Exploration of different Deep Convolutional Network for Thorax Disease Detection

Guanghao Chen	Qimin Chen	Yunfan Chen	Ke Xiao
A53276390	A53284263	A53287711	A53283874
guc001@eng.ucsd.edu	qic003@ucsd.edu	yuc014@ucsd.edu	k1xiao@ucsd.edu

Abstract

In this report, we implemented convolutional neural networks based on Pytorch and applied it to solve a task for detecting disease with X-ray image. We proposed 5 models in this report and compare the performance of them. Firstly, we proposed a baseline model which only includes simple convolutional layer and the BRC for this baseline model is 2.63%. Based on this model, we addressed the imbalanced issue by weighting the loss function and proposed an improved Baseline model whose BRC is 22.26%. In addition, we constructed two innovative model called Modified Model Version1 and Version2. The first model's BRC performance is 8.06% which actually doesn't converge and the second model's performance is 29.03%. Finally, we attempted to conduct transfer learning and we utilized Resnet as our prototype model, which leads to BRC of 33.30%. We compared our 5 models and analyzed them by performance of different metrics and visualization of filters.

1 Introduction

Deep convolutional neural networks have been applied to a broad range of problems and tasks within Computer Vision. In this report, we proposed to use convolutional neural networks(CNN) to solve a medical community disease detection problem. The specific disease detection problem we attempted to solve is to predict thoracic disease presented by chest X-rays images. The dataset we used is ChestX-ray. This dataset contains 112,120 images (frontal-view X-rays) from 30,805 unique patients, where each image may be labeled with a single disease or multiple diseases. To be specific, each X-ray images was labeled with positive or negative for each kind of diseases and there are 14 diseases included in this dataset. Therefore, this task can be abstracted as a multi-label multi-classification problem.

Although some of the previous works attempted to explore the relationship between images and disease diagnosis, there are still some reasons that this problem is hard to solve: 1. Training data can not be collected and labeled by crowd-sourcing platform such as AMT. The only way is to obtain from radiological reports with natural languages, which leads to extremely unbalanced problem for the dataset. 2. X-ray images are usually large but the pathological region always extremely small compared to the whole image. 3. Though there are already lots of pre-trained model but most of them are related to object recognition task, they can not be transferred to apply to the medical image diagnosis domain.

Xavier Initialization To tackle the above issues, we proposed a deep network architecture and trained based on ChestX-ray to complete the multi-label multi-classification task. Although commonly the performance will be better when network becomes deeper, it is also significant about the initialization of weights. Basically, CNN is still an optimization problem of loss function so the start point is rather important to the result. If the weights in a network start too small, then the signal shrinks as it passes through each layer until its too tiny to be useful. If the weights in a network start

too large, then the signal grows as it passes through each layer until its too massive to be useful. Therefore, one way to initialize the weights is using Xavier initialization. Xavier initialization sets the weights by drawing them from a typically Gaussian or uniform distribution with 0 mean and a specific variance,

$$\text{Var}(w) = \frac{1}{n_{in}} \tag{1}$$

where n_{in} is the number of neurons feeding into it.

Batch Normalization Even so, the weights transferring in the network also might be exploded because there are several activation layers inside the network. After activation, the distribution of the weights will not be the same one as input. In order to constrain it as the original distribution, Batch Normalization was been proposed. During training time, a batch normalization layer does the following:

1. Calculate the mean and variance of the layers input.

$$\begin{aligned} \mu_B &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma_B^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i - {\mu_B}^2) \end{aligned} \tag{2}$$

2. Normalize the layer inputs using the previously calculated batch statistics.

$$\bar{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3}$$

3. Scale and shift in order to obtain the output of the layer.

$$y_i = \gamma \bar{x_i} + \beta \tag{4}$$

where γ and β are learned during training state along with the original parameters of the network.

2 Related Works

There have been recent efforts on diagnosing with X-ray image based on computer vision approaches[1, 2]. Alexander[2] applied CNN directly to complete diabetic prediction based on eye-related images. Rakhlin[1] used VGG-16 architecture and concatenated the features from different convolutional layers to tackle a breast cancer analysis based on X-ray images. In this way, the input transferred to the fully connected network contains the features from each level of the convolutional layer. Besides, there are some architectures which lead to a series of breakthroughs for object recognition tasks. Deep networks naturally integrate low/mid/high level features and classifiers in an end-to-end multilayer fashion, and the levels of features can be enriched by the number of stacked layers (depth). Recent evidence reveals that network depth is of crucial importance such as VGGnet and Resnet[4, 5], and the leading results on the challenging. ImageNet dataset all exploit very deep models, with a depth of sixteen to thirty. He et al.[3] explored to use the feature map from different convolutional layers to improve the performance of pedestrian detection.

3 Methods

3.1 Baseline

As mentioned before, we need to solve a multi-label multi-classification problem. There are some existing architecture for classification. Therefore, we followed their architecture to construct a similar one. However, all of them are single label classification and so the activation function for the output is softmax, which is

$$y_n = \frac{exp(a_k^n)}{\sum_{k'} exp(a_{k'}^n)} \tag{5}$$

However, our task basically need to judge every class is positive or negative. For example, there are 14 diseases in this task. What we need to answer is "For this patient, he or she might catch disease 1 and disease 9." Then the prediction result of this patient can be represented with a multi-hot encoding "1 0 0 0 0 0 0 0 1 0 0 0 0". Therefore, in order to eliminate the inter-class's impact, we used **sigmoid function** as the output activation layer, which is

$$\text{sigmoid} = \frac{1}{1 + exp^{-z}} \tag{6}$$

Therefore, the baseline architecture we proposed shows in Figure 2.

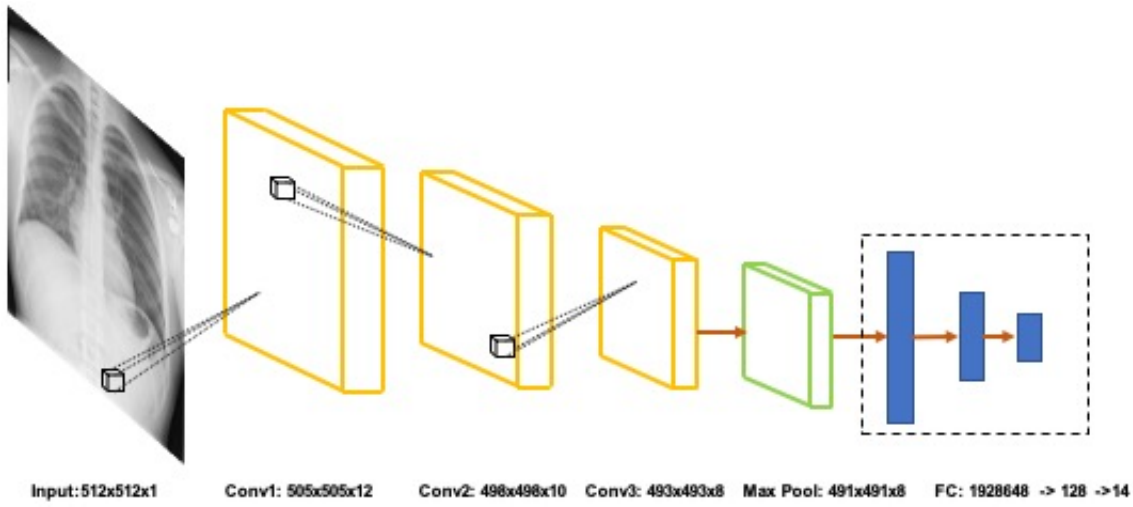

Figure 1: Architecture of Baseline Model. The yellow box represents convolutional layer and the green box represents the pooling layer.

From the figure above, we can find that there are 3 convolutional layers inside the architecture. After that, it was followed by a max pooling layer with 3×3 kernel size.

The loss function for baseline model is defined as

$$\text{Loss Function}(Y, P) = \sum_{i=1}^{N} \sum_{j=1}^{C} y_j^n log(p_j^i) + (1 - y_j^n) log(1 - p_j^i) \tag{7}$$

3.2 Improved Baseline Model

The baseline model is too naive to complete this complicated task. Therefore, we explored the reason of why this model is unsatisfactory which is the imbalanced dataset. Therefore, we improved it in 4 aspects.

Normalization In baseline model, we input the original grey scale image into the network directly and in this model we will convert it into an appropriate range at first. **Data Augmentation** Except normalizing the data, in order to generalize our model, we want to conduct data augmentation. To be specific, we will apply flip operation or crop operation to our original image. **Data imbalanced** The main reason about why the baseline model is bad is its loss function treat equally with positive samples and negative samples. Even though it make sense in most cases, the positive samples are too few to learn enough knowledge so we need to weight it more than negative samples. The loss function for baseline model is defined as

$$\text{Weighted Loss Function}(Y, P) = \sum_{i=1}^{N} \sum_{j=1}^{C} w_j y_j^n log(p_j^i) + (1 - y_j^n) log(1 - p_j^i) \tag{8}$$

Dropout In order to avoid our model is exploded because of the parameter arises, we utilized dropout technique. It will randomly shut down part of the neurons which removes the connection between the neighbouring nodes.

Therefore, the baseline architecture we proposed shows in Figure 2.

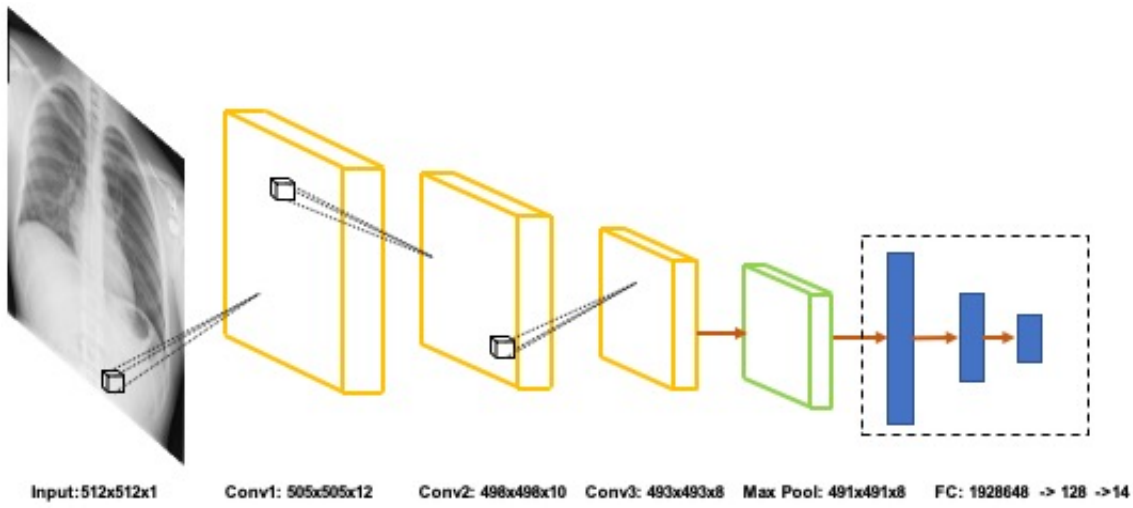

Figure 2: Architecture of Improved Baseline Model. The yellow box represents convolutional layer and the green box represents the pooling layer.

3.3 Modified Model Version1

As we discussed above, the pathological region is relatively smaller than the whole x-ray image. Besides inspired by [1], our first modified model is to extract the features from different convolutional layers and then concatenate them into a whole vector which should be the input of fully connected layer. The architecture of the network shows in Figure 3.

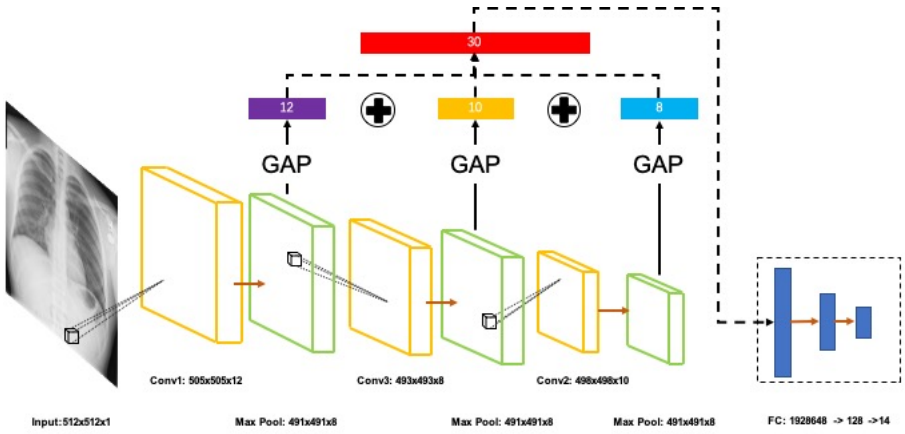

Figure 3: Architecture of Modified Model Version1. The yellow box represents convolutional layer and the green box represents the pooling layer.

Table 1: Configure of Modified Model Version1

Layer	Input Channel	Output Channel	Kernel
Conv1	1	64	3
ReLu	64	64	
Max Pooling1	64	64	2
Conv2	64	128	3
ReLu	128	128	
Max Pooling2	128	128	2
Conv3	128	256	3
ReLu	256	256	
Max Pooling3	256	256	2
GAP	64,128,256	448	
FC1	448	128	
ReLu	128	128	
FC2	128	14	
Activation Function	sigmoid		

As mentioned before, we noticed that the dataset is extremely imbalanced, which means the positive samples are few. This leads to the updating algorithm will learn more with negative samples and learn less with positive samples. Therefore, the output model will be uninformed about what should be positive. Therefore, the first method to deal with this issue is to weight more with the positive samples, which means to make positive ones to be more important. Further, the loss function will be represented as Equation (9), where w_j is the positive samples' weight for class j.

$$\text{Weighted Loss Function}(Y, P) = \sum_{i=1}^{N} \sum_{j=1}^{C} w_j y_j^n log(p_j^i) + (1 - y_j^n) log(1 - p_j^i) \tag{9}$$

The value of w_j is obtained by computed the multiple of numbers of negative samples over positive samples. Specific values of weights are shown in Table 2.

Table 2: Configure of Modified Model Version1

Disease	1	2	3	4	5	6	7
Weight Value	8.74	38.82	7.38	4.64	18.45	16.84	79.24
Disease	8	9	10	11	12	13	14
Weight Value	20.07	22.94	47.74	43.74	64.95	32.03	479.52

3.4 Modified Model Version2

From modified model version1, the input feature vector keeps only 448 dimension of feature vector while it is not sufficient to classify such a complicated problem. Therefore, we proposed a second modified version. In the Modified Model Version2, we directly reshape the feature maps into a vector from different pooling layers and concatenate them into one. The architecture diagram shows in Figure 4 and the dimension information shows in Table 3.

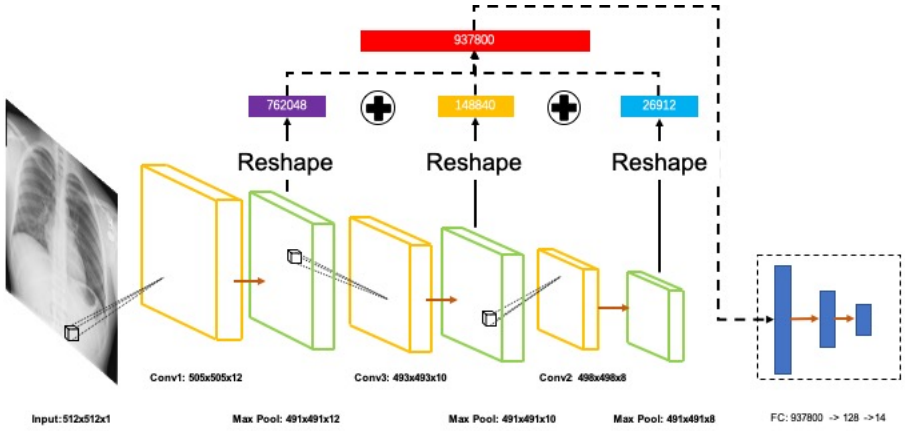

Figure 4: Architecture of Modified Model Version2. The yellow box represents convolutional layer and the green box represents the pooling layer.

Table 3: Configure of Modified Model Version2

Layer	Input Channel	Output Channel	Kernel
Conv1	1	12	8
ReLu	12	12	
Max Pooling1	12	12	2
Conv2	12	10	8
ReLu	10	10	
Max Pooling2	10	10	2
Conv3	10	8	6
ReLu	8	8	
Max Pooling3	8	8	2
Reshape	762048,148840,26912	937800	
FC1	937800	128	
ReLu	128	128	
FC2	128	14	
Activation Function	sigmoid		

3.5 Transfer Learning

There are two experiments conducted in transfer learning. The first one is to use ResNet18 pre-trained on Imagenet only (feature extraction): freeze all the layers and remove the last layer then stack last layer and train it on dataset. The second one is to use ResNet18 pre-trained on Imagenet only and fine-tune on dataset.

Table 4 shows the architecture of ResNet18 for this assignment.

Table 4: Architecture of ResNet18

layer name	input size	output size
conv1	512×512	256×256
conv2_x	256×256	128×128
conv3_x	128×128	64×64
conv4_x	64×64	32×32
conv5_x	32×32	16×16
average pool, 14-d fc	16×16	1×14

4 Results

Before discuss the results of models, it is necessary to state the metric of performance for this task. The common metric used before is **Accuracy**, which is

$$\text{Accuracy} = \frac{\text{total correct predictions}}{\text{total number of samples}} \tag{10}$$

However, in this task, the prediction can not reflect the performance precisely. Since the negative samples are more than the positive ones, a smarter decision is to classify all the queries with negative labels. Even though the model is naive, the accuracy can still be considerably high. Therefore, it is necessary to utilize other metrics. Two of them are called **Precision** and **Recall**, which is

$$\text{Precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|} \tag{11}$$

$$\text{Recall} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|} \tag{12}$$

Precision can reflect the ability of model to guarantee their positive predictions to be dependable and **Recall** can reflect the ability to find all the potential positive test samples.

Except for Precision and Recall, we can use **Balanced classification rate(BCR)** to measure both Precision and Recall, which is

$$\text{BCR} = \frac{\text{Precision + Recall}}{2} \tag{13}$$

4.1 Primitive Baseline Model

In this subsection, we will show a primitive version of baseline model to set as a comparison. For this model, we trained it from scratch. And the hyper parameters for this model shows in Table 5. Train loss and validation loss for **Baseline Model** shows in Figure 5. From the figure, we can notice that the model will be best after 2 epochs.

Table 5: Hyper Parameters for Primitive Baseline Model

Hyper Parameter	Value
batch size	8
learning rate	0.001
epochs	2

Table 6: Performance of Baseline Model

Disease	Accuracy	Precision	Recall	BCR
1	0.8965	0.0000	0.0000	0.0000
2	0.9748	0.0000	0.0000	0.0000
3	0.8801	0.5000	0.0351	0.2676
4	0.8250	0.2000	0.0003	0.1001
5	0.9475	0.0000	0.0000	0.0000
6	0.9431	0.0000	0.0000	0.0000
7	0.9876	0.0000	0.0000	0.0000
8	0.9516	0.0000	0.0000	0.0000
9	0.9586	0.0000	0.0000	0.0000
10	0.9797	0.0000	0.0000	0.0000
11	0.9771	0.0000	0.0000	0.0000
12	0.9854	0.0000	0.0000	0.0000
13	0.9706	0.0000	0.0000	0.0000
14	0.9976	0.0000	0.0000	0.0000
Aggregate	0.9482	0.0500	0.0025	0.0263

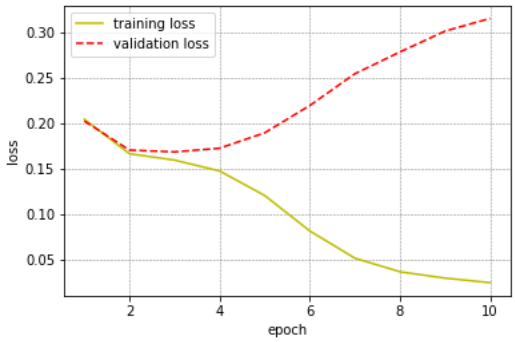

Figure 5: Training loss and validation loss of Baseline Model. The red dotted line represents validation loss and yellow line represents training loss.

Performance of Baseline Model shows in Table 6. From the table, we can find even though the accuracy looks good but the performance of remain metrics are terrible because we didn't do any additional operations such as normalization or adding weights to the loss function.

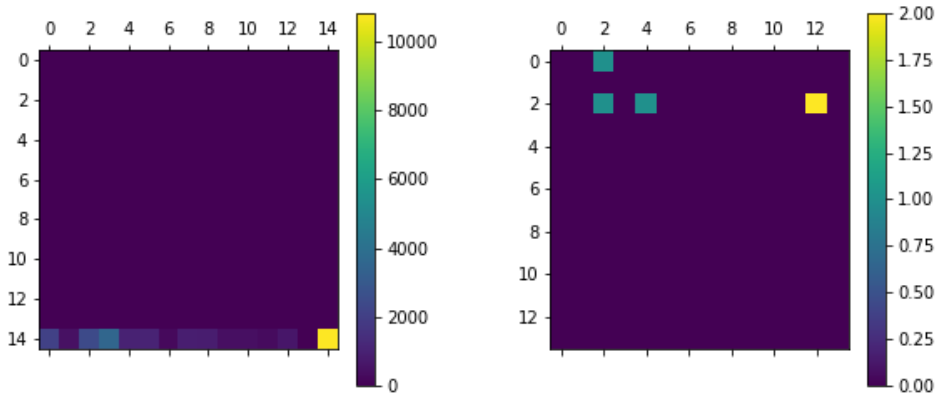

Figure 6: Left figure shows the confusion matrix of 15×15 (with no disease present) and Right figure shows the confusion matrix of 14×14.

4.2 Improved Baseline Model

In order to promote the primitive baseline model, we successively addressed the issues stated above.

Normalization The input image is a gray scale image and so the value range is from 0 to 255. In class, we discussed that it is better to project the input data to 0 mean which enables the input have positive and negative values. Therefore, we preprocessed the data with z-score operation, which is

$$\text{z-score} = \frac{x - \text{mean}}{\text{std}(x)} \tag{14}$$

Data Augmentation Even though the dataset is large, it is always useful to enlarge the dataset, which can generalize the model more. The operation to add more data to the original one is called Data Augmentation. There are several ways to complete augmentation and in Pytorch there is a function called **transformation** which provides us a convenient interface to achieve this aim. The data augmentation operations we adopted include

- horizontally flip the image
- crop a region of the image

Data imbalanced In subsection 4.2, we have already proposed one way to address the imbalance issue which is to use the weighted loss. Except that, we also re-sampled the data for a given training batch, which can make the distribution to be even. For example, when the batch size is 8, there are only 2 positive samples and then we randomly pick 2 negative samples and abandoned the rest negative ones. In this case, for each case, the model can learn evenly with positive cases and negative samples.

Dropout In case the model is begin trained overfitted, we introduced dropout technique to tackle it. Basically, it will randomly shut down part of the nodes in the network. In our model, we experimented with several dropout probability and finally determined to set it as 0.5. Figure shows the train and validation loss curve for this improved baseline model. Table shows the performance of this model. From the model, we can find that the improved baseline model enhance significantly compared with primitive baseline model. Especially, we can find both of the precision and recall value has been improved drastically, which illustrates that the improved model has been gradually learned evenly with positive and negative samples.

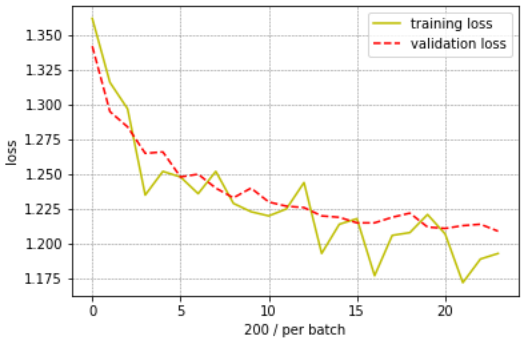

Figure 7: Training loss and validation loss of Improved Baseline Model. The red dotted line represents validation loss and yellow line represents training loss.

Table 7: Performance of Improved Baseline Model

Disease	Accuracy	Precision	Recall	BCR
1	0.7954	0.2132	0.3607	0.2869
2	0.8083	0.0644	0.5155	0.2899
3	0.7812	0.2580	0.4821	0.3701
4	0.7760	0.3135	0.2382	0.2759
5	0.8865	0.1277	0.2229	0.1753
6	0.9366	0.9366	0.0120	0.0556
7	0.9430	0.0219	0.0894	0.0557
8	0.8560	0.1185	0.3080	0.2132
9	0.8081	0.0942	0.4543	0.2743
10	0.8241	0.0740	0.6507	0.3624
11	0.8684	0.0595	0.3311	0.1953
12	0.8556	0.0290	0.2872	0.1581
13	0.8661	0.0664	0.2845	0.1754
14	0.8658	0.0052	0.4516	0.2284
Aggregate	0.8479	0.1702	0.3349	0.2226

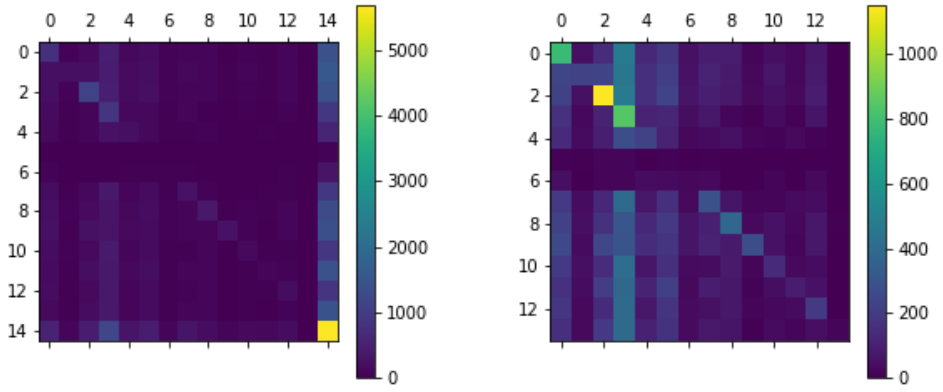

Figure 8: Left figure shows the confusion matrix of 15×15 (with no disease present) and Right figure shows the confusion matrix of 14×14.

4.3 Modified Model Version1

For this model, we still applied z-score normalization, data augmentation, dropout and weighted loss function to this model. In this subsection, we will explain how we extracted the features from different convolutional layers and how we concatenated them. In the front part of our model, there are 3 convolutional layers which respectively are 64, 128 and 256 channels. For each layer, we applied Global Average Pooling to it. To be specific, **Global Average Pooling (GAP)** will compute the average value of each channel. Therefore, for these three layers, GAP will return a vector with dimension of 64, 128 and 256. Further, we can concatenate these three vectors into one vector with dimension of $64 + 128 + 256 = 448$. Followed by that, the 448 dimension vector was transferred into the fully connected neural network. The training loss curve and validation loss curve was shown in Figure 9. The performance shows in Table 8.

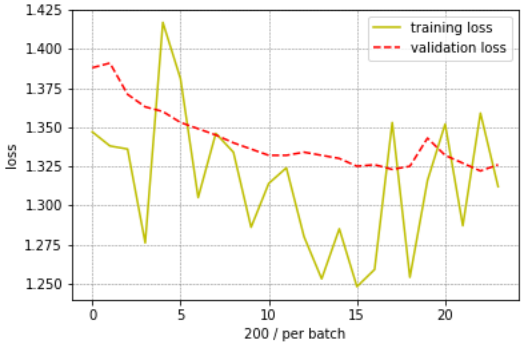

Figure 9: Training loss and validation loss of Modified Version1. The red dotted line represents validation loss and yellow line represents training loss.

Table 8: Performance of Modified Model Version1

Disease	Accuracy	Precision	Recall	BCR
1	0.8777	0.1971	0.0652	0.1311
2	0.9322	0.0372	0.0764	0.0568
3	0.8109	0.2058	0.2114	0.2086
4	0.8072	0.2869	0.0494	0.1681
5	0.9502	0.0000	0.0000	0.0000
6	0.9429	0.0000	0.0000	0.0000
7	0.9813	0.0141	0.0083	0.0112
8	0.9537	0.0000	0.0000	0.0000
9	0.8287	0.0849	0.3206	0.2027
10	0.7933	0.0522	0.5542	0.3032
11	0.9795	0.0000	0.0000	0.0000
12	0.8850	0.0243	0.0000	0.0000
13	0.9686	0.0400	0.0016	0.1034
14	0.9585	0.0050	0.0833	0.0208
Aggregate	0.9050	0.0677	0.0979	0.0861

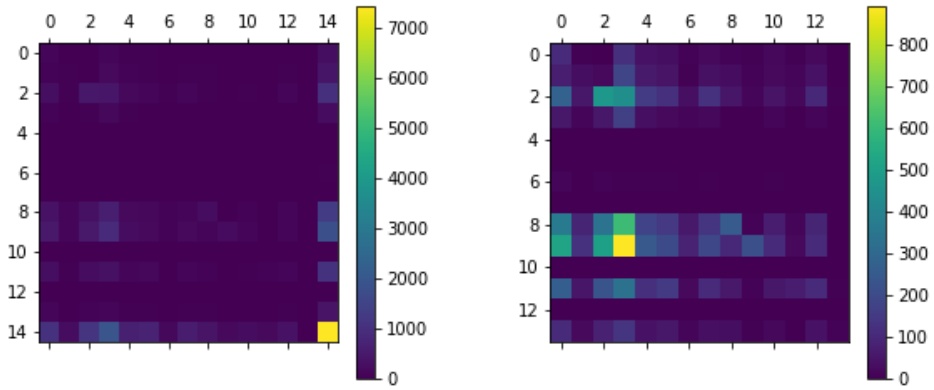

Figure 10: Left figure shows the confusion matrix of 15×15 (with no disease present) and Right figure shows the confusion matrix of 14×14.

4.4 Modified Model Version2

As mentioned before, instead of using Global Average Pooling, we directly reshape the feature maps into a vector. In our model, there are 3 convolutional layers. Each dimension of the reshaped vector is 762048, 148840 and 26912. Therefore, the merged feature vector is 937800. The training and validation loss of Modified Model Version2 shows in Figure 11. And the performance of Modified Model Version2 is shown in Table 9.

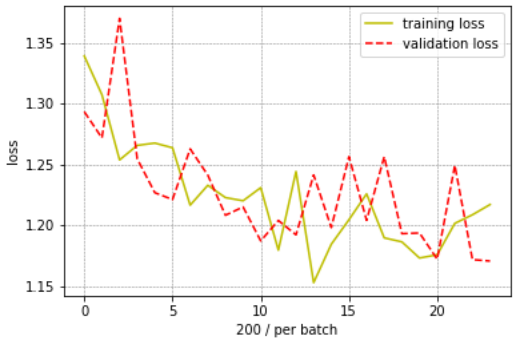

Figure 11: Training loss and validation loss of Modified Version2. The red dotted line represents validation loss and yellow line represents training loss.

Table 9: Performance of Modified Model Version2

Disease	Accuracy	Precision	Recall	BCR
1	0.8173	0.3115	0.2495	0.2805
2	0.8102	0.0562	0.6631	0.3597
3	0.7772	0.2520	0.6497	0.4509
4	0.7960	0.2343	0.7542	0.4943
5	0.9164	0.1251	0.5386	0.3319
6	0.9429	0.0000	0.4876	0.2438
7	0.9602	0.0702	0.4912	0.2807
8	0.9138	0.1039	0.6259	0.3649
9	0.8192	0.1209	0.3266	0.2238
10	0.8107	0.0765	0.5491	0.3128
11	0.9139	0.0000	0.1159	0.0580
12	0.8791	0.0243	0.2105	0.1174
13	0.9255	0.0400	0.2766	0.1583
14	0.9971	0.0050	0.7713	0.3882
All	0.8771	0.1014	0.4793	0.2903

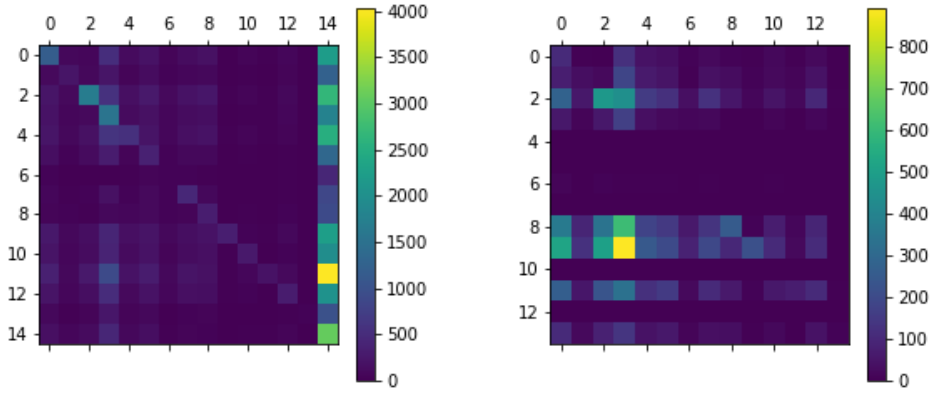

Figure 12: Left figure shows the confusion matrix of 15×15 (with no disease present) and Right figure shows the confusion matrix of 14×14.

4.5 Experiment with Ensembling

In this subsection, we combine our two best models and the baseline model together as a detector. Therefore, we need to define the rules for judgement. Since there are 3 models and consequently there will be 3 results. The rule for voting defines as

$$\text{rule} = \begin{cases} 1 & \text{where x} \geq 2 \\ 0 & \text{where x} \leq 2 \end{cases} \tag{15}$$

Based on this rule, we re-test our model on the test set. Table 10 shows the performance of this ensembling model. Compared with Table 7 and Table 9, we can find this ensembling model performs better than the previous models.

Table 10: Performance of Ensembling model

Disease	Accuracy	Precision	Recall	BCR
1	0.6283	0.1055	0.3435	0.2245
2	0.6435	0.0250	0.3563	0.1906
3	0.5824	0.1151	0.3829	0.2490
4	0.6684	0.1825	0.2474	0.2150
5	0.7833	0.0487	0.1700	0.1093
6	0.9412	0.0608	0.0085	0.0347
7	0.9242	0.0166	0.0887	0.0527
8	0.7119	0.0494	0.2959	0.1727
9	0.6685	0.0433	0.3187	0.1810
10	0.6671	0.0232	0.3469	0.1851
11	0.7231	0.0227	0.2693	0.1460
12	0.7323	0.0157	0.3007	0.1582
13	0.7221	0.0341	0.2898	0.1619
14	0.7326	0.0011	0.2000	0.1006
Aggregate	0.7235	0.0531	0.2585	0.1558

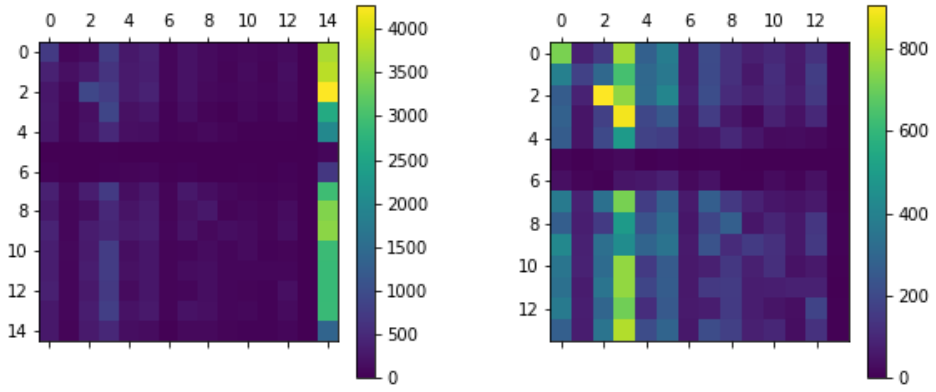

Figure 13: Left figure shows the confusion matrix of 15×15 (with no disease present) and Right figure shows the confusion matrix of 14×14.

4.6 Transfer learning

We use the same hyper parameters set and data augmentation as improved baseline model and two self-defined models for both feature extraction and fine tune. Both these two methods are trained with 2 epochs and the training loss and validation loss are saved every 200 batches with 32 batch size.

4.6.1 Freeze frontier

Figure 14 shows the train loss and validation loss. The loss is much more fluctuant than baseline because the ResNet18 is way complicated than baseline.

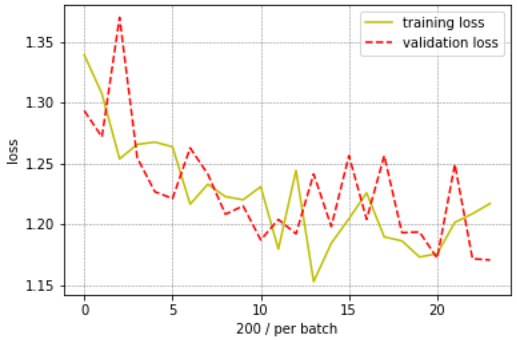

Figure 14: train loss and validation loss for feature extraction of ResNet18

Table 11 shows the performance of ResNet18. It can be seen from table 11 that ResNet18 with feature extract outperformed the baseline. The accuracy drops down while precision and recall are no longer zeors any more which means the model does not always predict 0 compared to baseline.

Table 11: Performance of ResNet18 feature extract

Disease	Accuracy	Precision	Recall	BCR
1	0.8753	0.3115	0.1286	0.2200
2	0.7099	0.0592	0.6858	0.3725
3	0.7275	0.2520	0.6660	0.4590
4	0.5375	0.2343	0.7152	0.4747
5	0.7350	0.1051	0.5318	0.3184
6	0.7103	0.0934	0.4679	0.2806
7	0.7949	0.0302	0.4902	0.2602
8	0.7662	0.1256	0.6752	0.4004
9	0.8786	0.1209	0.2928	0.2069
10	0.7192	0.0666	0.8971	0.4818
11	0.7892	0.0632	0.6044	0.3338
12	0.8307	0.0453	0.5049	0.2751
13	0.8872	0.0739	0.2665	0.1702
14	0.7379	0.0071	0.8085	0.4078
Aggregate	0.7642	0.1135	0.5525	0.3330

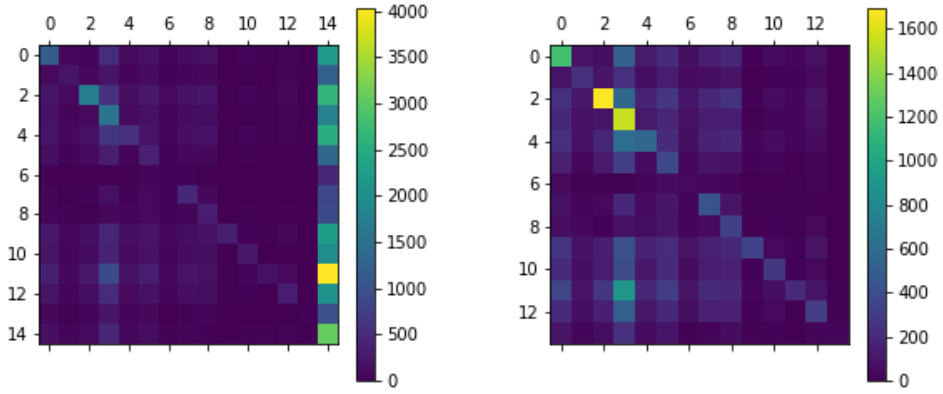

Figure 15: Left figure shows the confusion matrix of 15×15 (with no disease present) and Right figure shows the confusion matrix of 14×14.

4.6.2 Train Whole Network

Figure 16 shows the train loss and validation loss. Not freezing the weights of previous layers enables the network to learn much more informations of Thorax disease data. Also the loss of fine tune is less volatile than feature extraction.

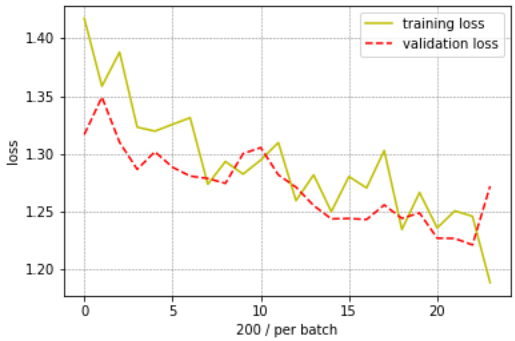

Figure 16: train loss and validation loss for fine tune of ResNet18

Table 12 shows the performance of ResNet18. It can be seen from table 12 that ResNet18 with fine tune outperformed the baseline and somehow performed a little better than feature extraction because not freezing previous layers weights enables the network to learn much more features of dataset and therefore more robust in predicting diseases.

Table 12: Performance of ResNet18 fine tune

Disease	Accuracy	Precision	Recall	BCR
1	0.8121	0.1987	0.2649	0.2318
2	0.7897	0.0690	0.7043	0.3867
3	0.8086	0.2805	0.3642	0.3224
4	0.6374	0.2674	0.7323	0.4999
5	0.7265	0.1276	0.5273	0.3275
6	0.7116	0.1133	0.4875	0.3004
7	0.8772	0.1417	0.3989	0.2703
8	0.8268	0.1010	0.6097	0.3553
9	0.8610	0.0706	0.4645	0.2675
10	0.7086	0.0761	0.8541	0.4651
11	0.7931	0.0865	0.6148	0.3507
12	0.7110	0.1226	0.4471	0.2848
13	0.8653	0.1221	0.2874	0.2048
14	0.7455	0.2248	0.6412	0.4330
Aggregate	0.7767	0.1430	0.5284	0.3357

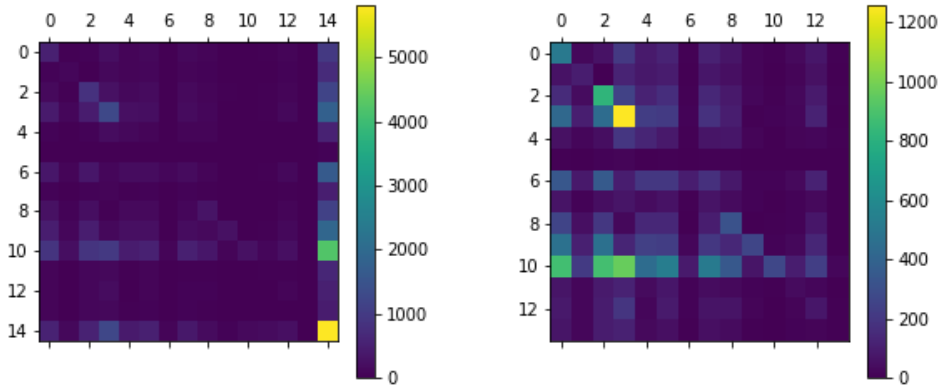

Figure 17: Left figure shows the confusion matrix of 15×15 (with no disease present) and Right figure shows the confusion matrix of 14×14.

4.7 Visualization of the learned filters

Figure 18 shows the visualization of the learned filters of modified networks version 1. These feature maps of filters are extracted from layer 1, layer 2 and layer 3, respectively.

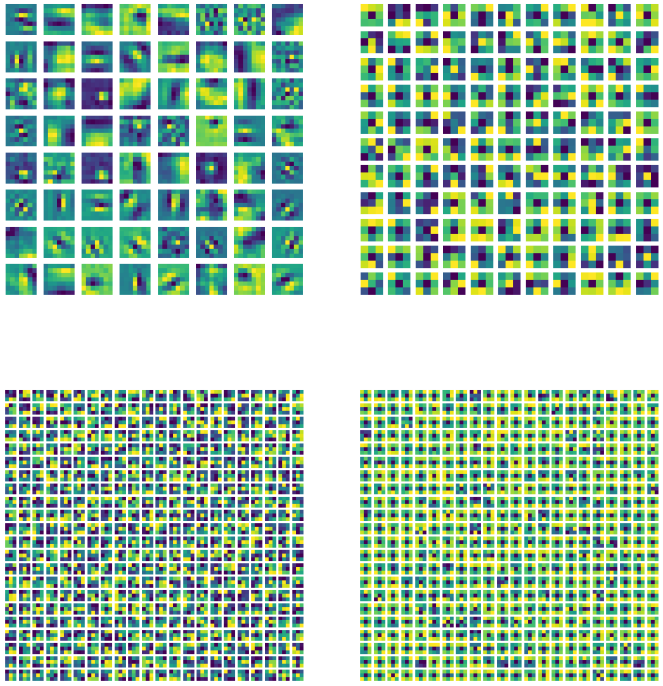

Figure 18: **Left Top**: layer 1. **Right Top**: layer 2. **Bottom**: layer 3.

5 Discussion

In this section, we will discuss the results of the 5 models covered before, including baseline model, improved baseline model, Modified Model Version1, Modified Model Version2 and transfer learning model. Besides, we will analyze the reason behind the result.

5.1 Different in performance

Among our above 5 models, we compared them using accuracy, precision, recall and BRC. Besides, we utilized confusion matrix to compare them also. Finally, we found that the model using transfer learning outperforms the other 4 models. The reason we think is the network is hard to go converge and so if we utilize the pre-trained weight, it will perform better than train the network from scratch.

5.2 Common Confusions

Since the positive samples are too few, the confusion matrix of the original shape (15×15) shows no difference among the cells except [14,14]. Therefore, we show another confusion matrix which abandoned the last column and last row. From all the confusion matrix, we can find that the cell[2,2], [3,3] and [4,4] reflects high value. It also can be confirmed by their high recalls(68%,66%,71%) in ResNet. However, at the same time, we found other disease also prefer to be classified as class 2,3,4, which can be observed from the bright area in the left bottom corner.

Except the baseline model, confusion matrices from other 4 models shows the similar result. And the reason Baseline Model is different mainly because of its bad performance.

5.3 Different metrics

Accuracy in all the models are relatively high, which was discussed before because of the imbalanced data. Therefore, we need to focus more on the other metrics. The precision can tell us the models' ability to judge correctly with their positive prediction. For example, the model might predict 1000 positives but only 100 are indeed disease. In this case, the precision is low. While recall can reflect the ability to discover all the positives. Take the same example as instance. Although the precision is low, but all the 100 positives are covered in the result. In this case, the recall value is high. For our models, most of them have relatively high recall value but with a low precision value. Therefore, it might be led by the high weight of positive samples for different classes.

5.4 Visualization of feature maps

As the above visualization of feature maps in different layers, the filers from early layers mainly focus on learning fundamental features such as points and lines. The filters from later layers mainly focus on learning features integrated from previous layers so that the informations are integrated and passed through the layers.

6 Authors' Contribution

Guanghao Chen In the project, Guanghao mainly focused on modified model version1 and modified version2. Besides, Guanghao programmed the evaluation codes including calculating the metrics and confusion matrix. For the report, Guanghao composed the asbtraction, introduction, related works, methods and results section for baseline model, improved baseline model and modified models.

Qimin Chen In the project, Qimin mainly focused on programming transfer learning model and running experiments. For the report, Qimin covered the methods description and result section of transfer learning as well as collating confusion matrix data and plots.

Yunfan Chen In the project, Yunfan focused on baseline model experiments and weighted loss coding. Also all the training and validation loss of 5 models are plotted by Yunfan. Yunfan also provided an extra local GPU for running experiments.

Ke Xiao In the project, Ke mainly focused on running experiments of modified network and coding visualization of weights. Ke also collated training and testing results of modified network and sent the results back to Guanghao.

References

[1] Rakhlin A , Shvets A , Iglovikov V , et al. Deep Convolutional Neural Networks for Breast Cancer Histology Image Analysis[J]. 2018.

[2] Diabetic Retinopathy detection through integration of Deep Learning classification framework

[3] He K , Zhang X , Ren S , et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 37(9):1904-1916.

[4] Simonyan K , Zisserman A . Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

[5] He K , Zhang X , Ren S , et al. Deep Residual Learning for Image Recognition[J]. 2015.