# Convolutional Neural Network for Chest Image Classification

**Zhengyuan Xu**
UCSD
zhx024@ucsd.edu

**Jialiang Zhou**
UCSD
Jialiang Zhou

**Jinglong Du**
UCSD
jid020@ucsd.edu

## Abstract

Chest X-ray examinations are commonly seen in medical imaging diagnosis. A huge number of clinical diagnosis data of chest X-ray are accessible in online datasets. As the task of diagnosing chest X-ray is so commonly seen in a medical examination, it can be helpful to train a automated system that will classify chest x-ray images into certain thoracic diseases, thus alleviating the burdens of professional radiologists. This paper proposes a Convolutional Neural Network(CNN) that will detect and classify 14 kinds of thoracic diseases, trained using the ChestX-ray14 dataset. Our CNN model cannot be used as a diagnosis tool for medical imaging yet, as its performance is still inferior to state-of-art models. Instead, the training of our model is mainly for educational purpose, thus the performance of the model still need to be enhanced in the future.

## 1 Introduction

Tremendous progresses have been made recently in the field of computer vision problems thanks to deep learning and large-scale annotated image datasets [1]. The need for automated diagnosis system of radiology departments in hospitals is rising. A common computer vision task to be tackled with deep learning algorithm is to classify diseases when given a clinical imaging. This paper will talk about our CNN model that is consist of 4 convolution-pooling layers and 1 fully connected layer. This model, trained using ChestX-ray14 datasets, produces a multi-label vector when fed a clinical chest X-ray image. An element in the output vector would be 1 if the model decides the image has the corresponding disease.

## 2 Related Work

Many have been working on classifying clinical images by training deep CNNs [2,3]. Specifically for the ChestX-ray14 dataset, a DCNN is trained to classify and localize thoracic diseases [1]. Recently CheXNet produced the state-of-art accuracy trained using the ChestX-ray14 datasets [4]. Unlike the above systems, our model is only trained and used as a tool for us to learn and familiarize the process of machine learning application in the field of medical imaging diagnosis.

Table 1: Table describing the basic architecture of our CNN

| Layer | DESCRIPTION |
|---|---|
| Convolution1 | in channels=4,out channels=16,kernel size=5, stride=1 |
| Max Pooling1 | kernel size=2, stride=2 |
| Convolution2 | in channels=16,out channels=32,kernel size=5, stride=1 |
| Max Pooling2 | kernel size=2, stride=2 |
| Convolution3 | in channels=32,out channels=64,kernel size=5, stride=1 |
| Max Pooling3 | kernel size=2, stride=2 |
| Convolution4 | in channels=64,out channels=128,kernel size=5, stride=1 |
| Max Pooling4 | kernel size=2, stride=2 |
| Dropout | dropout rate=0.2 |
| fully connected | in channels=12*12*128, out channels=15 |
| activation | sigmoid |

# 3 Methods

## 3.1 Training data preprocessing

The images extracted from the dataset are 1024 * 1024 in resolution. We resized the data to 256 * 256 in resolution. No further method is used in preprocessing the data. Downscaling is needed because the original data size is too large and took too much computing power from the GPU. We could have used multiple convolution layer and pooling layer with a large stride to reduce the image size significantly, but downscaling is much faster.

## 3.2 Balancing data

The original data is unbalanced, with more than 60000 'Not Finding' labels, which makes the network learning much much more on 'Not Finding'. This results the net work know less on other diseases. Therefore, in order to keep the distribution of other diseases, and we don't want to make the network learn that much on 'Not Finding' labels, we undersampling the 'Not Finding' labels data. Meanwhile, we notice that there are two disease has a very low frequency, with only 300 and 1000, we upsample this two disease a little, to make the data more balanced

## 3.3 Training, validation and test sets

The validation set is obtained from randomly selecting one tenth of the data from the training set. Test data are the last 5000 images in the whole dataset excluding the training and validation data.

## 3.4 Architecture

Our model has 4 convolution layers, each followed by a max pooling layer. The kernel size of any convolution layer is 5. Batch normalization is used with eps=0.001. Before max pooling, there is a ReLU activation function. In such way, the input feature dimension is reduced by a factor of 2, by each dimension. Prior to the final fully connected layer, there is a dropout with a rate of 0.2. Dropout helps to reduce overfitting significantly [5]. The final fully-connected layer consists of 12 * 12 * 128 of units, and outputs a vector of 15 elements, including 14 diseases and 1 normal case label. A visualization of some of the layers are given in Fig 5, and a table describing the architecture is given in table 1.

## 3.5 Final layer activation

Instead of softmax, we used a sigmoid function before the final output of the CNN. A softmax function is often used in a multi-class classification network, and usually produces an output that is easily processed. A one-hot-encode label can be produced by finding the max value across the output. However, in this case, since each image is likely labeled more than 1 disease, it is not feasible

Table 2: Precision/recall for validation set

| Recall | Precision |
|---|---|
| 1.0 | 0.0810024348239 |
| 0.677272727273 | 0.27970207097 |
| 0.480058651026 | 0.583288793871 |
| 0.480058651026 | 0.583288793871 |
| 0.480058651026 | 0.583288793871 |
| 0.480058651026 | 0.583288793871 |
| 0.016568914956 | 0.54854368932 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |
| 0.0 | 0.0 |

to generate predictions using merely one-hot-encoding. In our model we used a sigmoid function at the end of the network. This turns the problem into a 15-way binary classification problem. Choices of threshold, which is used to produce the one-hot- like label, is talked later.

### 3.6 Loss function

Our choice of loss function is Binary Cross Entropy Loss(BCE Loss), whose value is given by:

$$loss(o,t) = -1/n \sum_i (t[i] * log(o[i]) + (1 - t[i]) * log(1 - o[i]))$$

where o is the output vector and t is the target label. BCE Loss is useful in binary classification problems, hence the choice.

### 3.7 Hyperparameters and Training

Our training process is operated by mini-batches of size 10. The learning rate is set to 0.0005 at the beginning, and decay over each epoch. We initialize the weights using Xavier initialization and set the bias to -0.1 at the beginning. The optimizer we used is Adam Optimizer, as told by in the write-up. However, the loss does not converge after 5 epochs of training.

## 4 Result

We found that the resulting performance of training with 3 epochs and training with 5 epochs are almost identical, and 3 epochs take less time so our result are displayed with the model after 3 epochs of training. Fig 2 shows the training loss over time.

As can be seen from the graph, the training loss over iteration drops drastically at the very beginning of the training and continues to fluctuate until the end. The reason why it did not converge well is probably the imbalanced data and the loss function we picked. The training data is very imbalanced. To be more specific, some categories have a positive label count that outnumbered the rest. The loss, when being calculated, should be multiplied by a weight. But we did not have enough time to implement our customized loss function.

The way we measure the performance is to calculate the precision/recall curve. We sweep through the threshold value from 0 to 1 and plot various precision/recall value. The results are presented in Fig 2 and Fig 3.

The result for the validation set has similar behaviors. But we present them in a table(Table 2) rather than a graph.

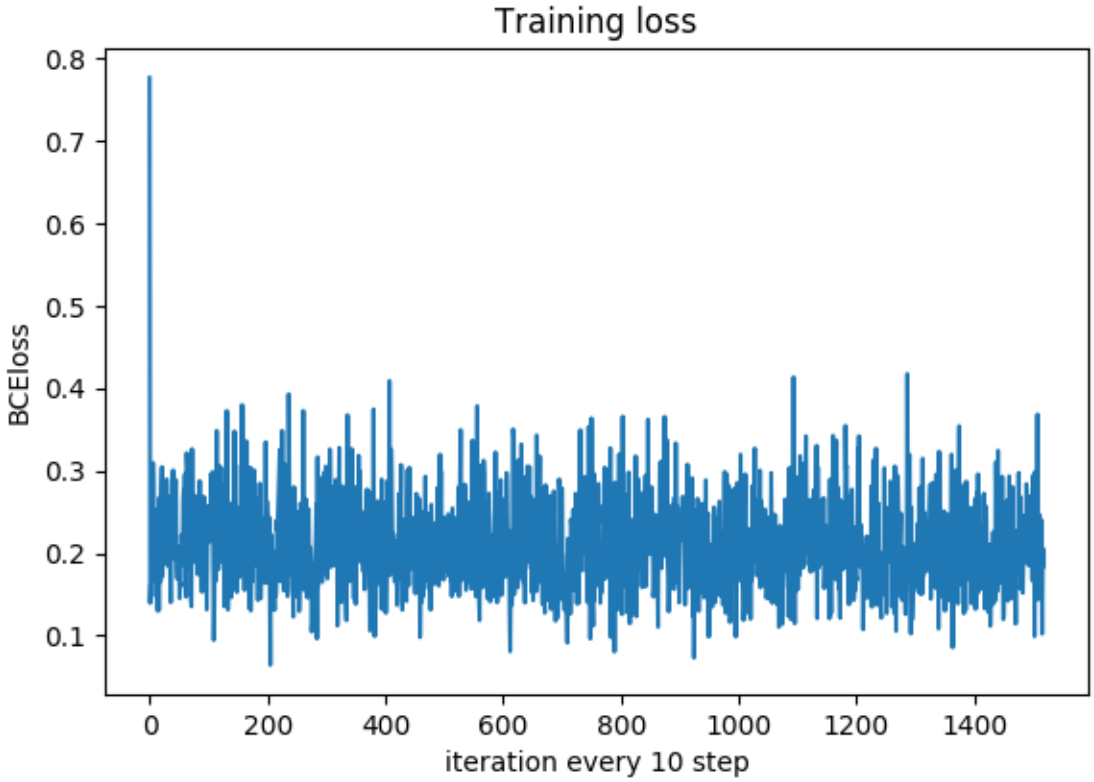


Figure 1: Training loss over time.

We noticed that we got similar plots for all three datasets (train, validation and test). This situation can be attributed to the divergent losses plotted in Fig 1. For the 11 thresholds we used for the final binary output label for the 15 classes, we found that 0.2 is the one with both precision and recall rate being around 0.5.

## 5 Conclusion

Chest X-ray imaging is one of the most commonly seen medical diagnosis. It has great significance for improving the diagnosis quality and deliver service to patients and doctors. By learning how to construct a CNN that is used to classify different kinds of thoracic diseases, we understand how to build from scratch and fine-tune the hyperparameters to better the performance of our model.

The performance of the algorithm we built did not reach anywhere near the state-of-art CNNs that are used to classify and localize diseases. But by building it we learned how to plan the architecture of a CNN and how to fine-tune the network so that its performance could be increased.

## 6 Transfer Learning

In this part, we apply a pre-trained model from pytorch library, we applied the model resnet152, only retrieve and train its fully connected layer on the same dataset with same batch size. We tried BCELoss as the lose function. We did the same thing as in part 1 to balance the data set, but this

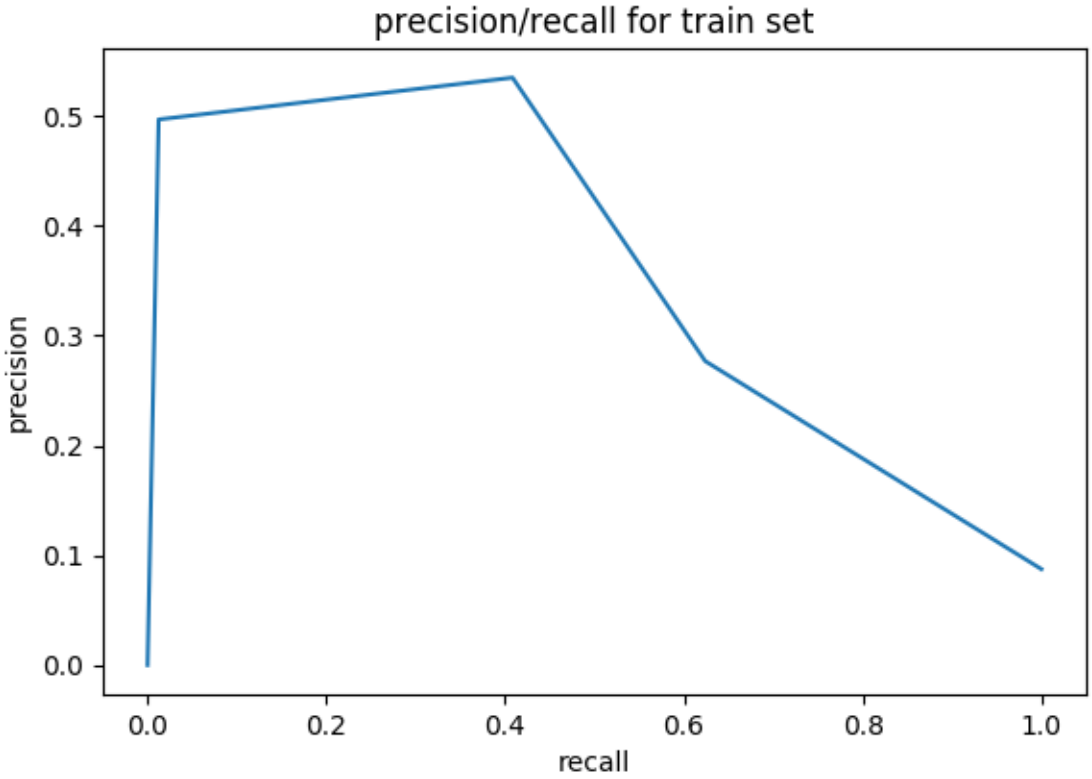


Figure 2: Precision recall curve for the training set. We set the threshold for output from 0 to 1.0. For some of the threshold, the recall and precision turn to 0.

time, we use upsampling instead of under-sampling the "Not Finding" samples. The loss tends to converge but still jumps sometimes. Generally, it performs better than our model. See fig. for result.

**Acknowledgments**

We would like to acknowledge the UCSD ETS (former ACMS) for the GPU used in the learning process.

**References**

[1] W. Xiaosong, P. Yifan, L. Le, L. Zhiyong, B. Mohammadhadi and S. M. Ronald. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases, National Institutes of Health, Bethesda, MD 20892

[2] Gulshan, Varun, Peng, Lily, Coram, Marc, Stumpe, Martin C, Wu, Derek, Narayanaswamy, Arunachalam, Venugopalan, Subhashini, Widner, Kasumi, Madams, Tom, Cuadros, Jorge, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. Jama, 316(22):24022410, 2016.

[3] Rajpurkar, Pranav, Hannun, Awni Y, Haghpanahi, Masoumeh, Bourn, Codie, and Ng, Andrew Y. Cardiologist-level arrhythmia detection with convolutional neural networks. arXiv preprint arXiv:1707.01836, 2017.

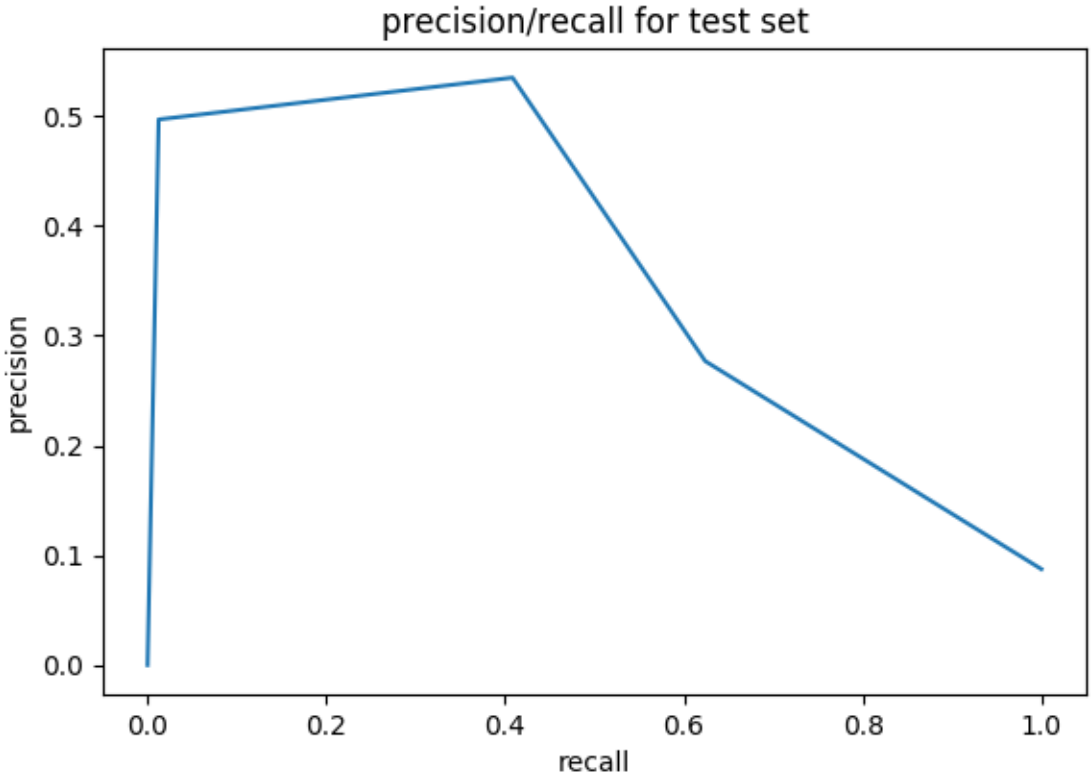


Figure 3: Precision recall curve for the training set. We set the threshold for output from 0 to 1.0. For some of the threshold, the recall and precision turn to 0. The best performance is reached when the threshold is set to 0.2.

[4] Rajpirka, Pranav, Irvin, Jeremy, Zhu, Kaylie, Yang, Brandon, Mehta, Hershel, Duan, Tony, Ding, Daisy, Bagul, Aarti, Langlotz, Curtis, Shpanskaya, Katie, Lungre, P. Matthew, Ng, Andrew Y. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. arXiv: 1711.05225v1

[5] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, Salakhutdinov, Ruslan, Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014) 1929-1958

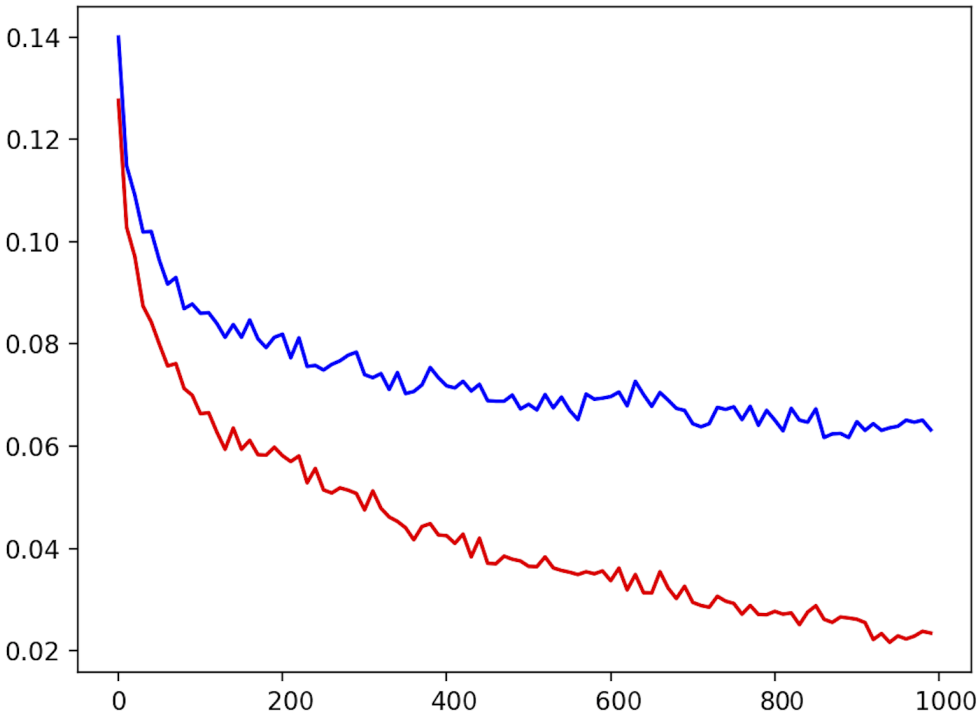


Figure 4: Training loss over time.

(a) conv1

(b) conv2

(c) conv3

(d) conv4

Figure 5: Visualization of our convolution layers. The images are of same dimensions with their corresponding convolution layers. As a result, the first one has a low resolution because it has a lower weight dimension than the others. All visualizations are rescaled to the same size for formatting purposes. Their actual dimensions are the same as their corresponding weights