Medical Image Classification

Junjie Liao 301355940

1. Background

In the age of big data, massive data has influence on medicine. Analyzing plenty of data, doctor can take the analytical result as a reasonable reference. Specifically, in order to tell if a patient has pneumonia given his or her chest X-ray image, a doctor can use classifier to predict if the patient has pneumonia.

2. Problem Statement

The problem is to build a pneumonia classifier using massive chest X-ray images. Given a chest X-ray image, the classifier output the probability that the patient has pneumonia. This is a binary classification problem. The overall goal is to classify chest X-ray images as accurate as possible. However, since the use case is medicine, the result is unbalanced. That is, the accuracy of predicting pneumonia images should be pretty high, while the accuracy of predicting normal images does not matter as pneumonia images. That is, the true positive rate is more important the false positive rate.

3. Dataset

The dataset is Chest X-Ray Images (Pneumonia) [1] from Kaggle. This dataset has 5863 images and 2 categories, one of which is normal and another of which is pneumonia.

4. Methodology

4.1. Preprocessing

The images from the dataset have various large resolutions, which are not good for further processing. Thus, I cut the middle square part of each image and shrink it into a 64×64 resolution. Shrinking images normally leads to the aliasing of images, so I use cubic interpolation when shrinking images to preserve the details as much as possible.

4.2. Spatial Transformer

Another problem of the dataset is that the contents in these images are not strictly aligned. For example, some images are left skewed while some images are right skewed. The skewness is one of the noises of the dataset, so one of the tasks is to rotate these skewed images so that there are all aligned. Aligned images are better input for the classifier.


However, it is difficult to transform these images by hand when preprocessing images. One natural idea is that how I can automatically preprocess these skewed images. Spatial transformer is able to achieve this goal easily. Spatial transformer is a kind of neural network, whose input is an image and output is a matrix that represents a affine transformation. By applying the generated transformation, all images should be roughly aligned. This step of preprocessing is done while building the classifier rather before building the classifier.

4.3. Convolutional Neural Network

Convolutional Neural Network (CNN) is the most powerful image classifier, and it has much higher performance than other classifiers such as SVM. The output of convolutional layers is called feature map, which is the input of successive fully connected layers. The final output is the score of each class, and in this project the output is the probability of pneumonia

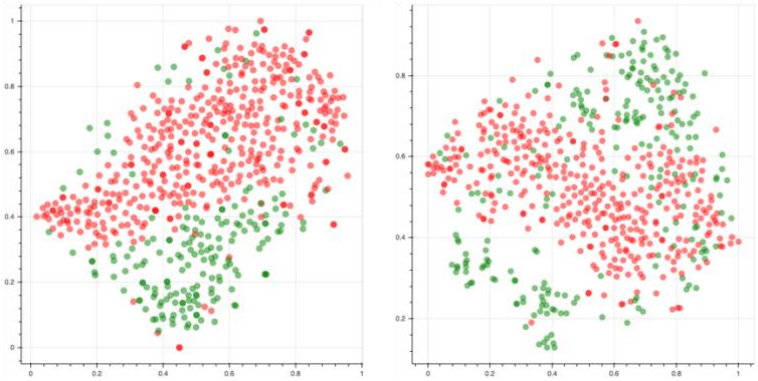
4.4. Pipeline

To sum up, the method mainly has three parts. The first part is simple image preprocessing. The purpose of this step is to make the image valid for later steps. Then, the second part is adding a spatial transformer before the CNN. The purpose of this step is to clean the noises in these images. The third part is training a CNN. The goal of this step to get an accurate classifier with high true positive rate.

5. Evaluation

The performance is reasonable. In terms of training set, false positive rate is 98%; the true positive rate is 99%. In terms of testing set, false positive rate is 46%; the true positive rate is 99%. These final result matches the problem requirements. The reason that the performance if good is CNN. The core idea of CNN is that learning to extract features automatically. This is achieved by adding convolutional layers, which learn the best feature extractors. Besides, the spatial transformer improves the performance.

In terms of testing set, even the true positive rate is pretty high, the model seems fails in false positive cases. In order to explore the reason why the model has much higher false positive rate on training set than testing set, I try to visualize the dataset. Specifically, I first reduce the dimension of images using t-SNE algorithm, I project the images onto 2-dimension space. Then, I plot these projected images on a plane. I find out that the distribution of training set is different from the testing set. The left is the training set distribution and the right is the testing set distribution. The green points represent normal cases and the red points represents pneumonia cases. The training set is almost linearly separable, while the testing set is not. The decision boundary for testing set is more likely to be an ellipse. In a word, due to the difference in distribution, the classifier does not have the same performance while training.



6. **Results**

The final outcome is that I build a CNN classifier that can predict pneumonia, and the classifier has pretty good performance on true positive cases.

In this project, I learned to manipulate images using OpenCV; build and train neural networks using PyTorch; visualize high dimensional data using dimensionality reduction algorithms and Bokeh.

7. **Summary**

Big data contributes a lot in medicine. A specific application scenario is pneumonia detection using CNN. I used preprocessed image to train a CNN with spatial transformer. The final CNN classifier has good performance, but it fails to diagnose the false positive cases on testing set, which has a different distribution from the training set.

8. **Additional Work**

To further make use of the model, I developed a simple web application using Flask framework. Users upload a chest X-ray image and send it to the server, and the sever is responsible for diagnosing and returning the diagnostic result.

9. **References**

[1] https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

[2] https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[4] https://docs.bokeh.org/en/latest/docs/gallery/color_scatter.html