On the Robustness of ConvNets to Training on Noisy Labels

David Flatow
Stanford University
dflatow@stanford.edu

Daniel Penner
Stanford University
dzpenner@stanford.edu

Abstract

In this work, we examine how sensitive the convolutional neural network model is to noise in the training set, particularly when the training set contains mislabeled or subjectively-labeled examples. We first test the robustness of the standard convolutional model by randomly permuting the labels of the training set with increasing frequency, and plotting the corresponding change in classification accuracy. Next, we implement a robust loss function proposed recently by Reed et al., which attempts to account for weak labeling in the training set by mixing the observed training targets with the model's predicted class probabilities. Finally, we implement an extension of the robust model which dynamically adjusts the mixing parameter, to increase our confidence in the model's predictions as they become more accurate.

1. Introduction

A major goal of computer vision concerns image classification, distinguishing images by assigning them labels based on their content. The majority of the successes acheived in this domain have been models trained on supervised, hand-labeled image datasets. More precisely, the images used to build these models (for example, the MNIST handwritten digits, and ImageNet) were assigned labels individually by people in a consistent manner. Extending these results to the vast majority of image datasets, which have either no human-assigned labels, or far less consistent labeling schemes, is a topic of much recent research in computer vision. In particular, some of the largest labeled image datasets available can be found on social media websites, where labels are assigned by vast numbers of different users, and as a result are far less consistent than those of hand-labeled datasets such as ImageNet.

While convolutional neural networks have been found to perform exceedingly well on datasets with consistent labelings, there is still work to be done on generalizing their performance to datasets with weak labels, where the labels do not consistently correspond to the image content. As a step in this direction, in this paper we investigate how sensitive the convolutional neural network model is to mislabeled training data, and examine some proposed methods for making the model more robust to weakly labeled images. A good understanding of how much error a convolutional neural network can handle in the training set could inform researchers as to how well they might expect the model to work on real world, highly imprecise image data.

When building a classification model for weakly labeled data such as social media photos, a typical approach would be to train the model on a dataset with more precise labeling, and then to use it to generate predictions on a more general dataset. One limitation of this model is that the labels found in the test data (or real-world data) are often not included in the training set, so that it becomes difficult to generate predictions outside the label domain of the training set.

Another possible approach is to train on a sample of the weakly-labeled data, and then use the model to predict on a held-out test set. This model is not inherently limited to making predictions on a smaller, precisely-labeled subset of the real-world data. However, including weakly-labeled or mislabeled data in the training set could very likely ruin the prediction performance of the model.

As such, there would be a number of benefits to knowing the extent to which convolutional neural networks are robust to weakly-labeled or mislabeled data being included in the training set. First, many of the largest image datasets available are those extracted from social networking websites such as Facebook, Instagram, and Flickr. However, as these datasets' images are labeled by users, building a consistent model on a precisely labeled training set is very labor-intensive, and also may not be generalizable to a broader set of social media images, due to the issues discussed above.

2. Related Work

The motivations for research on training machine learning algorithms on noisy labels are numerous. First, as documented by Sorokin and Forsyth, collecting labeled data can be costly. Convolutional neural networks, given their high model complexity, require large amounts of data to ef-

fectively train and thus are limited by the size of existing datasets. Large datasets of high quality are of great importance as is evidenced by the popularity of ImageNet. However, an alternative approach to obtaining training data is to tap into existing datasets (often user generated datasets). However, these dataset are notoriously unreliable. Nonetheless, the scale of these datasets have the potential to offer new grounds for experimentation. Next, we discuss a few approaches for training in these noisy environments.

Previous work investigating the robustness of convolutional neural networks to noisy training data generally focuses on either introducing visual noise to the training data, or dealing with training data with many missing labels (semi-supervised learning). However, there have been a number of works dealing directly with the issue of weakly-labeled training set as well. For example, a paper published by Mnih & Hinton (2012) detailed robust loss functions for two particular types of mislabeled data points in a particular type of dataset (aerial images). A more general approach is found in a paper recently submitted for publication by Reed, et al. entitled "Training Deep Neural Networks on Noisy Labels with Bootstrapping," which explores the sensitivity of convolutional neural networks to noisy labels by simulating noise in a training set. The authors also propose a class of more general modified loss functions which they claim increases robustness to noisy labeling. The authors test their proposed model on a variety of publicly available datasets (MNIST handwritten digits, the Toronto Face Database, and the ILSVRC2014 detection challenge data). It is worth noting that the authors acknowledge the inspiration of a technique called minimum entropy regularization, originally detailed in papers by Grandvalet & Bengio (2005, 2006), which improves predictions on unlabeled examples without a generative model, instead incorporating the model's own predicted class probabilities to adjust targets. This is similar to the robust loss detailed by Reed, et al., which adjusts targets on weakly-labeled training data by weighting the weak labels with the predicted class probabilities.

We have implemented some of the loss functions detailed in the paper by Reed, et al., and have tested it on a subset of the ImageNet dataset, as we describe in section 4 below. The particular scheme for generating noise in ImageNet will be described in section 3.

3. Approach

In order to design experiments to test the robustness of the convolutional neural network model to weakly-labeled training data, it is necessary to design a scheme in which noise can be introduced to the training set in a controlled manner which can be measured. Though there is an abundance of weakly-labeled data (largely, as noted before, on social media websites), we lack a metric to determine precisely how noisy the labels of these datasets are. In the following sections, we discuss our considerations in modeling noise in the training set.

3.1. Sources of Noisy Labels

Our primary consideration is to simulate noise in a training set which is generated in a manner that mimics, in some way, the way noisy labels are generated in weakly-labeled datasets in the wild. To get an idea of how some of these labels might come about, we looked at a random set of labeled photos from Instagram, and examined the labels which we might consider to be weak (in the sense that they do not clearly reflect the content of the image).

Among the most common situations arises when a label in fact describes the situation of the image appropriately, but the vantage point of the camera obscures the content (e.g. a photo labeled "bicycle" taken from the point of view of a person riding a bicycle; a more apt descriptor of the visual content of the image may have been "road"). Another common situation arises when an image is given a description associated with its content only abstractly (e.g. an image of a man on a couch labeled "bear"). Unlike the first type of weak labeling described above, these labels have the additional weakness of mislabeling to a class that may share no visual features with the visually-appropriate class.

We considered the possibility of devising a metric of similarity between classes (this could be computed from the similarity of the class probabilities, for example), so that when an image of a class is intentionally mislabeled, the distribution from which its label is selected weights classes which are 'closer' to it more heavily than others. This would reflect the first situation, in which weakly assigned labels are still similar to the more appropriate label in some way. However, a more conservative approach is to view mislabelings in the second sense above, in which there is no apparent visual connection between the images of the two classes. In this scenario, we can permute the class labels without worrying about which specific mislabelings occur, and is thus also a simpler model (in addition to being more conservative, since the noise is arguably less controlled).

3.2. Simulating Noisy Labels

In this section we describe the particular approach we took to simulate noise in the training set. As explained above, we decided not to distinguish between classes when intentionally mislabeling data, instead choosing new labels from a uniform distribution over the labels. We propose that one can approximate the effect of weakly labeled training data on prediction performance by randomly permuting the labels of subsets of strongly labeled datasets. In particular, we plan to train a sequence of N convolutional neural networks M_n on a subset of the ImageNet dataset, such that every training data point in model M_n is randomly as-

signed a new label with probability p_n, where p_n will range from zero to one. After obtaining test set prediction errors for each model M_n, we can get an idea of the relationship between the mislabeling metric parameter p and the model prediction error.

We anticipated, at the very least, that increasing p will increase prediction error. However, the nature of this increasing function is what we are interested in. If the increase in error can be well-approximated by a convex function, it will strongly suggest that standard convolutional neural networks are relatively robust to mislabeled training data. Otherwise, if the relationship between mislabeling and prediction error can be better approximated by a linear (or concave) function, the results will suggest that we can expect our prediction error to increase consistently (or rapidly) as the proportion of mislabeled data increases. Such a result would inform the use of convolutional neural networks in practice for classifying social network image data, indicating how weakly-labeled of a dataset can be used for model training. We will discuss the results of this experiment in section 4.2 below.

3.3. Measuring the Results

Of course, assessing the robustness of convolutional neural networks to weakly labeled data by measuring its robustness to explicitly mislabeled data is at best an approximation to the truth. However, it seems likely that the sensitivity to weak labels should be bounded below by the sensitivity to explicitly mislabeled data. That is, the data labels on social media sites are likely to be of classes related to the content of the actual image, and not explicitly mislabeled. A possible extension of our work would be to limit the random reassignment of image labels to labels similar to the correct ones (in accordance with that social media users do not randomly mislabel their images), or to instead introduce random noise images (from social media, or from a held out set) to each training set class with increasing proportions, and see how this changes the classification error.

4. Experiments

4.1. Testing Robustness of ConvNets to Noisy Labels

Our first experiment was concerned with determining the relationship between frequency of label permutation and classification accuracy of a standard convolutional neural network model. In this section we detail the approach, considerations, and result

4.1.1 Model Considerations

As we compare the models of varying permutation frequency p, there are a number of considerations which can adversely affect the results of our experiment, and thus need to be treated carefully, or at least acknowledged.

The first, and most problematic, is the determination of appropriate model hyperparameters.

First, we we will train a standard convolutional neural network on a subset of the ImageNet dataset, and tune the model's hyperparameters to achieve maximum validation accuracy, as usual. Our initial assumption is that for very small changes in p (the mislabeling proportion), the optimality of a set of model hyperparameters will not change much. With this admittedly naïve assumption, we will train a sequence M_n of models as described in the previous section, keeping the hyperparameters the same throughout, and storing the prediction error of each model on a fixed test set.

It is of course possible that the result is very sensitive to hyperparameter settings as the frequency of permutation p_n increases, and one could take a number of approaches to fix this. One would be to optimize at each model M_n, to produce a sequence of hyperparameter-optimized models. A benefit of this is that it gives us a better approximation of the upper bound of error at each n. Another option is to randomly sample many hyperparameter settings for each n, and to simply store all of them. Then one could produce a plot of the different outcomes for classification error for each n, and statistically fit a curve to the data to estimate what the general shape of the relationship between classification error and mislabeling rate is.

4.1.2 Implementation Details

As described above, we trained a sequence of three-layer convolutional neural networks which vary the frequency of permuted training labels. Specifically, the models were trained for 20 epochs using a softmax loss function, and the model's structure is two conv-relu-pool layers followed by an affine layer. While it is very possible that the optimal hyperparameter settings (specifically, learning rate, regularization strength, and batch size) are sensitive to the permutation frequency p, we decided to optimize the hyperparameters only for the initial, unpermuted model, and to hold the settings constant as we increased permutation frequency. While this may lead to suboptimal results, we can be reasonably confident that the accuracy val_n we observe for model M_n is an underestimate of what one would obtain if one were to re-optimize the hyperparameters, and so the curve we obtain estimating the decay of classification accuracy with increasing permutation frequency is likely a lower bound of the true decay curve.

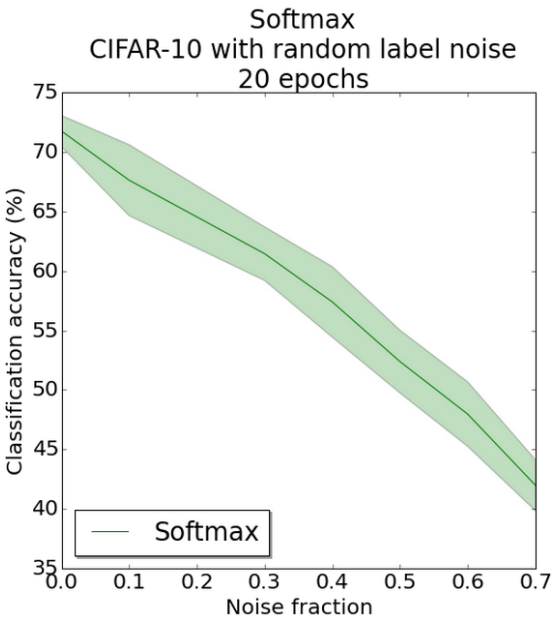

Figure 1. Average test accuracy over 5 repeated simulations with given parameters. Shaded regions represent ± 2 point-wise standard errors

4.1.3 Results

Here we present the results from varying the level of training label noise. With no training label noise the regular softmax model achieves a mean test error of 71%. Then, for increasing levels of label noise the training accuracy decreases. In our experiments the relationship appears to be approximately linear where an additional 10% noise corresponds to a 4% reduction in test accuracy. The normal baseline model is somewhat resistant to noise even without using a robust loss function. With noise levels of up to 50%, the model achieves roughly 50% test accuracy.

Softmax
Average test accuracy and standard deviations

noise fraction	mean accuracy	stdev
0.0	71.78	0.65
0.1	67.64	1.49
0.3	61.46	1.12
0.4	57.40	1.48
0.5	52.38	1.32
0.6	47.96	1.35
0.7	41.98	1.07

4.1.4 Topics for Further Research

A simple extension to this experiment would simply be to train a costlier and more accurate base model, for example using a much deeper network, training for far more epochs, or training on a bigger dataset (tiny ImageNet for a similar-sized set with more classes, or full ImageNet if feasible). Another possibility is to optimize the hyperparameter settings iteratively for each new model M_n, to try and obtain a tighter lower bound estimate for the decay curve of classification accuracy as p_n increases.

Another interesting approach would to find out at what level of the neural network hierarchy the mislabeled data does the most damage to the model. It is often hypothesized that in a deep convolutional neural network, the model 'learns to see' in the middle layers of the network, and adapts its understanding to the specific classes and dataset in the parameters closer to the network's output. It would be interesting to investigate whether training the model with randomly mislabeling data points only really affects the model's adaptation to the specific data, or if it actually affects the model's fundamental understanding of vision.

If the latter is the case, then we hypothesize that if instead of training a full model from scratch using the mislabeled dataset, we did transfer learning by attaching a model trained on correctly labeled data to our partially mislabeled training labels, we might actually obtain better classification performance. If this were the case, then an approach to classifying noisy social media datasets would be to train the model initially on a precisely labeled dataset, and then to use the parameters to initialize the final model on the noisy dataset.

Conversely one could also try training two convolutional neural networks, one on a weakly-labeled training set (in this case, with random label permutations) and one on a correctly labeled dataset, and transfering the resulting parameters to a model to be trained on a new dataset, examining how much worse the randomly permuted transfer model is.

4.2. Robust Models

In this section we detail the approach we used and the results we found by implementing the robust loss-adjusted convolutional neural network model proposed in the paper by Reed, et al.

4.2.1 The Model

The proposed model replaces the traditional softmax loss by a loss which takes into account for each data point (X_i, y_i) both the observed weak class label y_i (where we instead use an indicator notation with a vector t_i for which $t_{ij} = \mathbf{1}_{\{y_i=j\}}$), as well the vector p_i of class probabilities output by the model.

The paper proposes two different robust loss functions, which it denotes soft bootstrapping and hard bootstrapping, respectively.

The soft bootstrapping loss considers a convex combination of the quantities, $\beta t_{ij} + (1-\beta)p_{ij}$ as a metric for the membership of data point i belonging to class j. Here, β is a hyperparameter of the model which must be decided by cross-validation. We will discuss some results about β afterwards.

In accordance with the softmax loss, the loss function takes the form

$$L_{soft} = \sum_{i=1}^{n}\sum_{j=1}^{m}(\beta t_{ij} + (1-\beta)p_{ij})\log p_{ij}.$$

In contrast, the hard bootstrapping loss does not take into account the distribution of the model's predicted class probabilities over all classes, instead only taking the class of maximum predicted probability into account. Therefore the loss function takes the form

$$L_{hard} = \sum_{i=1}^{n}\sum_{j=1}^{m}(\beta t_{ij} + (1-\beta)z_{ij})\log p_{ij},$$

where $z_{ij} = \mathbf{1}_{\{j=\text{argmax}p_{ij}, j=1,\ldots,m\}}$ is an indicator for whether j is the class of highest predicted probability by the model for data point i.

4.2.2 Implementation Details

As in the previous experiment, we implemented sequences of convolutional neural network models M_n^{hard}, M_n^{soft} with proportion p_n of the training labels permuted, where the superscripts specify the robust loss function used in the implementation. To best be able to meaningfully compare the results of these experiments with those of the previous one, we changed as little as possible about the model's setup. In particular, the model structure and depth were held constant throughout, with two conv-relu-pool layers and one affine layer, and all experiments were ran for 20 epochs.

Though it is very likely that the optimal learning rate, regularization strength, and batch size change with the loss function and the mixture parameter β, we decided use the same hyperparameters from the previous experiment, optimized for the model with softmax loss and no permutation (that is, $p = 0$ and $\beta = 1$), and to hold these parameters constant on all modified models thereafter. We recognize that this is a possible confounding variable in the results of the experiment. However, as the classification accuracy for all models (that is, varying the choice of loss function and varying β) with $p = 0$ is roughly the same for the hyperparameter settings we used, it doesn't appear that we are losing too much information by keeping the same hyperparameters. As discussed earlier, the decaying classification accuracy with p we observe is likely an underestimate of the true accuracy. A possible topic for further investigation is to repeat these experiments, re-optimizing the hyperparameters as the we vary the loss function and β.

4.2.3 Annealing the mixture parameter β

Concerning the setting of the mixture parameter β in the robust loss functions, we first note that by taking $\beta = 1$, we can reduce the both the soft and hard bootstrapping loss functions reduce to the traditional softmax loss (and thus reduce the model to a standard convolutional neural network model). Reed, et al. specify that they found $\beta = 0.95$ to work best for the soft bootstrapping loss, and $\beta = 0.8$ for hard boostrapping, on the datasets they examined.

We trained models using the robust loss functions and a variety of settings of β, to see whether any improvements could be made by increasing the confidence allotted to the model predictions. In particular, in addition to $\beta = 1$, we trained models with $\beta = 0.9, 0.8$. The results of these preliminary experiments are described in the next section.

As an extension of the robust models proposed by Reed, et al., we examined the effect of annealing the value β over time, in a sense growing to trust the predictions of the model more (and the observed labels less) as we train for more epochs, and the model becomes more confident in its predictions. Of course, the annealing rate of β is a new model hyperparameter which we had to decide how to implement. Since $1-\beta$ reflects our confidence in the model's predictions, we concluded that (a) initializing $\beta = 1$ is sensible, since at initialization we do not trust our model's predictions at all; and (b) that β should decline as the validation accuracy increases. Since validation accuracy generally increases with the number of epochs, one possibility is to simply increase β by some fixed amount at each epoch. However, the natural setting $\beta = 1 - val$, where val is the validation accuracy, and since we compute val at each epoch in our training process, we simply set $\beta = 1 - val$ at the end of each epoch. The results of this extension are discussed in the following section as well.

4.2.4 Results

Here we examine the results from the two bootstrapping methods with the baseline softmax model. We highlight three models for this analysis. First, we compare hard bootrapping for different parameter settings of beta. We also compare the results to our two new proposed beta annealing methods. We find that for hard bootstrapping, mild annealing works best and for soft bootstrapping aggressive annealing worked best but the results were mixed.

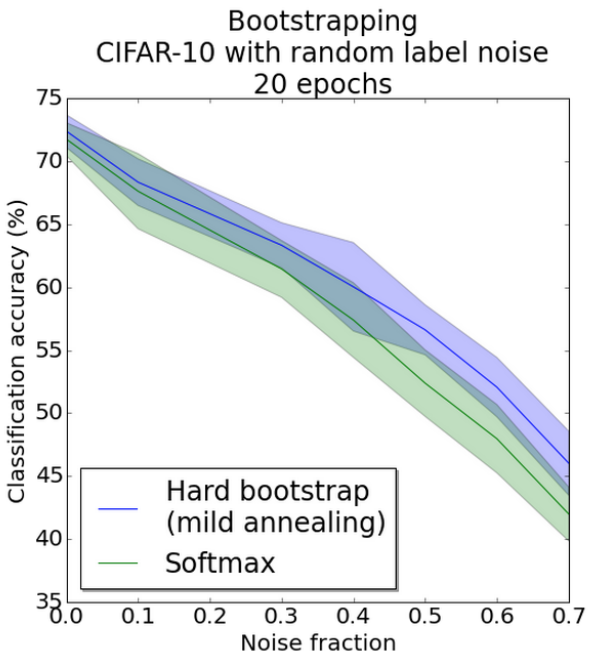

Figure 2. Average test accuracy over 5 repeated simulations with given parameters. Shaded regions represent ± 2 point-wise standard errors

Hard Bootstrapping Average test accuracy (%) relative to softmax			
anneal method (or beta level)	mild annealing	aggressive annealing	fixed beta=0.8
noise fraction			
0.0	0.625	-2.425	-0.625
0.1	2.250	-1.650	2.250
0.2	-0.175	-2.325	0.325
0.3	0.450	-2.750	-2.650
0.4	1.250	-0.500	0.100
0.5	-0.300	-0.800	3.000
0.6	1.400	-3.500	1.300
0.7	1.400	-4.400	1.900
0.8	3.900	-12.800	1.200

Soft Bootstrapping Average test accuracy (%) relative to softmax			
anneal method (or beta level)	mild annealing	aggressive annealing	fixed beta=0.8
noise fraction			
0.0	-0.525	-0.175	1.525
0.1	0.050	-0.150	0.350
0.2	-0.975	-1.175	-0.075
0.3	-1.750	0.950	-1.250
0.4	1.000	0.500	-0.250
0.5	-1.400	2.500	-0.100
0.6	-0.250	-0.500	0.600
0.7	1.600	3.100	0.500
0.8	3.600	4.100	3.200

5. Conclusions

We conclude from our first experiment that the convolutional neural network model without any modification is surprisingly robust to random noise in the training labels. As shown in the plot of figure 1, we observed that permuting 70% of the training labels caused less than a 50% decrease in classification accuracy. We speculate that if we had trained deeper models with higher initial classification accuracy (that is, with $p = 0$), this proportional rate of decrease in accuracy would persist. This guess is supported by the results found by Reed, et al. in their work, as well.

In our second experiment, we found that between the two robust loss functions proposed, the hard bootstrapping loss performed consistently better than the standard softmax loss function, while the soft bootstrapping yielded mixed results. We observed, as well, that our modification of annealing the mixture parameter β as validation accuracy improves the model's robustness to noisy training labels further.

References

[1] Chatzilari, E.; Nikolopoulos, S.; Patras, I.; Kompatsiaris, I. Leveraging social media for scalable object detection. *Pattern Recognition 45*, 2012.

[2] Cour, Timothee; Sapp, Benjamin; Jordan, Chris; Taskar, Ben. Learning from Ambiguously Labeled Images. *Technical Reports (CIS), Paper 902,* 2009.

[3] Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai; Li, Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR 2009, pp.248,255, 20-25*, June 2009.

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

[5] Hüllermeier, Eyke; Beringer, Jürgen. Learning from ambiguously labeled examples. *IDA-05, 6th International Symposium on Intelligent Data Analysis*, Madrid, Spain, 2005.

[6] Lee, Dong-Hyun. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 2013.

[7] Mnih, Volodymyr; Hinton, Geoffrey E. Learning to label aerial images from noisy data. *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 567-574, 2012.

[8] Sheng, Victor; Provost, Foster; Ipeirotis, Panagiotis. Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers. *KDD*, 2008.

[9] Sorokin, Alexander; Forsyth, David. Utility data annotation with Amazon Mechanical Turk. *Urbana 51.61*, pp. 820, 2008.