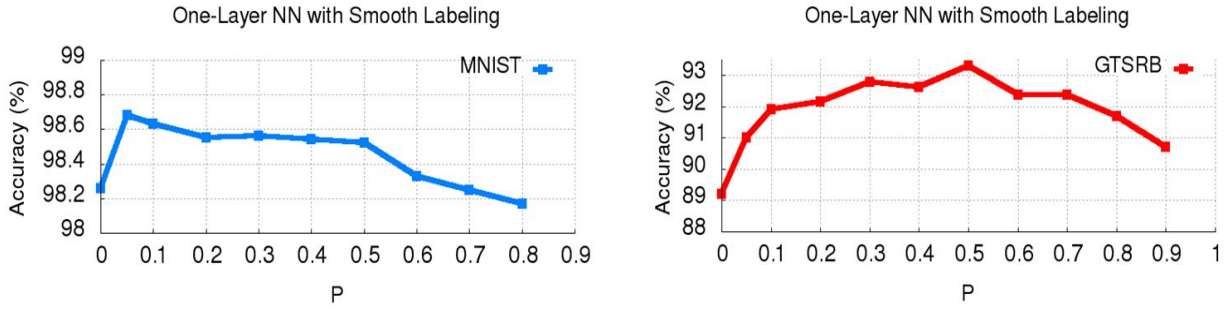


Figure 4. Accuracy of one-layer NN using Smooth Labeling method

For the confidence penalty, we varied the weight values over [0,0.1, 0.2, 0.5, 1.0, 2.0] and we find out that confidence penalty weight of 0.5 to work best for both datasets. Results of this experiment is shown in figure 5..

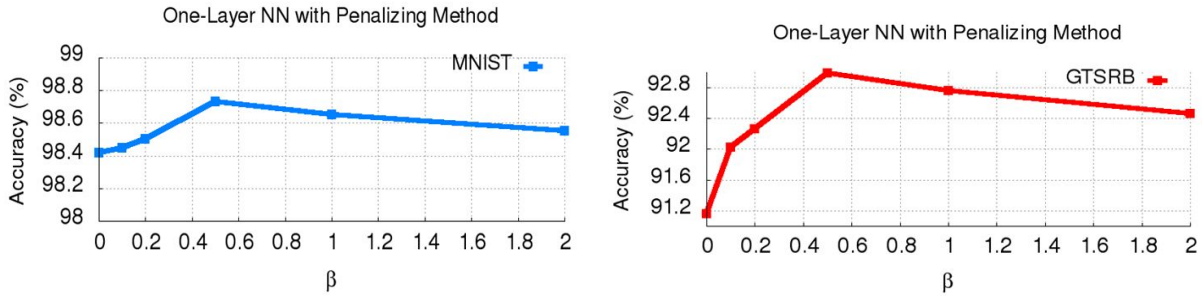
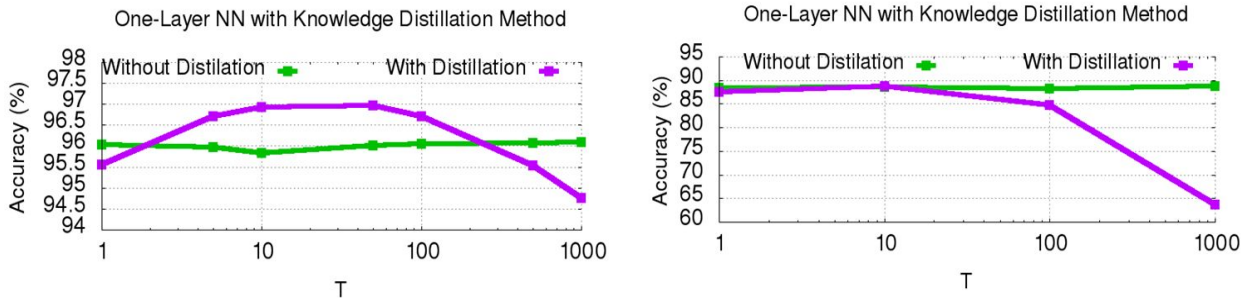


Figure 5. Accuracy of one-layer NN using penalizing the output method.

For Knowledge distillation, we run our experiment two times to see the effect of distillation and temperature (T), one of them neural network without distillation and the other is neural network with distillation. In knowledge distillation, we have two neural network first one is large neural network with 1000 hidden nodes and the small neural network has 100 hidden nodes. You can see the results of this experiment in figure 5. Considering two cases of using distillation versus using just a small NN, we can see accuracy improvement in both cases.



(a) GTSRB **(b) MNIST**

Figure 6. Comparison NN with and without distillation as a function of tempreture

Comparison

In this section, we compared unregularized neural network with our approaches in terms of accuracy for both datasets.

Model	Accuracy on MNIST (%)	Accuracy on GTSRB (%)
Unregularized NN	98.34	89.9
NN with Label Smoothing	98.63	93.32
NN with Confidence penalty	98.73	92.99
NN with Distillation	88.7(0.1% improvement)	96.9(1.1% improvement)

Table 1. Test accuracy comparison for different methods

Summary

Motivated by recent successes of output regularizers, we conduct a systematic evaluation of three output regularizers: label smoothing, the confidence penalty and the knowledge distillation. We show that this form of regularization, which has been shown to improve exploration in reinforcement learning, also acts as a strong regularizer in supervised learning. We find that the confidence penalty, label smoothing and the knowledge distillation can improve a wide range of state-of-the-art models by modifying hyper-parameters.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998. [on-line version]

[2] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011.

[3] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

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

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