Training CNNs with Noisy Data

Ozge Yalcinkaya
Hacettepe University
Computer Engineering
ozge@cs.hacettepe.edu.tr

Abstract

In this work, by inspiring from the previously introduced works, we explore training a CNN with noisy labeled real world dataset. We add a linear layer on top of the proposed AlexNet architecture in order to decrease the adverse effect of noisy labels on learning process. Differently from previous works, we fine-tune the last fully connected layers instead of learning the model from the scratch. According to results, getting information from a model that is trained on clean data is beneficial and computationally cheap. We experiment on a small subset of NUS-WIDE dataset which is formed from internet images. As a result, proposed network increases the accuracy rate 10% compared to base model by jointly training the noise layer with the model.

1. Introduction

The need for massive data is increasing with the developing of learning processes. It has been proved to use a large data while training a deep model in many works which generally use existing datasets such as ImageNet [5]. Thus, the more data is used the better models will be trained. However, it is expensive to create required training data due to the labeling process.

As a solution, it is suggested to use the internet data which contains a huge number of images and videos on every topic that people share. People use some tags while sharing images for the purpose of specifying the content of the image. However, tags may not be relevant to the content of that image, because tagging is generally subjective.

Since the datasets are created by querying some predefined categories, we must rely on these tags of images which can be referred as noisy labels. For example, in Figure 1, all images are tagged as "library". However, some of them don't include any properties about "library". It is possible that those images are from a library or inside a library but they are not useful in order to learn a model for "library" category. Therefore, such examples are called as noisy labels.

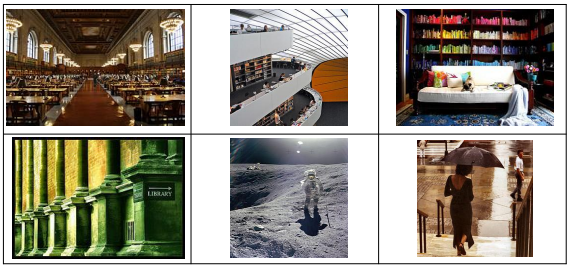


Figure 1. Images that are tagged with "library" from NUS-WIDE dataset. Top: appropriate samples. Below: irrelevant ones.

Figure 2. Images that are tagged with "palm" from NUS-WIDE dataset. Top: relevant samples which are palm trees. Below: noisy labels which include the palm of a hand.

In addition to irrelevant instances, there may be different category images that has the same tag such as "palm" due to being synonyms as in Figure 2. It can be a hand's part or a tree. Since we are collecting samples automatically by querying related tags, again we retrieve irrelevant images to that category. Hence, if the category is the palm tree, the rest is referred as noisy labeled images.

All of those noisy labeled images are misleading examples for model learning. They cause the model to be trained in a wrong way that test prediction results may be decrease. The reason is that, the noisy labeled image gives wrong information while optimizing the objective function. Thus, resulting model won't be able to predict true category images due to be updated wrongly.

In this work, by utilizing the methods from previous works [12, 7, 1], we decrease the effect of noisy labels on the training process of a Convolutional Neural Network. We basically add a linear layer on top of the CNN architecture which adapts the softmax outputs to match the noise distribution as in [12].

Since it is shown that using information from clean data is beneficial in [1], differently from [12], we don't train the network from scratch, instead we fine-tune last layers jointly with the linear noise layer.

In the following, first we give the related works on training with noisy labels. Then, we explore the applied approach in detail. Finally, we give the experimental results on the real world NUS-WIDE [4] dataset.

2. Related Works

In order to handle noisy data problem, some works suggest to eliminate those misleading samples and some of them propose more robust classifiers over SVMs, kNNs or logistic regression [6, 2, 10].

With the enhancement of neural networks [9, 11], CNNs are proposed as a generic solution for image classification. With the requirement of huge datasets in order to train these networks, training deep networks with weakly labeled data is also investigated. For that purpose, many methods have been introduced.

First of all, Sukhbaatar *et al*. [12] suggested to use a linear layer on top of the softmax layer which learns the noisy label distribution in the data and improves the prediction results. After the success of this work, similar approaches are introduced.

Jindal *et al*. [7] propose an architecture similar to [12]. In addition, they add a dropout regularizer after the softmax and before the noise layer which tends to give better results.

Meanwhile, Azadi *et al*. [1] utilized a pre-trained model of AlexNet in order to derive information from clean data. They also add an image regularizer at the end of the network and fine-tune the last layers. The regularization encourages the model to identify and discard incorrectly-labeled images.

In this work, we utilize the mentioned works. Similar to [1], we use pre-trained AlexNet and fine-tune the last layers with a noisy dataset. We use linear noise layer after the softmax in order to learn noise distribution [12] and we investigate the effect of dropout [7].

Furthermore, after the success of training CNNs with noisy data, many other approaches are introduced. Chen *et al*. [3] adapt curriculum learning approach to train CNNs with weakly labeled data. They train their network with easy samples and then they feed real world data for a more robust learning. As a result, their model gives higher accuracies than a fine-tuned network.

Xiao *et al*. [14] use a clean labeled data to train the network. Then, with a probabilistic graphical model, their system predicts the true label for a given noisy labeled sample.

Krause *et al*. [8] again use noise data in order to increase the prediction results for fine-grained recognition systems. Finally, Vo *et al*. [13], explore the training CNNs with a massive weakly labeled data by considering different architectures.

3. The Approach

In this section, first we give the background methods, that we utilize in this work, in detail. Then, we explore how we combine the ideas in order to decrease the adverse effect of noisy labels on training process of a CNN.

3.1. Background

Sukhbaatar *et al*. [12] shows that using a linear noise layer after the softmax makes the network to be trained in a suitable way for noisy labels. It basically adapts the outputs by matching noise distribution. Since it is a linear layer, they say that noise distribution is learned automatically with back-propagation.

They learn noise distribution Q which corresponds to weights of noise layer. It is a KxK probability transition matrix where K is the label amount. Hence, final prediction of an input depends on the both model parameters θ and noise distribution Q.

This Q changes output probabilities that come from softmax into a distribution that better matches with the noisy labels. In order to learn this noise weight matrix Q, they use confusion matrix C, on the probability matrix Q. Hence, $QC = Q^*$ where Q^* is the true noise distribution.

C actually comes from a trained model for 5 or 10 epochs. Thus, they train their model upto some point in order to learn noisy distribution first and fit the model on noisy data. Until this point, Q is an identity matrix so that there is not any linear calculations that change the results.

After learning the noise distribution, they start to update Q weight matrix and the noise layer changes the output of the softmax to a better distribution that matches with noisy labels. Basically, linear noise layer forces the C to be close to an identity matrix.

They train the network by maximizing the cross-entropy between model prediction and given noisy labels. Hence, loss function includes both base model parameters and the Q weight matrix of linear noise layer ($\mathcal{L} = (\theta, Q)$) and Q is learned jointly with the CNN.

However, they imply that it is not enough to minimize the loss. Therefore, they also add a regularizer which forces Q to converge in Q^*. This regularizer is the weight decay on Q with 0.1. Note that, they use AlexNet [9] architecture and train the entire network from the scratch.

Jindal *et al*. [7], follows the same structure except they use a dropout regularization after the base model's softmax layer in order to avoid the model to overfit on noisy labels.

In addition, they minimize a different loss function. They basically add another softmax layer after the linear layer and then minimize the cost. Therefore, their objective function is $\mathcal{L} = (\theta, \sigma(Q))$ where σ is the softmax. Since we use the same architecture, the details can be seen in Section 3.2.

Other than mentioned works, **Azadi** *et al*. [1] proposed to use a pre-trained network in order to decrease the impact of noisy labels. They use AlexNet architecture with the weights that are trained on ISLVRC2012 dataset (ImageNet) and fine-tune the last layers with an auxiliary image regularizer.

They imply that getting activations from a network that contains clean data information is beneficial and more successful. Similarly, by inspiring from this work, we use a pre-trained network while training the noise layer.

All of the works use a noise layer or a regularizer while training the network. Then, in the test phase, they remove this last layers and get predictions from the trained base model.

3.2. Used Method

As it is mentioned before, we basically combine three ideas from three different state-of-the art works (Section 3.1). First of all, we use AlexNet architecture trained on clean ImageNet dataset (ISLVRC2012). We get base model results by fine-tuning the $fc8$ and $fc7$ layers of this network with noisy training data.

We also learn the C which is mentioned in [12] after 50 epochs and use these weights while training the noise model. Detailed experiments are given in Section 4.

We use the same architecture with [7] and an overview of the model can be seen in Figure 3. Instead of training the network from scratch, we use pre-trained weights. Moreover, we don't use the proposed dropout regularization since our network does not converge. Putting a dropout layer after the last softmax layer of base model causes the network not to fit on noisy data. However, since we use pre-trained weights, fitting on this noisy data is important rather than avoiding it to overfit. Therefore, it may not be useful for our model.

In the network [7], firstly, they define the usual softmax function as

$$\sigma(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}. \tag{1}$$

Then, the base model predicts the related label for a test image x by using this traditional probability distribution:

$$p(\hat{y}|x;\theta) = \sigma(h) \tag{2}$$

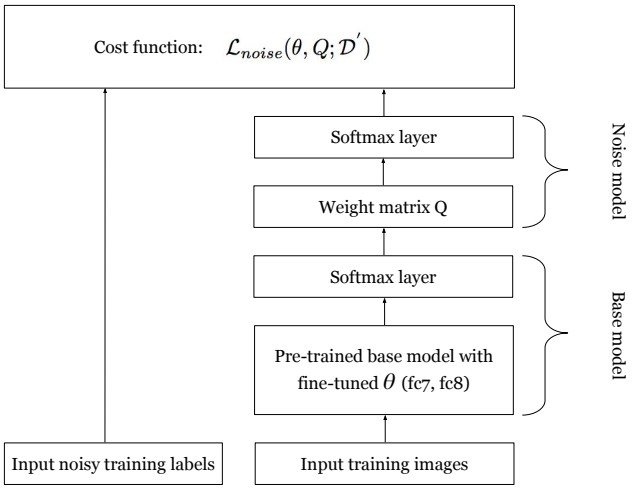


Figure 3. An overview of the used model.

where the h is the output vector of $fc8$ layer of base model, $\hat{y}$ is the true noise label and θ represents the parameters of base model which corresponds to weight of $fc7$ and $fc8$ in our work. Therefore, in order to train this base network the loss function will be

$$\mathcal{L}_{base}(\theta; \mathcal{D}') = -\frac{1}{n}\sum_{i=1}^{n} \log(\sigma(h)_{y_i'}) \tag{3}$$

where $\mathcal{D}'$ is the noisy data. However, we want to use only the clean label information while optimizing this objective. Hence, after that, they add the linear noise layer which only apply the weight matrix Q on to the outputs of softmax. Since we derive information from clean labels, noisy or irrelevant images will be classified wrongly and this situation causes our model to be trained in a non accurate way. It has been shown that with this operation, the misleading effect of noisy labels is decreased. The reason is that, weight matrix is converged in a way that it forces wrongly predicted instances to be predicted with their true noise label.

Furthermore, they again put a softmax layer after the noise layer before calculating the cross-entropy. Consequently, for a learned Q weight matrix (noise distribution) and base model parameters θ, the estimate of the distribution of the noisy class label is

$$p(\hat{y}'|\hat{y}) = \sigma(Q.\sigma(h)). \tag{4}$$

Thus, with the cross-entropy loss, the resulting objective is

$$\mathcal{L}_{noise}(\theta, Q; \mathcal{D}') = -\frac{1}{n}\sum_{i=1}^{n} \log(\sigma(Q.\sigma(h))_{y_i'}). \tag{5}$$

Similarly, we use SGD in order to optimize this loss function. Moreover, we experiment with the weight decay on Q while updating it since in [12] they say that it helps to converge Q to Q^*.

4. Experiments

4.1. Dataset

We use a web image dataset which is collected from Flickr, NUS-WIDE [4]. The dataset includes 269,648 images and the associated tags from Flickr, with a total number of 5,018 unique tags. We extract a subset of NUS-WIDE by only using 10 classes: "computers", "desert", "door", "library", "oceans", "palm", "railroad", "sailboats", "tanker", "tent".

We separate this subset into train, validation and test sets. For each category, there are 250 images for training set and 20 images for test and validation sets. While creating validation and test sets, we choose the most representative instances manually and be sure there is not any noisy labeled image, in order to compare base and noise models accurately.

4.2. Implementation

We implement the proposed method in Tensorflow framework. We use provided AlexNet model and weights in [1]. We feed training images by shuffling them and at each step we calculate the validation accuracy in order to monitor the learning process. We do not use data augmentation since the training data already contains many variations. Because of that reason, we have not faced with overfitting problem in early epochs. After some point, we utilize early stopping in both models.

We choose batch size as 128 and learning rate as 0.01. We initialize convolutional layer weights from pre-trained model and we fine-tune last layers (fc7, fc8) by randomly initializing them. The noise layer's weight matrix Q is initialized as identity matrix for 50 epochs. Then, it is started to be updated jointly with the fully connected layers. This epoch number of training process to learn C is determined by the experiments. Detailed results are given in Section 4.4.

4.3. Baseline

Baseline is calculated by fine-tuning the base model. We apply early stopping in order to obtain the best model. As it can be seen from the Figure 4, validation loss is stop to decrease after some point. Hence, we get the weights from 150^{th} epoch and test prediction accuracy is calculated as "61%".

[1] http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/

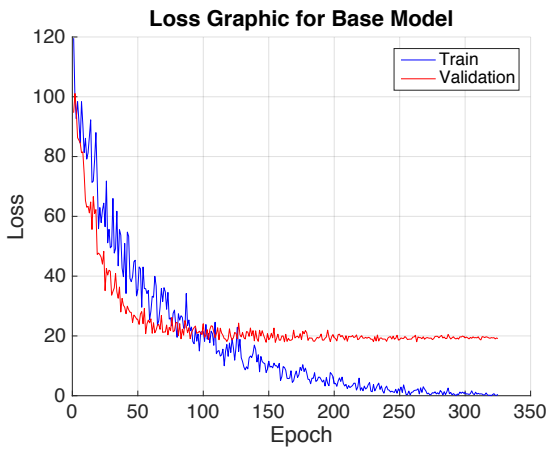


Figure 4. A graphic for train and validation loss throughout the training steps of base model.

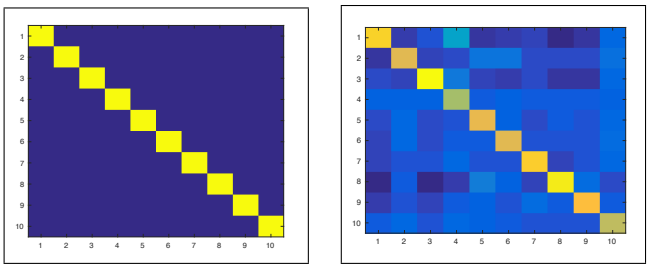
4.4. Noise Layer

After training the base model, we experiment with different initializations for learning the noise model. As it is mentioned before, we derive weights for $fc7$ and $fc8$ which are fine-tuned on noisy data. Then, we start to update Q weight matrix for learning the noise model. Since we get clean information from convolutional layers of pre-trained model, it is important to fine-tune the network with more epochs than 5 which is stated in [12], in order to learn noise distribution.

Afterwards, Q can learn to decrease the adverse effect of noisy labels. It is also essential not to use weights that learned noisy data too much. Therefore, according to experiments, 50^{th} epoch weights are proper to train the noise layer (Table 2) since they give the best test accuracy. In Figure 6, we give loss graphic for noise model's learning process. For each noise model experiment, we again apply early stopping after the convergence of validation accuracy. Here, we can see that validation loss is started to increase after 250^{th} epoch. Therefore, while calculating the test accuracy, we use the weights from 250^{th} epoch.

Moreover, we can see that train loss is higher than validation loss. It actually proves that our noise model learns noise distribution and fits the deep model to the clean instances. Hence, we get higher accuracies for validation or test set. We also give the visualization of the weight matrix Q for its initial and trained forms in Figure 5. We can see that Q matrix is converged in a way that to fix prediction results that come from softmax. The values are evolved so that matrix is far from identity which takes the same probability values from softmax.

In addition, we investigate the effect of weight decay regularization on Q as it is suggested in [12] in order to avoid overfitting on noise distribution. In Figure 7, we give the



(a) (b)

Figure 5. Initial identity weight matrix (a) and the trained noise distribution matrix for noise layer (b).

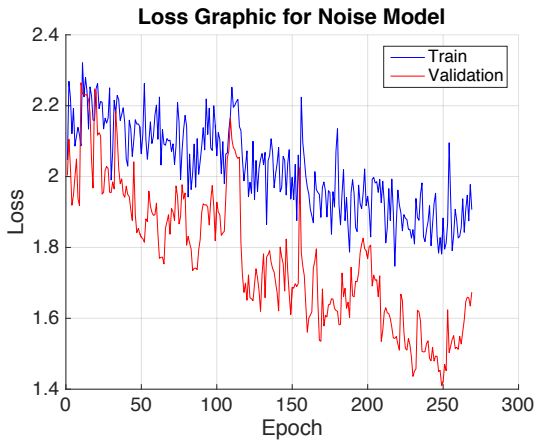


Figure 6. A graphic for train and validation loss throughout the training steps of noise model.

Table 1. Noise model accuracies for different base model weights. Baseline: 61%

Base Model Epoch	Noise Model Accuracy
150	62%
100	64.5%
50	**71%**

loss curves for train and validation. We can see that weight decay considerably avoids the oscillation in learning and provides a faster convergence. Therefore, the same result can be obtained in early epochs. However, it seems loss is not decreasing as it is decreased in previous model for the first 300 epochs and it should be considered further.

4.5. Result

As a result, we obtain C from the proposed base model by utilizing the weights of 50^{th} epoch. Then, we update Q by calculating $QC = Q^*$ and we stop the learning at 250^{th} epoch when validation loss is started to increase. Thus, we get 10% higher test accuracy than the base model by training our noise model. The final results can be seen in Table 2.

We can say that base model has not enough capacity to

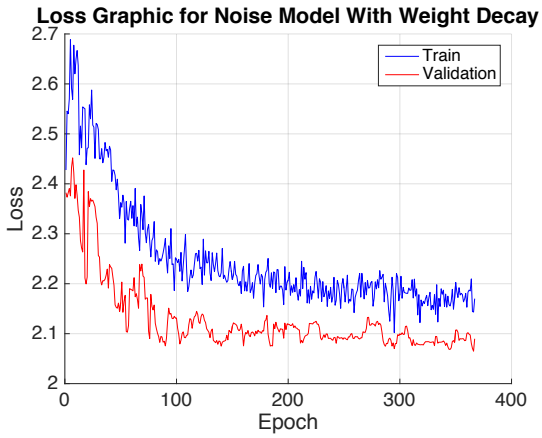


Figure 7. Loss curve for noise model with weight decay on Q.

Table 2. Base model test accuracy vs noise model test accuracy.

Base Model Accuracy	Noise Model Accuracy
61%	**71%**

do accurate training on a noisy data since it is overfitting after 150^{th} epoch. In contrast, by using the benefits of noise layer, we are able to train the model up to 300^{th} epoch and get higher accuracies on test set.

5. Conclusion

In this work, we explore the negative effects of noisy labeled datasets while training CNNs. Then, by utilizing and combining previously proposed state-of-the-art methods, we show the enhancement of the training process over a real world image dataset. We show the benefits of using a pre-trained model by decreasing the number of model parameters and deriving clean label information which makes our model to learn noise distribution better. Just by adding a linear layer, we make the base model more robust to noisy labels. Since we will need larger datasets for accurate predictions, improving the methods on this problem will be essential. Therefore, in the future, more deeper and state-of-the-art network architectures should be considered with novel noise models or approaches.

References

[1] S. Azadi, J. Feng, S. Jegelka, and T. Darrell. Auxiliary image regularization for deep cnns with noisy labels. *arXiv preprint arXiv:1511.07069*, 2015.

[2] J. Bootkrajang and A. Kabán. Label-noise robust logistic regression and its applications. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 143–158. Springer, 2012.

[3] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1431–1439, 2015.

[4] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., 2009.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pages 248–255. IEEE, 2009.

[6] B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[7] I. Jindal, M. Nokleby, and X. Chen. Learning deep networks from noisy labels with dropout regularization. In *IEEE 16th International Conference on Data Mining (ICDM), 2016*, pages 967–972. IEEE, 2016.

[8] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision*, pages 301–320. Springer, 2016.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[13] P. D. Vo, A. Ginsca, H. Le Borgne, and A. Popescu. Harnessing noisy web images for deep representation. *Computer Vision and Image Understanding*, 2017.

[14] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.