ROBUST CLASSIFIERS FOR HISTOPATHOLOGY DATA

Gurparkash Singh
160050112
Department of Computer Science
Indian Institute of Technology Bombay

June 23, 2020

ABSTRACT

Traditional Deep Learning Algorithms require huge and reliable training datasets to perform with high accuracy. Obtaining such volumes of data, however, is not trivial and usually involves an error prone automatic or a manual labeling process. This is especially true for Medical Datasets, labelling which requires laborious efforts from multiple institutions. Because of the difference in conventions of various labs, the resulting datasets tend to be noisy. Since Deep Learning algorithms are highly adept at function estimation, they also tend to fit to the noisy samples of the data. This is known as over-fitting. The noise in an image-dataset can either be in the visual data or in the labels. The variance in appearance is often tackled by using classical techniques such as Color Normalization or Color Augmentation. In this project, I explore various techniques to train classifiers reliably in the presence of noisy labels. The resulting models are called Robust Classifiers and I demonstrate their efficacy on the classification task of breast-cancer detection.

1 Introduction

The aim of Image Analysis is to automate several tasks traditionally done by human experts. Over the past decade, a lot of tasks have been successfully automated, primarily due to the rise of Deep Learning algorithms. However, Deep Learning algorithms require large and reliable datasets to learn the hidden patterns to make correct decisions consistently. The unreliability in Histopathology datasets which results from the variance in appearance has greatly affected the usage of Deep Networks on these tasks. This variance in datasets arises from the different methods of slide preparation used in various labs. Even for a single lab, the methods used may vary over time. Furthermore, the difference in anatomy among different patients results in different appearance of samples for different people. Traditional methods, such as color normalization and color augmentation have been used to reduce this variability.

In color normalization, the training dataset is normalized before training the Neural Network. The test dataset is also normalized using the mean and variance of the training dataset. This method accounts for the variance in color between samples. In color augmentation, random jitters are made to each channel of each image as follows. $I_c \leftarrow a_c.I_c + b_c$, where a_c and b_c are drawn from uniform distributions a_c ~U $[0.9, 1.1]$ and b_c ~U $[-10, +10]$. By adding color augmentation to the data, we can expect the network to learn the trend and ignore the color variation and hence, perform better on the test-set. These methods sound promising, but their success rate has been limited. These methods can reduce the variance in appearance but are not helpful to alleviate the issues stemming from noisy labels.

The images that are prepared from slides are giga-images and analysis of such large images is unfeasible. So, what we do is that we crop one large image into several small images and each of the small image is given the same label as that of the original large image. Now, this is a source of noise in the labels. For example, say the large image has a small part of it which contains cancerous cells, so it's label would be positive. However, many of its cropped images would have no cancerous cells and still, will have positive labels. This is undesirable and we need a method which can work well even in the presence of noisy labels. Color Normalization is often used to deal with the variance in appearance of the images, however no technique has been firmly established to deal with noisy labels. This is the motivation behind coming up with a novel weighted loss function. The weights are learned in a way that the data points that contribute positively to adaptation will have higher weights.

2 Prior Work

2.1 Curriculum Learning

Using cropped segments and the labels of large full-sized images to train any Neural Networks introduces noise in the labels of our dataset. This means that there will be several images in the dataset which will not have cancerous cells but still, their label will be positive. This is a common problem in any task which uses such cropped datasets. Thus, we have to find a method to train our neural networks in a way that can tackle the noise in the dataset. Curriculum Learning [5] is one such paradigm to help us in alleviating this problem. Let us first see how we can train a vanilla classification network on a noisy dataset and later, we will extend the framework for our problem.

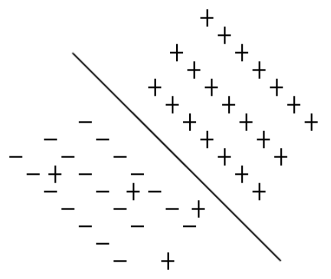

Figure 1: Dataset with noisy labels

First, let us define the notion of a difficult example. Let us assume that we have a noise free dataset. Therefore, the images with cancerous cells will be labeled as positive whereas the images without cancerous cells will be labeled negative. Now, if we were to train a classification network on this dataset, it will be reasonably easy since we have a distinguishing feature between the two classes. Now, if we label some of the negative examples as positive, the network will have a difficult time trying to separate those examples as they effectively look the same as those with negative labels. Even if the network is able to fit well to the training data, it will just be over-fitting. Now, we call these data points, which are noisy as difficult. The loss function value for these difficult points will be higher than those which are easy and hence classified correctly. We refer the data-points which are not noisy as learn-able.

A simple and logical way to deal with noisy labels is to ignore the difficult data points and to only train on the easy data points. This is exactly what we need since the network will now be learning only on the correct positive and correct negative data points. This can be achieved by setting an upper-bound on the loss value of a point. This value will be a hyper-parameter and let us denote it by γ. As usual, we will initiate the weights appropriately and on each iteration, we will calculate the value of loss function for each training example. After this, however, the weight update will only be made on the basis of those data-points for which the value of the loss function is below the threshold γ. This way, the network will start with learning on easiest of data-points in the datasets and will progressively start learning from the more difficult data-points. The value of γ denotes the progression of learning of the network, also known as the curriculum.

The above method described how to train a neural network only on the learn-able examples. However, the problem of domain adaptation is such that we also want to consider the transferability of data-points. The difficult points of the dataset will also negatively affect our learned model since the model is very likely to overfit to those noisy examples. The learnability of a data-point as described above is indicated by the the score of the target class given by the classifier. The notion of transferability is slightly more complex. For this, we have to think about which kind of data-points would make our learned classifier more robust. The data-points which are not easily distinguishable by the distinguisher component D are the data-points we want to use to train the weights of classifier C and feature generator F. The transferability of a data-point in class A is indicated by the score given for class B by D. This would mean that the features for that point are not easily distinguished. So, the new equation for selecting data-points is $S_L + \lambda S_T < \gamma$, where λ is the hyper-parameter corresponding to the weighing factor and S_L and S_T are the learnability and transferability of a data-point. Therefore, for suitable values of γ and λ, only the noise free points will be used to update the weights making the learned model more robust.

2.2 Adversarial Autoencoders

Adversarial Autoencoders [8] are used to fit the features to a prior distribution in an unsupervised way. They take motivation from Generative Adversarial Networks and use adversarial training to achieve their goal. The vanilla version of an Adversarial Autoencoder consists of three basic elements: Encoder, Decoder, and Discriminator. The encoder network simply consists of stacked Convolutional Layers which project the input to a feature space. The output from the encoder are the features corresponding to the input image. The Decoder takes the features and tries to reconstruct the original input image using a network of Transpose-Convolutional Layers. The Discriminator is a fully connected network which tries to distinguish between the output of the encoder and samples from the prior which are of the same dimensions as the output of the encoder.

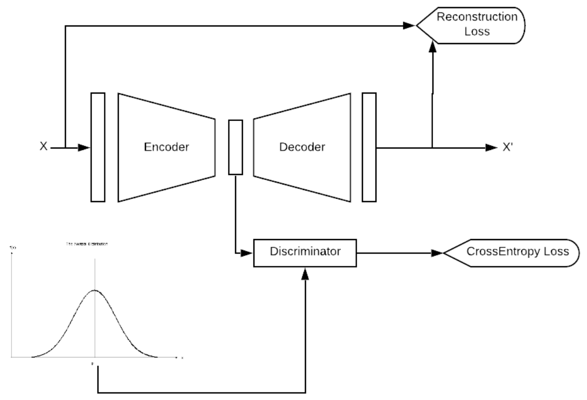

Figure 2: Vanilla Adversarial Autoencoders

The basic principle is that if the encoder is able to construct features that the discriminator fails to separate from the prior distribution, then the distribution of our features will closely follow our prior. To achieve this, the training is done in two steps. In the first step, the parameters of the discriminator are updated to distinguish the prior and features. In the next step, the rest of the network is updated. The decoder is simply updated to minimize the MSE Loss between its output and the original image. The encoder is updated to both decrease the loss of the Decoder Network as well as increase the loss of the Discriminator. This is known as the reverse gradient step. If the Encoder generates the features that reduce the accuracy of the discriminator, this would mean that it has achieved its goal of generating features that mimic our prior distribution.

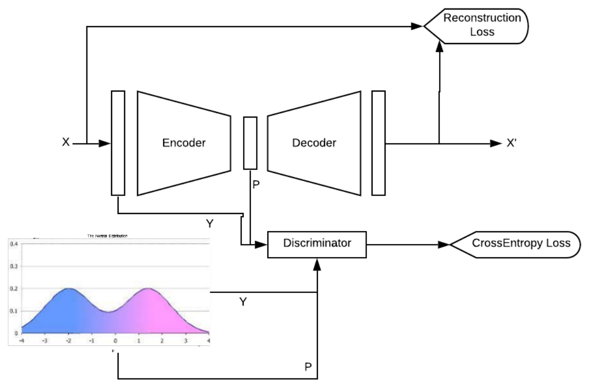

Figure 3: Adversarial Autoencoders with Multi-Modal Prior

The vanilla version of the Adversarial Autoencoders is not very useful for us, since we need features in which the classes can also be distinguished. For this purpose, we try to fit our binary classification dataset to a bi-modal prior, one mode for each class. This can be used by a small modification to our discriminator which now also takes the label of an image as input. The sample from the prior also comes appended with the one-hot label corresponding to the mode from which the sample has been taken. Training this in the similar way, we can hope that the features for an image from a class will coincide towards the corresponding mode from the prior. If this is indeed the case, we can train a classifier over these features once the Autoencoder system has been completely trained.

3 Motivation

As it happens, simply using the Adversarial Neural Networks to fit the features to a bi-modal Gaussian Distribution and training a classifier separately over these features does not perform very well in the presence of noise. One of the main issues with this method is that once we've obtained the features and fixed a threshold, the weights will be same for each epoch. This principle is different than the one discussed in Curriculum Learning in which the weights changed after each iteration with the change in the underlying features and the model learned from easier to the more difficult points in a gradual manner. Here, since the features are fixed, such gradual change does not occur and the model ends up over-fitting to the features corresponding to the noisy labels. Therefore, we need to modify the approach to build a model in which a classifier can learn in a gradual and iterative way. This will also help us to obtain better features that are more suited to the task at hand rather than at the secondary task of fitting to a prior distribution.

4 Methodologies

As we discussed above, the features from Adversarial Autoencoders do not suffice to train a classifier on their own. Therefore, we make a small modification to the network and add a classifier to the previous model. This classifier takes as input the features from the encoder and tries to classify the features. The training remains the same except a small addendum. The second step now also has an added term of the Cross Entropy Loss from the classifier which it tries to increase. After proper training, we can expect this model to be a solid base model for a classifier. The only problem here is that it does not account for the noisy labels in our dataset.

The basic principle of Adversarial Autoencoders is that they can be used to fit a given dataset to a prior probability distribution in an unsupervised way. If we have a labeled dataset, we can also use those labels to fit the data to a multi-modal prior in a supervised way. The hypothesis of our method is that an incorrectly labeled sample will fit poorly to the corresponding mode of its class. Therefore, the value of the probability distribution function can be used to weigh the points in our weighted loss function. If our hypothesis is correct, the features corresponding to the noisy labeled inputs will fit poorly to the prior, and thus will have smaller weight values. We can use these weights appropriately in a weighted cross entropy loss function for updating the encoder and classifier. I used a bi-modal Gaussian distribution as a prior and the weights are calculated by calculating the value of the probability distribution function corresponding to the class. I further discuss the various ways in which we can formulate the weights in our loss function.

- **Normalized Weights :** Usually we train by using mini batches of data. Therefore, since the weights corresponding to all the data-points will be small to start with, we can divide each of them with their sum. This way, the model will learn at a uniform rate throughout. Furthermore, this method has regularization properties. This is owing to the fact that the weights are dependent upon other members of the mini-batch, similar to batch-normalization.

- **Binary Weights :** In addition to the above method of normalizing weights, we can also convert them into binary values by thresh-holding on a pre-decided value. This will mean that we only train on the points which fit well to our prior. We expect this number of points to increase over the training period as the Encoder improves. This is the closest method in spirit to the original Curriculum Learning Method.

- **Time-Annealed Variance :** We can start off by fitting features to a distribution with higher variance and slowly decrease the variance, so that the features aggregate gradually. This approach will give sufficient weight to the features to start with and slowly converge to the prior distribution. I found this method difficult to train and therefore, left out of the observations.

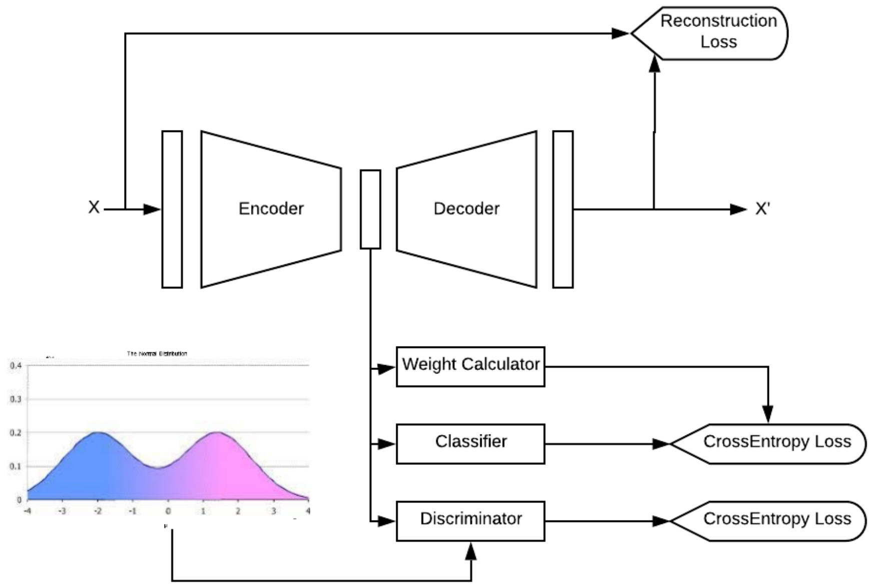

Figure 4: The Proposed Architecture

5 Experiments and Observations

5.1 Dataset

I'm using the Break-Hist Cancer dataset [6] to train and test my approach. The Break-Hist dataset contains Histopathology scans of various zoom levels. I use the 400X zoom for my models. The dataset contains scans of tissues for breast cancer, labeled either positive or negative. The images are all sized equally at 700x460 pixels. The data splits are as follows :

	Train	Test	Eval	Total
Benign	470	59	59	588
Malign	985	124	123	1232
Total	1455	183	182	1820

5.2 Experiments and Results

I programmed and tested various variations of our method and compared it against a few standard networks. Here are the models that I tested against and their results.

- **Convolutional Networks :** Convolutional Networks that use batch-normalization and leaky-ReLU activation function perform the best amongst the various other versions. However, the performance significantly drops with an increase in the label noise of the dataset. The results are tabulated.
- **Adversarial Autoencoders (unimodal distribution) :** Since we are not projecting our data to class-sensitive prior, it is not expected to work well even on a dataset without noise. The results here are as expected. Even without noise, the accuracy is only about 70%.
- **Adversarial Autoencoders (bimodal distribution)** As expected that with a slightly complex dataset, extracting meaningful features get difficult and doesn't perform very well with even a small amount of noise. When the dataset is not noisy, the accuracy is 86% but the accuracy falls steeply with the increase in noise.
- **Our Model (with binary weights):** When compared against the other networks, our model performs consistently better even in the presence of noise. The results are tabulated.
- **Our Model (with normalized weights):** Even though we expect this model to work equally well, training it is not as easy. The current accuracy is not only lower than the one with binary weights but I've gotten varying results on training multiple models with same parameters.

Noise %	Train [Conv]	Eval [Conv]	Train [Ours]	Eval [Ours]
00	89	79	85	79
05	90	76	78	82
10	89	72	71	78
20	85	67	68	82
25	87	61	63	81

As we can see from the above results, Convolutional Networks keep fitting the noisy dataset with high accuracy, but when tested on a noise free eval set, the accuracy keeps getting lower with increasing noise. However, we see that the results are contrasting for our approach. Our model doesn't fit to the noise in the dataset, which is shown in the decreasing training set accuracy. But the eval set accuracy stays more or less constant with increasing noise.

6 Ongoing and Future Work

In [9], the authors have discussed an interesting method to tackle the noise in the labels of a dataset. The spirit of their method is similar to curriculum learning which was also one of the motivations behind our approach. They observe that a convolutional network model first learns the noise-free data points after which it starts over-fitting to the noisy samples. They have used this observation to build their own weighted loss function which they call bootstrapping loss function. In their loss function, they fit a bi-modal Beta Mixture of Means distribution to the values of loss function to obtain a bootstrapping style of loss function updates. This does not ignore the noisy labeled points but on the contrary, uses them to train a robust model. This gives us a hope that by using their loss function instead of the weighted BCE Loss, we can further improve our results. However, I feel that their method to obtain weights just on the basis of loss values might be noisy. Therefore, I'm trying to combine our approach of getting weights from the embeddings with their bootstrapping loss function to obtain better results.

References

[1] Domain-adversarial neural networks to address the appearance variability of histopathology images

[2] Unsupervised Domain Adaptation with Similarity Learning

[3] Conditional Generative Adversarial Network for Structured Domain Adaptation

[4] Transferable Curriculum for Weakly-Supervised Domain Adaptation

[5] Curriculum Learning - Yoshua Bengio et. al.

[6] Breast Cancer Histopathological Database (BreakHis)

[7] Domain Generalization with Adversarial Feature Learning - Li et. al.

[8] Adversarial Autoencoder - Goodfellow et. al.

[9] Unsupervised Label Noise Modeling and Loss Correction - Arazo et. al.