Programming Assignment 3 Report for CSE 190

Shengju Qian
U07736587

Linfeng Zhao
U07684312

Abstract

The multi-label classification for diseases detection is widely used for high large-scale high precision computer-aided diagnosis(CAD) system including X-ray images. Based on the large dataset, the deep convolutional neural network can be fully trained. We purpose a VGG-like deep CNN according to the properties of the X-ray images and apply it on ChestX-ray8 dataset. We also compare it with several pre-trained networks and choose the ResNet18 to fine-tune. Our experiments show that our network is competitive even comparing to the network from the paper purposing this dataset.

1 Introduction

Multi-label Classification with Convolutional Network: The tremendous progress has been evidenced in a variety of computer vision problem via deep neural networks with large-scale image datasets with annotations[9]. Deep convolutional neural networks are extremely powerful machines and have shown unbelieved ability on predicting and detection works, especially on computer vision work. Since the great progress AlexNet developed in the LSVRC-2010 ImageNet training set[9], deep neural networks began to show its magic on large datasets detection work.

Classification for Diseases Diagnoses: Though there are so many working going on 'everyday' image, the convolution neural networks(CNN) still have some limitation on medical images and diseases diagnoses, which requires more accurate localization. Followings are two main reasons that make CNN cannot work too well in this fields: the large scale of medical image datasets and difference between medical images and usual images.

In order to improve the accuracy of medical images and generalization on different diseases, two main kinds of methods have been proposed these years. One focuses on introducing new, large-scale datasets, the other is related to improving the power of Convolution Neural Networks, especially the ability to find the exact position of items in a picture, which is a much hard task.

To tackle the first issue, [16] proposed a new chest X-ray database, namely "ChestX-ray8", which comprises 108,948 frontal-view X-ray images of 32,717 (collected from the year of 1992 to 2015) unique patients with the text-mined 14 common disease labels, mined from the text radiological reports via NLP techniques. Also, the author demonstrated that some commonly occurred thoracic diseases can be detected and even spatially-located via a unified weakly-supervised multi-label image classification and disease localization formulation. Just like his work, we are going to use some deep learning techniques to show the power of CNN on diseases diagnosis, which is promising in building a large-scale computer-aided diagnosis (CAD) systems.

2 Related Work

There have been recent efforts on creating openly available annotated image databases. Particularly for chest X-rays, which is the mostly-common used radiology images in diseases diagnosing system.

Before the Chest X-ray 8 dataset, the largest public dataset is OpenI [1], which contain 3,955 radiology reports from the Indiana Network for Patient Care and 7,470 associated chest X-rays from PACS.

Also, boosting the power of convolution neural networks is always a big thing, lately, R CNN [5] has shown its power for accurate detection, then, a number of related research in this area have begun, such as Faster RCNN[11] and of course, the wonderful Mask RCNN[7]. All the works above are very great improvement in detection, the mostly important task in computer vision world. Also contribute a lot to our medical problems. However, in diseases diagnose system, it is different because it contains classification as well as detection tasks, it becomes much harder. Thanks to those work, there is also some weakly-supervised R CNN work going on finding the specific regions of specific diseases.

3 Methods

3.1 Data Preprocessing

3.1.1 Data Processing

In ChestX-ray8 dataset, the images are directly built from DICOM file and resized as bitmap images with original detail contents[16].

We use the downsampled 256 * 256 grayscale images as the input.

The "average" image: We construct the "average" image over all images to speed up the training. The image is shown in **Figure 1**. We use this image to normalize all the images in the dataset.

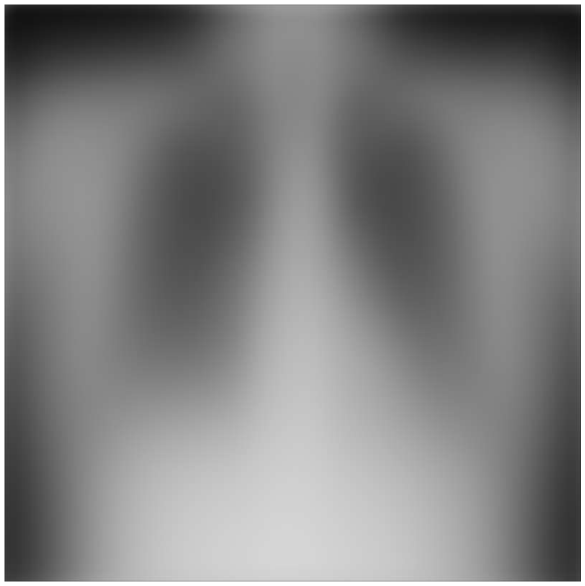


Figure 1: The "average" image over all images

3.1.2 Processing Unbalanced Data

Oversampling: Class imbalance is a challenging issue in practical classification problems for deep learning models[3]. Because the unbalanced labels in the dataset, we introduce oversampling before training. The statistics of every class of disease in the dataset is shown below:

```
'Atelectasis': 11535.0,
'Cardiomegaly': 2772.0,
'Consolidation': 4667.0,
'Edema': 2303.0,
'Effusion': 13307.0,
'Emphysema': 2516.0,
'Fibrosis': 1686.0,
'Hernia': 227.0,
'Infiltration': 19870.0,
'Mass': 5746.0,
'Nodule': 6323.0,
'Pleural Thickening': 3385.0,
'Pneumonia': 1353.0,
'Pneumothorax': 5298.0.
```

3.2 Deep Convolutional Network Architecture: Learning From Scratch

We build several models to test the performance of different architectures. Among these models, we will mainly introduce two models to compare the performance: a simple convolutional network and a VGG-like convolutional network[12]. Because the VGG network is too large for the server we use, we modify it based on the situation of our training platform and disease classification dataset.

To handle the unbalanced problem mentioned above, we use a weighted loss function in our models and is introduced below.

Weighted Loss Functions: In training our deep CNN on CheatXray8 dataset, we use weighted cross entropy loss function because of the extremely unbalanced contribution of categories. The definition of weighted-CEL (W-CEL) is shown as follows,

$$L_{W-CEL}(f(\mathbf{x}), \mathbf{y}) = \beta_P \sum_{y_c=1} -ln(f(x_c)) + \beta_N \sum_{y_c=0} -ln(1 - f(x_c)) \tag{1}$$

, where β_P is $\frac{|P|+|N|}{|P|}$ and β_N is set to $\frac{|P|+|N|}{|N|}$. $|P|$ and $|N|$ are the number of positive samples ('1's) and negative samples ('0's) in a batch of image labels.

Before using W-CEL as the loss function, we try several different loss function, such as original unweighted cross entropy loss. However, we find that because the unbalanced proportion of positive and negative instances, it is very difficult to learn positive samples.

At the following subsection, we will introduce two network architecture that we built.

3.2.1 Simple Convolutional Network

Our simple version multi-label CNN architecture is implemented using PyTorch. The architecture of this simple CNN just has two convolutional layers.

However, this simple network has poor performance on the ChestX-ray8 dataset. So we introduce another well-designed CNN architecture in the next part.

3.2.2 Deep CNN: VGG-like network

Another network we built is a VGG-like convolutional network[12].

Modified VGGNet: After trying some "naive" and simple convolution networks, which always give us all 0 outputs, we purpose a modified version of VGG network and make it learn from scratch. We have tried several "homemade" networks, only to get some disappointing results. Thanks to on-going researches these years, we decided to learn from those modules. After roaming about them with a flashlight, we find our best teacher VGG Net[12], whose structure is very useful for image recognition, extremely powerful at feature retrieval.

However, it is too big for us to accomplish our task and may cost a lot of disk storage to store those millions of parameters. Therefore, we try to build a network from scratch. With the inspiration of VGG Net, we attempt to build a "VGG-like" convolutional network. However, our model is quite different from VGG16, with fewer layers, fewer parameters and only 1 channel input and 14 classes one-hot encoding output. What's more, we delete some of the convolution layers and cut back 1/2 of the between fully connected layers nodes to 2048. We believe that our approach will make the computation cost and become more efficient than both VGG and AlexNet.

Structure of Purposed VGG-like network: This model is modified for the disease classification task and reduced to fit the server because the original model is too large. The architecture of the purposed VGG-like CNN is shown in the **Figure 3**.

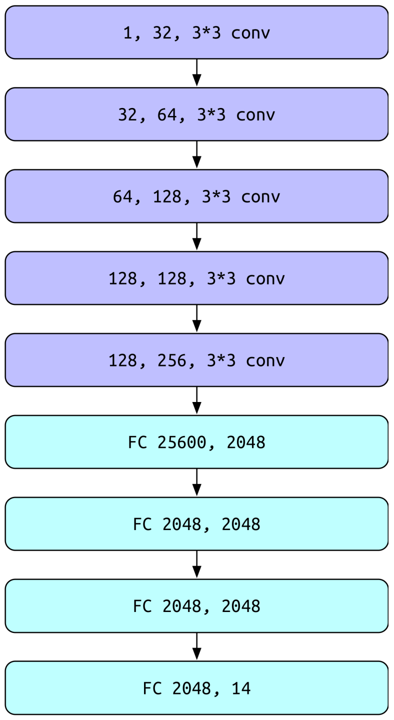


Figure 2: The architecture of VGG-like CNN

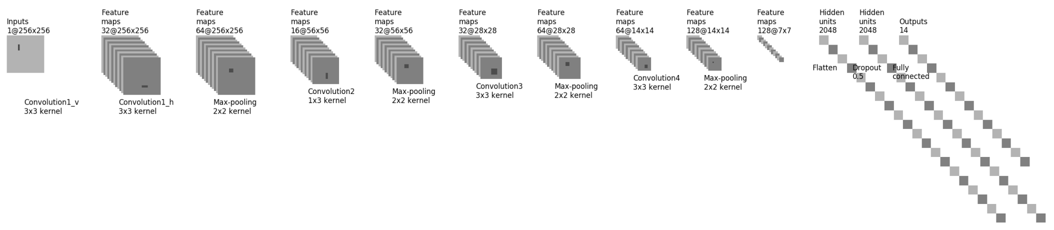


Figure 3: The sketch of VGG-like CNN

The VGG-like architecture has several advantages[12]. It it very simple. Among many choices, VGG adopts the simplest: only 3 * 3 convolution and 2 * 2 pooling are used in the whole network[15]. Also, the VGGNet-based network has good performance in classification task and widely used in some medical field[2].

However, one drawback of VGG network has one drawback: the network is usually too big[13]. So that is why we choose to optimize the architecture of the network.

3.3 Fine-tuning Pre-trained Model and Transfer Learning

Nowadays, many state-of-the-arts deep networks are released by famous research groups, i.e., Caffe Model Zoo and VGG Group. Thanks to the wonderful generalization abilities of pre-trained deep models, the pre-trained network can be easily used in the transfer learning with only the modification and fine-tuning of the final layer[10][17].

In transfer learning, we need to choose a suitable model with the same training data as we used (disease classification), or the transfer learning may not work well, and we can just train a final linear layer to fit the dataset. We tested several pre-trained convolutional network architectures.

Finally, we use ResNet18[8] to initialize the weights and train the final layer.

The ResNet-based (residual network) CNN is relatively small[8], so they can be put in our server (considering the fact that we just have 500 MB disk space for each person.) The other networks like the VGGNet-based CNN is too large to be fine-tuned in the server because they are too large.

Multi-label Setup and Loss Function: The representation of image-label we chose is to use a n-dimensional label vector $y = (y_1, ..., y_C), y_i \in 0, 1$, where y_i indicates the presence with respect to the image. This representation transits the multi-label classification problem into a regression-like one-hot encoding setting. We use the modified weighted cross entropy loss function introduced in the beginning of the section.

Comparing learning from scratch and fine-tuning We proposed two kind of architecture. The first one is VGG-like convolutional network for learning from scratch and another one is pre-trained ResNet18 to fine-tune (applying transfer learning). In fine-tuning ResNet18[8], only the final layer will be trained with our modified W-CEL loss. The proposed VGG-based network will learn from scratch.

4 Experiments and Results

In our experiment, we've tried several approaches, including simple two convolution layers networks, well-organized complex CNN, which really looks like CNN and fine-tuning on the pre-trained ResNet18 model. We mainly focused on our VGG-like model and transfer learning on fine-tuning ResNet18[8].

4.1 Datasets

We evaluate our simple convolutional network and VGG-like network on the ChestX-ray8 dataset. In ChestX-ray8 dataset, the images are directly built from DICOM file and resized as bitmap images with original detail contents[16]. In total, 108,948 frontal-view X-ray images are in the database, of which 24,636 images contain one or more pathologies. The remaining 84,312 images are normal cases.

For the multi-label classification task, we randomly shuffled the dataset and divided the entire dataset into two subset: training (80%), validation(10%) and testing(10%) set. We have used cross validation techniques, and the results below are all under those approaches. However, because of the limitation of time and resource, it seems impossible for us to execute the whole 10-fold process, which is still in progress. Nevertheless, our neural networks have showed its great performance after about 150 epochs.

Furthermore, due to the unbalanced labels in each class, we used the proposed data preprocessing steps stated in the last section to handle the unbalanced dataset.

Note that we didn't use the standard 10-fold validation since the limited time and limited resource make it impossible for our relatively complex network to be trained so many times.

4.2 Convolutional Network Framework Setting: Training the Model

Our CNN models are implemented using PyTorch. The server we use has one NVIDIA 1080Ti GPU with 12GB GPU memory.

We mainly focus on training the purposed VGG-like CNN. The results of simple CNN will be briefly described. Fine-tuning ResNet18 will be covered at the end of the section.

4.2.1 Loss Function

Note that we use weighted cross entropy loss function in trining to handle the unbalanced labels, or the positive prediction will be 0% if we use unweighted CEL.

The definition of W-CEL is in the last section.

4.2.2 Regularization

We do not apply regularization in the training.

4.2.3 Weights Initializations

We use Xavier initialization and modified version in training our models from scratch[6]. The formula that we use is shown as follow.

$$\sigma(w_i) = var(w_i) = \frac{2}{n_{in} + n_{out}} \tag{2}$$

, where $\sigma(w_i)$ is the variance of generated random numbers for initialization and n_{in} and n_{out} are the number of inputs and outputs.

4.2.4 Optimization

Adam Optimizer: The Adam optimizer is the currently best optimizer. It can be seen as the combination of momentum and learning rate decaying (RMSprop) method.

In Adam, there are several hyper-parameters. *We use the default values in practical:*$\beta_1 = 0.9$ *and* $\beta_2 = 0.999$.

Learning Rate and Batch Size: The learning rate should be decaying with the training. Another way is to increase the mini-batch size rather than decreasing the learning rate[14].

We adjust the learning rate manually in training VGG-like network. At the first 60 epochs, the learning rate is 0.001 and we reduce it to 0.0001 to make the network converge.

The training rate for fine-tuning ResNet18 is 0.0001.

Note that because the capacity of the GPU memory, we need to choose a suitable batch size in training. *The batch size is 32 or 64 because of the limitation of GPU memory.*

Batch Normalization: Training Deep Neural Networks is complicated because the distribution of each layer's inputs changes during training, as the parameters of the previous layers change[cite!!!!]. So by using backpropagation, the weights of deeper will be changed because the changing of the previous layers.

Batch Normalization is used to normalize the inputs of each hidden layer like the normalization of input data. It allows the training to use much higher learning rates and be less careful about initialization[4].

The formula is shown below.

$$\mathbf{z}^{(i)}_{norm} = \frac{\mathbf{z}^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{3}$$

$$\tilde{z}^{(i)} = \gamma \mathbf{z}^{(i)}_{norm} + \beta \tag{4}$$

, where σ is the sample variance of the inputs of each hidden layer, γ and β are the parameters to learn by backpropagation.

4.3 Results of Learning from Scratch and Fine-tuning

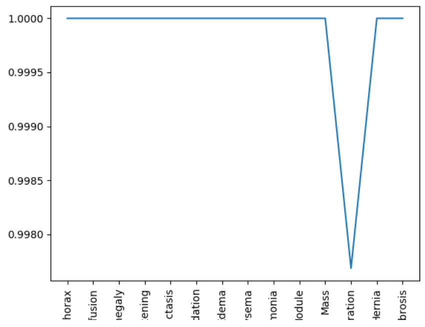
4.3.1 Building from Scratch

Simple CNN: The simple convolutional network does not have good performance. Comparing to our well-design VGG-like CNN, this simple CNN does not have a complex enough architecture to learn from the ChestX-ray8 dataset.

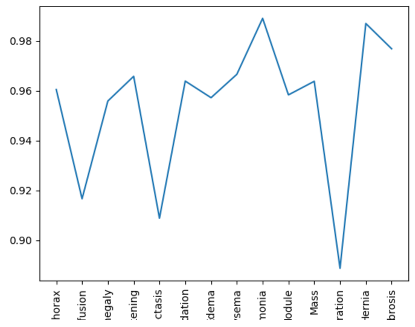
VGG-like CNN: In the first 60 epochs, we set learning rate to 0.001 with Adam solver. The loss roughly converges to about 0.20. Then we manually decrease it to 0.0001 and the loss converges to about 0.02.

Finally, the VGG-like CNN converges in about 120 epochs and loss converges to about 0.02.

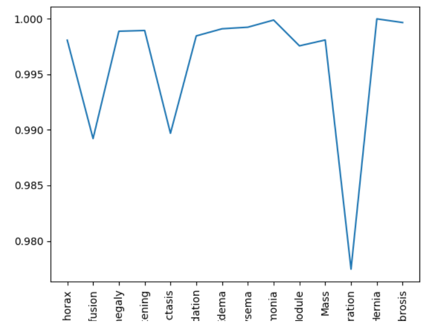
The analysis is in the next section. The results are shown below.



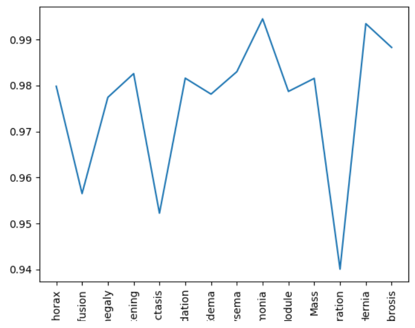
(a) Recall of VGG-like CNN



(b) Precision of VGG-like CNN



(c) Accuracy of VGG-like CNN



(d) F-score of VGG-like CNN

Figure 4: Results of VGG-like CNN

Fine-tuning ResNet18: We used Adam solver to fine-tune with the learning rate being 0.0001. In term of the loss function, we tried two ways: one is Multi-label binary cross entropy with weighted loss and without weighted loss.

1. ResNet18 transfer learning with weighted loss: doesn't converge after about 100 epochs, the loss is jumping up and down from 1.0 to 0.7. Finally, the loss is about 0.80.

2. ResNet18 transfer learning without weighted loss: loss goes down only a few batches, but doesn't converge after 100 epochs. Loss jumps from 0.1 to 0.3. Finally, the loss is about 0.20.

Results: The fine-tuned ResNet18 does have good result. The results are shown below.

True Positive Sample:

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

True Negative Sample:

```
[ 106822.   98813.  109348.  108735.  100585.  107453.  109817.  109604.
 110767.  105797.  106374.   92250.  111893.  110434.]
```

False Positive Sample:

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

False Negative Sample:

```
[  5298.  13307.   2772.   3385.  11535.   4667.   2303.   2516.   1353.
   6323.   5746.  19870.    227.   1686.]
```

Accuracy:

```
[ 0.95274706  0.88131466  0.97527649  0.96980913  0.89711916  0.95837496
  0.97945951  0.97755976  0.98793257  0.94360507  0.94875134  0.82277917
  0.99797538  0.98496254]
```

The analysis is in the next section.

4.4 Analysis of Learning from Scratch and Fine-tuning

Pre-trained Models: Due to the limitation of server (disk quota and GPU memory), the transfer learning models that we test did not perform well. The final one we choose is ResNet18[8] because of it small size, but it still do not perform well after training the final output layer.

The pre-trained network that we show is ResNet18, which is introduced in the last section. We train the final layer of ResNet18[8] with the ChestX-ray8 dataset.

Analysis of the Performance: Our VGG-like network shows amazing ability in those diseases detection, but we still need to run this training proceed in other data set.

However, even after about 100 epochs of training, the pre-trained ResNet still can't converge very well although its loss is not very high, which is just "tolerable"

Also, we tried to see how output is like, which turns out to be mostly '0's results. Those kinds of results lead to a satisfying accuracy, however, falling short in detecting any kinds of diseases.

Sometimes, The network may return some positive samples rarely, those samples are mostly right.

After analyzing the ResNet, we guess it is because of ResNet's characteristic and its original goal. This network is not directly fit the goal, instead, fitting the residual. As we know, it is much easier to optimize and find the match between residual than the original mapping. However, since we couldn't set relatively big batches, our two approaches (batch size = 32, batch size = 64) give the network a much smaller input. Due to the sensitivity of ResNet, it may be more "confused" as the small and biased input. For every batch, there won't be too many positive samples, leading to ResNet's misunderstanding of their "Residuals", which in other words, regarding those positive samples as "noise". As a result, in terms of gradient descent, the "blind" network will go to the totally wrong direction and end up at a very "deep" local minimum, making it feels satisfied with the loss.

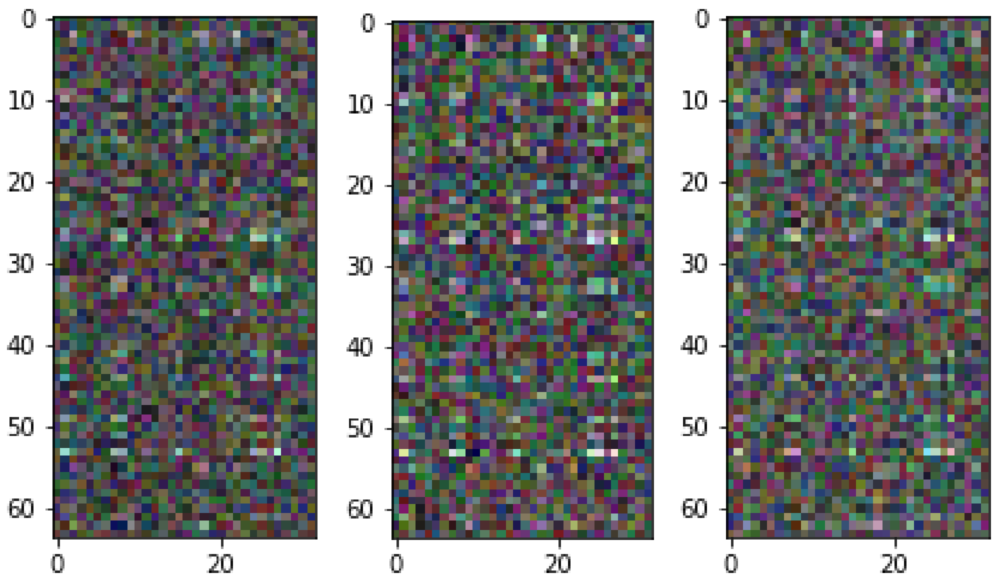
4.5 Visualization

Those are found convolutional layers of our well-organized CNN, each graph is made of a small (3*3) convolution kernel. We didn't include the first layer because our input is 1 channel, so the "convolution map" of the first convolutional layer is just a line made of (3*3) convolution kernel, which doesn't provide any useful and visible information to us. As is shown in every layer, at first two or three layers, the kernel map seems to be noisy while we can still see some white regions which can possibly be some disease region. As the layer goes up, most noise is reduced and we can get a much more clear map. Also, the useful information inside a map, including those white regions and some special shapes become very neat. Visualization of neural networks makes us understand each layer and how they work together better, provide insights into each layer and each kernel.

The images are shown below from **Figure 5 to Figure 8**.

5 Conclusion and Discussion

Conclusion: We attempt to build a general convolutional neural network architecture to classify X-ray images, especially for multi-label disease classification.



(a) conv2 0 (b) conv2 1 (c) conv2 2

Figure 5: Visualized Layer 1

In order to handle some problems such as unbalanced labels in the dataset, we apply several data preprocessing techniques before training. In our experiment, we've tried several approaches, including simple two convolution layers networks, well-organized complex CNN, which really looks like CNN and fine-tuning on the pre-trained ResNet18 model.

We mainly focused on our VGG-like model and transfer learning on fine-tuning ResNet18[8].

With the promising result of out deep convolutional neural network, we can conclude that our approach provides an efficient way of diseases detection. On the ChestX-ray8 dataset, our so-called "VGG-like" CNN shows great potential.

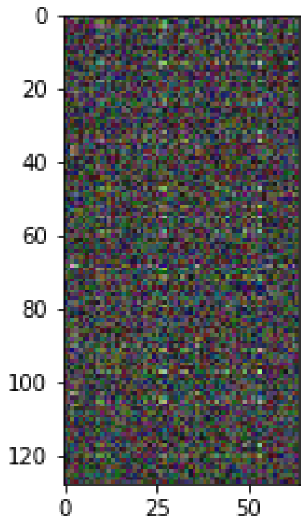
However, the transfer learning by fine-tuning ResNet18 has poor performance on the dataset due to some of its characteristics as we analyzed in the experiment. Also, it is shown in the paper purposing ChestX-ray8 dataset that transfer learning cannot achieve satisfactory result now. The reason may be the distinctiveness of the X-ray medical images in which the feature and information are represented in a totally different way than the "everyday" photos such as ImageNet.

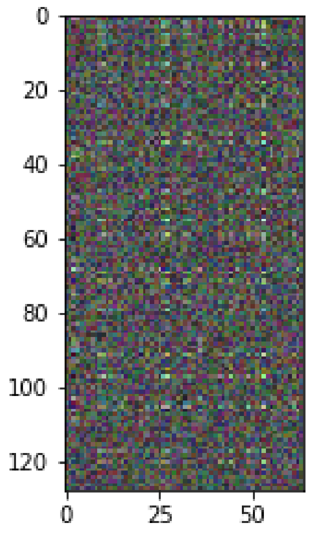
Feature Work: This dataset maybe can be extended to cover more disease classes and integrated with clinical information to make the diagnosis more accurate. Besides, finding the position of each disease can also be useful to boost the development of zero/less shot learning.

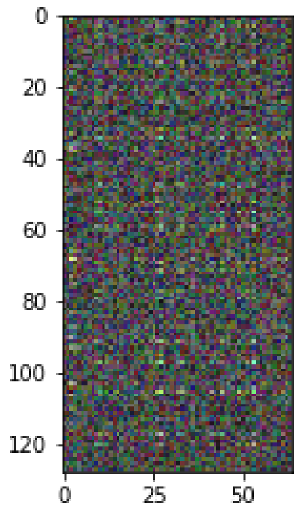
6 Personal Contributions

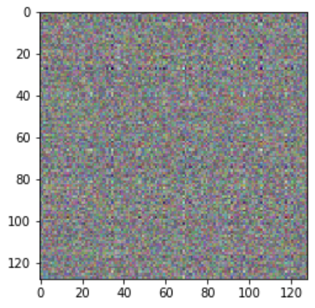
Shengju Qian: He mainly focused on the coding. In addition, he trained and fine-tuned the models. Also, he and Zhao analyzed the results together and plot the results.

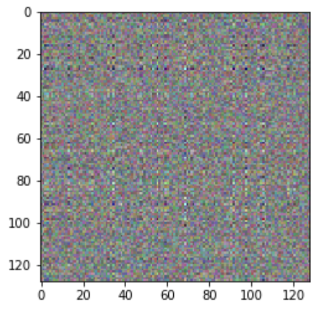
Linfeng Zhao: He was mainly in charge of the report and analysis. And he and Qian figured out the best architecture of CNN on the given dataset. Also, he tested part of the code and our models by my own.


(a) conv3 1 0


(b) conv3 1 1


(c) conv3 1 2


(d) conv3 2 0


(e) conv3 2 1

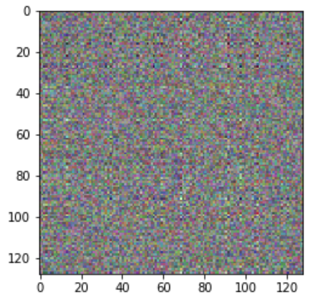
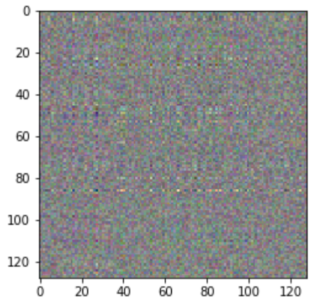
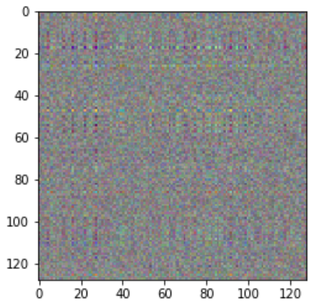

(f) conv3 2 2

Figure 6: Visualized Layer 2


(a) conv4 0


(b) conv4 1

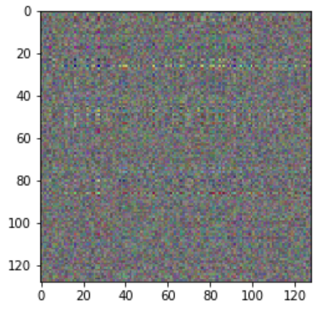
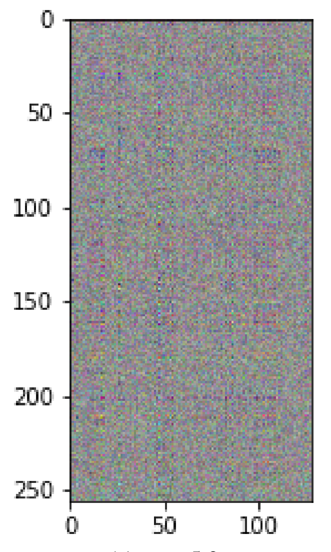

(c) conv4 2

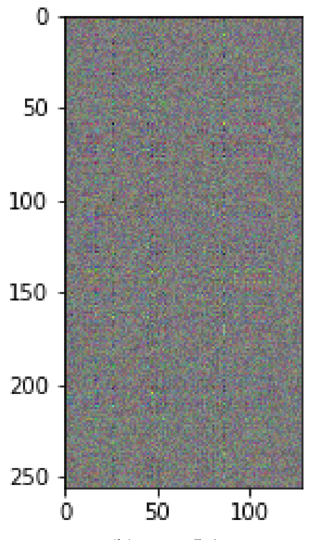
Figure 7: Visualized Layer 3

References

[1] Open i. `https://openi.nlm.nih.gov/`. Open-i: An open access biomedical search engine.

[2] *Very deep convolutional neural network based image classification using small training sample size*, 2015.


(a) conv5 0


(b) conv5 1

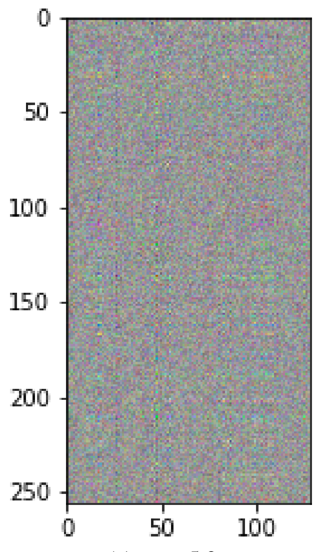

(c) conv5 2

Figure 8: Visualized Layer 4

[3] S. Ando and C.-Y. Huang. Deep Over-sampling Framework for Classifying Imbalanced Data. *arXiv.org*, page arXiv:1704.07515, Apr. 2017.

[4] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *arXiv.org*, July 2016.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *AISTATS*, 2010.

[7] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[8] K. He, X. Zhang, S. Ren, and J. S. 0001. Deep Residual Learning for Image Recognition. *CVPR*, pages 770–778, 2016.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[10] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7):436–444, May 2015.

[11] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[12] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, 1409:arXiv:1409.1556, 2014.

[13] V. K. Singh, S. Romani, J. Torrents-Barrena, F. Akram, N. Pandey, M. M. K. Sarker, A. Saleh, M. Arenas, M. Arquez, and D. Puig. Classification of Breast Cancer Molecular Subtypes from Their Micro-Texture in Mammograms Using a VGGNet-Based Convolutional Neural Network. *CCIA*, 2017.

[14] S. L. Smith, P.-J. Kindermans, and Q. V. Le. Don't Decay the Learning Rate, Increase the Batch Size. *arXiv.org*, page arXiv:1711.00489, Nov. 2017.

[15] L. Wang, S. Guo, W. Huang, and Y. Q. 0001. Places205-VGGNet Models for Scene Recognition. *CoRR*, 2015.

[16] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers. ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, pages 3462–3471. IEEE, 2017.

[17] P. Wieschollek and H. P. A. Lensch. Transfer Learning for Material Classification using Convolutional Networks. *arXiv.org*, Sept. 2016.