Diagnosing Pneumonia with Convolutional Neural Network

Huixuan Tan, Jianing Zhang

Abstract

In this paper, we show comparison between different existing convolutional neural networks and a CNN model we built on multiple dataset mainly surround the Chest X-Ray (Pneumonia) dataset. We aim to create a CNN model to better detect whether the patient has pneumonia or not. We compared our model with VGG-16, Densenet, resnet-50, and Alexnet. From the result, resnet has the best performance which it has high accuracy in classification both health lung and pneumonia. To the end, we explore and study multiple neural network architectures and how accuracy of the model is influenced by number of convolution layers, pooling layer, activation function, and also the number of classes within a dataset.

Introduction

Pneumonia is a persistent disease that is common in human society. Most people don't even know they have Pneumonia because its symptoms are pretty similar to having a cold. Because of limited and valuable resources in hospitals, it would be ideal if we can use machine to detect whether the patient has pneumonia. Classifying x-ray images is a tedious work, and during the process, there is probability that people could make mistakes. By this approach, we could save time for doctors and we could eliminate false positive and false negative rate. In this project we intend to build a unique neural network structure and compare with existing convolutional neural networks like VGG-16, Densenet-201, resnet-50, and Alexnet.

Figure 0

Many researches have been approaching to solve this problem, for example, Backpropagation Neural network used to diagnose chest disease(Rahub H., 2018), regression neural network used on diagnosis of chronic obstructive pulmonary and pneumonia disease. (O. Er, 2010). The above researches have successfully built architecture that could help classify medical issues. However we want to test out more

modern architectures that already proved to be efficient in dealing with images and see how their performance is on classifying data that are more obscure than classifying objects.

The dataset is from kaggle, it is chest x-ray images of normal people and pneumonia patients. There are basically three types of pneumonia. As shown in figure 1, the left one is a normal lung. The middle one is pneumonia caused by bacterial as we can see, there are focal lobar consolidation in the picture. The right one is viral pneumonia which shows a more 'diffuse 'interstitial' pattern in both lungs. The dataset is organized in three folders, which are training, testing and validation. There are in total 5863 x-ray images categorized into pneumonia and normal. As told in the description of the dataset, the data was collected 5 years ago in Guangzhou Women and Children's Medical Center. The dataset was graded three times to eliminate grading errors.
The reason we are using convolutional neural network is because of its convenience and variation to deal with images. It is widely used in computer vision such as image classification, object detection and neural style transfer. The concept of convolution is based on Fourier theory and linear system analysis. It is more feature learning than other neural networks. Its usage simulated the process of dealing with people in human brain which is hypothesized to be beneficial for classifying pneumonia since diagnonizing it will be more difficult than telling difference of objects.

Method/Architecture

Table 1: Top-1 and Top-5 error of selected models provided by PyTorch Documents

Model	Top-1 error	Top-5 error
VGG-16	28.41	9.62
AlexNet	43.45	20.91
Densenet-201	22.80	6.43
ResNet-50	23.85	7.13

1.Simple convolutional neural network
The convolutional neural network model we built consists of four convolutional blocks and one sequence of fully connected layers. Within each convolutional block, ReLU is used after every convolution layer to replace negative values by zero except that the first block consists of two consecutive convolution layers to get more informative data before pooling. After that, there is an average pooling layer followed by batch normalization. Within the fully connected layers, a dropout function applies to the data before linear transformation to reduce overfitting in the learning process. We apply a sigmoid function as the activation function to the final output data for the reason that there are only two classes in the pneumonia dataset we used.

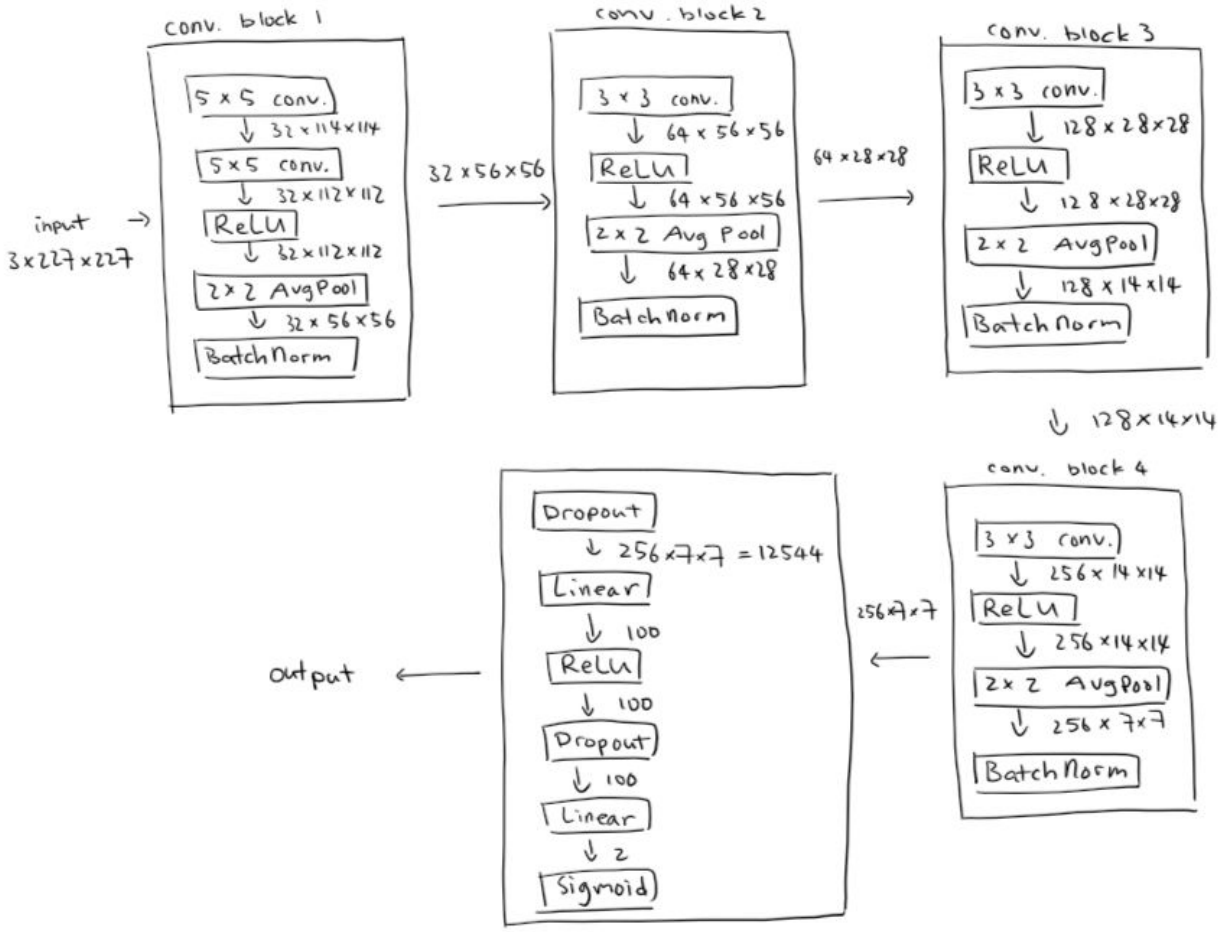

Figure 1. Architecture of our CNN model

2.VGG-16

The VGG-16 model is mainly composed by a large number of 3 * 3 convolution layer that captures the notion of direction and a 1-D convolution filter which acts as a linear transformation. It takes fixed-sized 224 * 224 RGB images as input. VGG-16 has 16 weight layers and a total of 134 millions parameters. Max pooling is used in the pooling step. (Simonyan et al., 2015)

3.Alexnet

The significant feature of Alexnet is data parallelism and model parallelism in convolutional layers and fully connected layers. In parallel with part of the workers working on forward computation, backward propagation is also operated by other workers. Workers sent their batch of computation result after convolution layers to other workers and repeat the computation process.(Krizhevsky, 2014)

4.Densenet-201

Densenet-201 has three dense blocks in addition to convolution and pooling layers. Layers are densely connected. It receives the feature-maps of all preceding layers as input and reserver information prevent them from loss between connection of different layers. It has a growth rate at k0 + k * (L - 1), where k0 is

the depth of input layer, L is the number of layers, and k is the depth of the current layer. At the end of the fully connected layer, output size is reduced to one-by-one. Average pool is used as spatial pooling.It is a logical extension of Resnet model but deeper and more accurate. It can reduce vanishing gradient and enhance feature propagation. (Huang et al., 2018)

5.resnet-50

Resnet-50 is a 50 layer Residual Network. The meaning of Residual is subtraction of feature learned from input of the layer. The way it is doing it is by connecting input of nth layer to the n+x layer. Usually without adjustment, other deep networks could suffer from vanishing gradients, but by connecting layers that are far away, this allows you to stack additional layers and skip through layers that are less relevant in training. The form of network can reduce degrading accuracy and is easier to train than other CNN networks. In the pooling steps of the network, the network used relu and it also used batch normalization. (He et al., 2015)

Datasets

To test out and consolidate the performance of the neural network model we deployed. In addition to the Chest X-Ray Images (Pneumonia) dataset (Mooney, 2018) which is introduced in the introduction section, we also used small datasets from Kaggle including Alien vs. Predator images (Migdał, 2018) which contains a total of 694 images of alien and predator, Hello_World_Deep_Learning_SIIM (Wiggins, 2019) which has chest and abdominal x-ray images in separate folders, Garbage classification (cchangcs, 2018) which contains six classes of garbage images including cardboard, glass, metal, paper, plastic, and trash, and Blood Cell Images (Mooney, 2018) which have four different cell types. These datasets are used for testing how accurate the convolution model we built if we implement our model on other datasets. In addition, testing multiple datasets aims to compare how the model work on different types of object and how the dataset have influence on this model.

Experiment

In the progress of building our convolutional neural network model, we tried to change the training function, transformation of the input image, types of spatial pooling layer, and activation function. We also did a comparison between a simple model with four consecutive convolution layers and another model with five consecutive convolution layers. These models for comparison purposes have ten epochs whereas the final result is produced by our best model with twenty epochs. During each comparison, there is only one variable.

1.Validation Set

Whether using validation set or not to modify hyperparameters during the training process give a different result. Cross validation is also useful for detecting overfitting. When training the above convolutional neural network model with validation set, the accuracy of the network on the test set is 97% where accuracy of the normal class is 52% and the pneumonia class is 96% while training with only the training set, the overall accuracy is 78% which of normal class is 52% and of pneumonia is 94%.

Table 2: Training result of training with validation set and without validation set

	Validation acc	Training acc	Testing acc	Normal	Pneumonia	False Positive	False Negative	Time
With Validation set	0.6875	0.8940	0.85	0.65	0.97	0.35	0.03	62’37’’
Without validation set	N/A	0.740	0.78	0.52	0.95	0.48	0.05	N/A

2.Transformation and Normalization (Data Augmentation)

We believe that how the image input is transformed also affect the accuracy of the model. We compared two types of cropping: CenterCrop, RandomResizedCrop. The effects are shown in figure 2 and figure 3. And the comparison is shown in table 3.

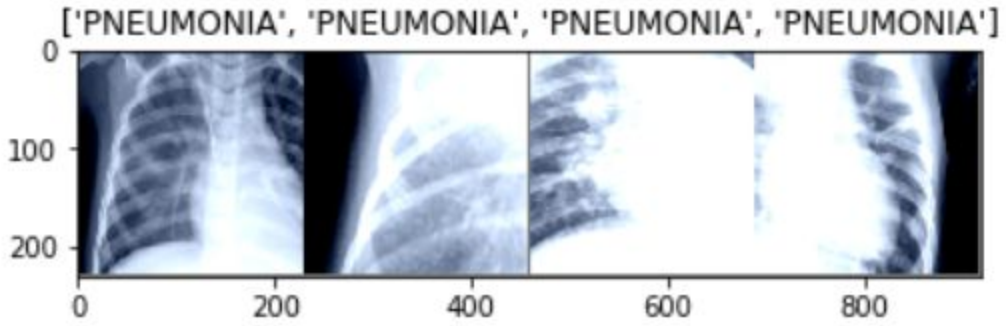

Figure 2: Tensor images after RandomResizedCrop

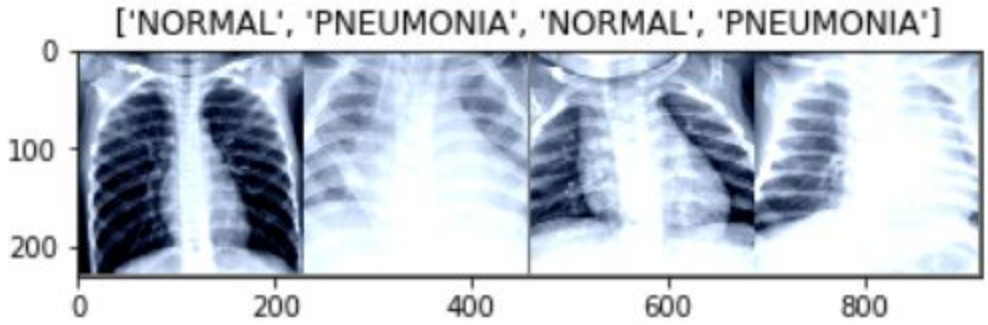

Figure 3: Tensor images after CenterCrop

Table 3: Training result of training with RandomResizedCrop and CenterCrop

	Validation acc	Training acc	Testing acc	Normal	Pneumonia	False Positive	False Negative	Time
Random Resized	0.6875	0.8940	0.85	0.65	0.97	0.35	0.03	62’37’’

Crop								
CenterCrop	0.6875	0.9860	0.79	0.46	0.98	0.54	0.02	61’28’’

Another Step we did is to normalize the picture into a tuned state so that it could serve training process better. This is for input data scaling and we referenced the value to the tutorial dataset, which are already tuned. We saw the results of the scaling, and the results seem to augment the clarity of the images. We choose to use this tuning without changing any number because we didn’t want to eliminate useful information.

3.Pooling

In choosing spatial pooling, we realized that the knowledge of knowing how to read a chest radiograph is critical on processing image input. We compared 2 * 2 max pool filter with 2 * 2 average pool filter. Max pool filter output the maximum value among a collection of input matrix whereas average pool calculate of average value of the collection of input matrix. The result is shown in table 4.

Table 4: Training results of model with AvgPool and model with MaxPool

	Validation acc	Training acc	Testing acc	Normal	Pneumonia	False Positive	False Negative	Time
AvgPool	0.6875	0.8940	0.85	0.65	0.97	0.35	0.03	62’37’’
MaxPool	0.5626	0.9078	0.71	0.45	0.91	0.55	0.08	38’2’’

4.Activation function

Activation function generalize output into either one or another such as yes or no. A sigmoid function classify output depending on the sigmoid function $S(z) = 1/(1+e^{-z})$. And softmax function calculates the possibility of the input over all possible classes. It is like sigmoid function but it works on multi-label classification. Since the chest x-ray dataset only consist of two classes, we expected that Sigmoid function would give a good performance than Softmax function.

Table 5: Train results of model with Sigmoid function and model with Softmax function

	Validation acc	Training acc	Testing acc	Normal	Pneumonia	False Positive	False Negative	Time
Sigmoid	0.6875	0.8940	0.85	0.65	0.97	0.35	0.03	62’37’’
Softmax	0.5000	0.8763	0.72	0.34	0.97	0.66	0.08	45’51’’

5.Numbers of Convolution layer

We ran this comparison to see how a layer of convolution filter affect the overall accuracy. The four-layer model is composed by a convolution block with four consecutive convolution filter. The first filter has

size of 5 * 5 and others has size of 3 * 3. ReLu layer, average pooling, and batch normalization follow the last convolution filter. In the fully connected layer, dropout, linear transformation, ReLu, and sigmoid function are used. The five-layer model is the same as the four-layer model except this model has five consecutive convolution filters. The result is shown in table 6. These model also ran ten epochs.

Table 6: Training results of model with 4 convolution filters and model with 5 convolution filters

	Validation acc	Training acc	Testing acc	Normal	Pneumonia	False Positive	False Negative	Time
4-layer	0.6250	0.8622	0.77	0.52	0.96	0.48	0.04	35’30’’
5-layer	0.6250	0.8589	0.78	0.49	0.94	0.51	0.06	37’17’’

6. Comparison on different datasets

Our model tested on other dataset with twenty epoch. Among these datasets, Alien vs. Predator and Hello_World_Deep_Learning_SIIM have two classes, and Blood Cells has four classes, Garbage Classification have six classes. As we can see in Table 7, our model works relatively well on dataset that have four classes, but in the Garbage dataset, our model is overfitted to some class over another. Although we expected the accuracy to low since we applied sigmoid function in the model. When classifying dataset with multiple labels, for instance, the Garbage Classification dataset, changing the sigmoid function to softmax function will have better performance on such dataset.

Table 7.1: Comparison on different datasets (Chest X_Ray, Alien vs Predator, SIIM)

	Validation acc	Training acc	Testing acc	Class 1	Class 2	False Positive	False Negative	Time
Chest X_Ray	0.6975	0.9043	0.79	0.52	0.96	0.48	0.04	67’43’’
Alien vs Predator	0.7000	0.7072	0.69	0.82	0.67	0.18	0.33	2’46’’
SIIM	1.0000	0.9492	1.00	1.00	1.00	0.00	0.00	0’20’’

Table 7.2: Comparison on different datasets (Blood Cells)

	Validation acc	Training acc	Testing acc	Class 1	Class 2	Class 3	Class 4	Time
Blood Cells	0.7324	0.7685	0.71	0.84	1.00	0.75	0.79	31’21’’

Table 7.2: Comparison on different datasets (Garbage Classification)

	Validation acc	Training acc	Testing acc	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Time
Garbage	0.5556	0.4938	0.59	0.71	0.68	0.26	0.63	0.53	0.00	13’36’’

7.Comparison with other models

We ran VGG-16 with batch normalization and without batch normalization, Resnet-50 and Resnet-50 with parameters from pretraining on ImageNet, and Densenet without pretrain. The comparison result is in Table 7.

Table 7: Training result of selected convolutional neural model

	Validation acc	Training acc	Testing acc	Normal	Pneumonia	False Positive	False Negative	Time
Vgg16 without batch normalization	0.6875	0.9089	0.78	0.55	0.92	0.45	0.08	68’28’’
Vgg16 with batch normalization	0.5625	0.9225	0.86	0.74	0.94	0.26	0.06	72’40’’
resnet 50(pretrained)	0.6875	0.9561	0.88	0.73	0.98	0.27	0.02	32’15’’
resnet 50(no pretrain)	0.6250	0.9133	0.87	0.80	0.89	0.20	0.11	32’29’’
Densenet (no pretrain)	0.5625	0.9112	0.85	0.80	0.88	0.20	0.12	82’58’’
Alexnet	0.5000	0.7429	0.62	0.00	1.00	1.00	0.00	53’46’’
Our model	0.6975	0.9043	0.79	0.52	0.96	0.48	0.04	67’43’’

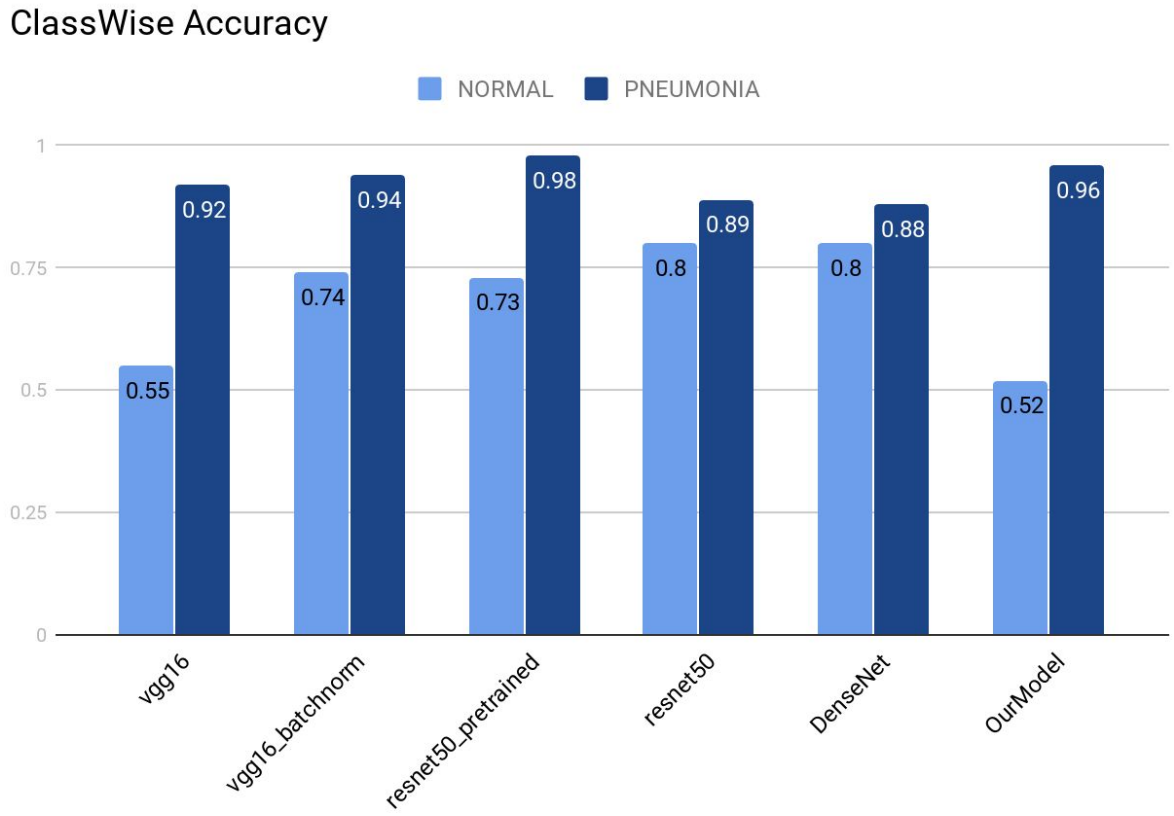

Figure 4: ClassWise Accuracy across different models

Alexnet results was dropped here because there is strangely bias towards pneumonia. We chose not to compare this model before we figure out the reason why this is happening.

Discussion

The way human diagnosis pneumonia on x-ray photos is by analyzing the blood vessel condition around chest area. As we can see the white “heart” shaped white matter is the heart. Since under x-ray blood vessels can be shown as white fibers. Mainly we will be looking at the central area on the chest where there are lot os of fibers extending from the heart and into lungs. So the most critical part of the picture we want the neural nets to recognize is the center. There are different kinds of pneumonia which causes the lung to behave differently on x-ray, generally they will make the “fiber” denser around lung area, some even create a fog like pattern that covers the entire lung. Here we will only train the neural network to tell the difference between a clear lung and a diseased lung.

For the Pneumonia dataset, we compared results between 5 main architecture and different models in general. Besides the 7 models shown in figure 4, we also have slightly modified models to try out the influence of the parameters. We choose these 7 models to discuss here because they have attributes that we find interesting to talk about.

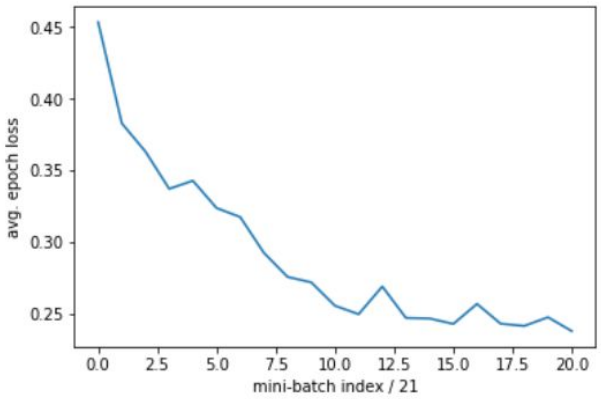

Figure 5. Train loss of our model

```
Epoch 18/20
----------
train Loss: 0.2416 Acc: 0.8995
val Loss: 1.2429 Acc: 0.5000

Epoch 19/20
----------
train Loss: 0.2476 Acc: 0.8965
val Loss: 1.2338 Acc: 0.5000

Epoch 20/20
----------
train Loss: 0.2380 Acc: 0.9043
val Loss: 1.5702 Acc: 0.5625

Training complete in 67m 43s
Best val Acc: 0.687500
```

Figure 6. Loss and accuracy of our model

First of all, we tried out our model. We started with learning rate of 0.001 and decay by 0.1 for every 7 epoches. The training process was ran with 20 epoches in total. Our model has a higher training accuracy and lower validation accuracy in the training process. Since the validation accuracy was slowly increasing, it may point to a little overfitting problem. However the problem wasn't only due to overtraining because validation accuracy was slowly increasing by every epoch. There should be more tuning to the hyper parameters to the model. It could also point to the problem that we have little validation dataset. As it turns out, the model was slightly biased with overall accuracy of 0.79 but class accuracies are 0.52 and 0.96. The result was biased into pneumonia with high false positive rates. Besides validation dataset being too small and potential overfitting, we also need to adjust convolution layers and pooling process to make the model recognize "NORMAL" class better. There is also a need an increasing data augmentations because right now random cropping may create some irrelevant data that could ruin the training process and make it more biased.

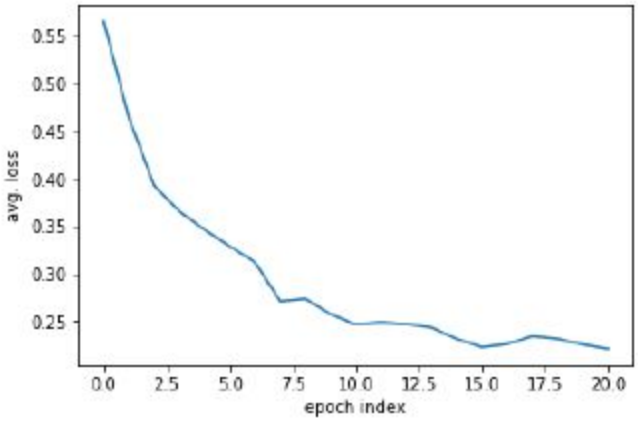

Figure7. VGG16 no batch norm

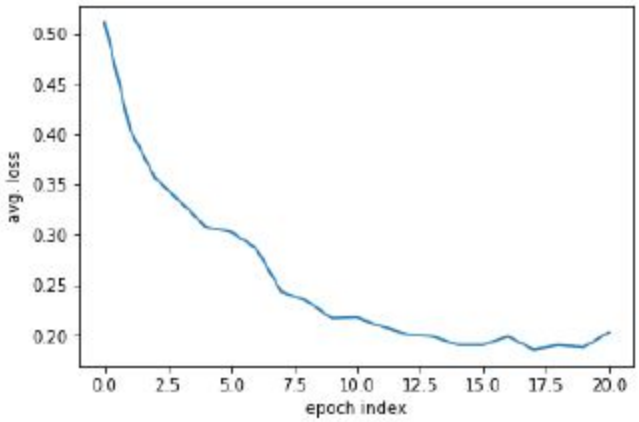

Figure8. VGG16 batch norm

```
Epoch 18/20
----------
train Loss: 0.1900 Acc: 0.9268
val Loss: 1.3966 Acc: 0.5000

Epoch 19/20
----------
train Loss: 0.1880 Acc: 0.9252
val Loss: 1.3651 Acc: 0.5625

Epoch 20/20
----------
train Loss: 0.2029 Acc: 0.9225
val Loss: 1.2720 Acc: 0.5625
```

Figure9. Loss and Accuracy of VGG16 batch norm

```
Epoch 18/20
----------
train Loss: 0.2314 Acc: 0.9032
val Loss: 0.7767 Acc: 0.6875

Epoch 19/20
----------
train Loss: 0.2253 Acc: 0.9072
val Loss: 0.8158 Acc: 0.6875

Epoch 20/20
----------
train Loss: 0.2210 Acc: 0.9089
val Loss: 0.8897 Acc: 0.6875

Training complete in 68m 28s
Best val Acc: 0.750000
```

Figure10. Loss and Accuracy of VGG 16

Then we looked at the results of resnet50 and vgg16 because the two architecture are usually put together to compare. Both are common model used on classification of images, but previous studies show that resnet model are both faster and more accurate. So we started using learning rate of 0.001 for both models. First, for VGG16, we didn't use pre-trained model because the model was pretrained on imageNet dataset, data is very different from what we tried to classify here. Besides, we want to try a model without having built in parameters in it and just see how it goes after training without dataset. Since there is an option for vgg model to train without batch normalization, we decide to use vgg 16 to see how batch normalization influence the results. We trained a vgg16 with batch size of 4 and all batches are normalized, the other one we used is a vgg16 without batch normalization. The model was both ran twice to balance noise factors and the best model was chosen to compare. In training process, although training accuracy of both models both increased from low to high, eventually around 90%. Validation accuracy decreased for batchNorm model but seems to be doing fine for no-BatchNorm Model. So we looked at the final testing accuracy, the model with batch normalization did better by almost 10% more than model without batch normalization. Besides, false positive rate is significantly decreased by batch normalizing. We think it is because batch normalization eliminated the outlying data selected when we randomly cropped the images. Although random cropping create more randomness in the model, it also creates pictures that didn't cover the most important part of the image. For example, some data we checked only contains the bone structure of the chest x-ray. Batch normalization fixes the mean and variance in each layer's input. Because out input dataset contains bone and heart information, batch normalization will help augment input data from each layer of the model. For the model with batch normalization, the reason validation accuracy decreases might be because we don't have enough validation dataset. Another big possibility is we might have overtrained the model but it still performed pretty well. This is proved by figure 9 and figure 10 which we can see train loss is way smaller than validation loss. This overfitting problem also exists in model without batch norm, with a loss of 0.22 and 0.88 for training and validation loss. We should try decreasing learning rate or shorten epochs to prevent overtraining and run the results to see how it goes. It might also because of other hyper parameters like learning rate or batch size that may create this pattern. Since the loss curve is normal and the testing accuracy was normal, we believe this model still behaves fine.

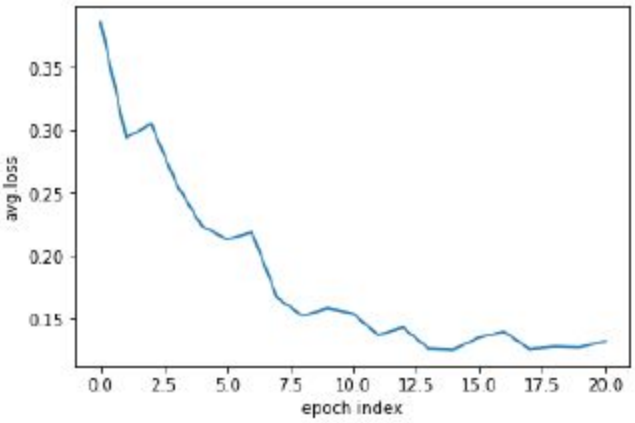

Figure11. Pretrained model Resnet50

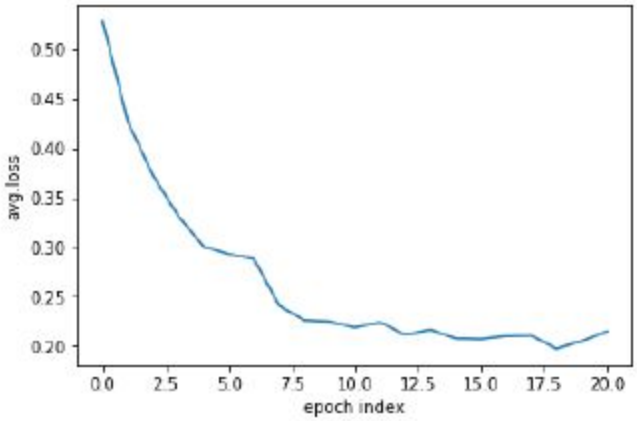

Figure12. Un-pretrained model Resnet50

```
Epoch 18/20
----------
train Loss: 0.1973 Acc: 0.9248
val Loss: 1.2271 Acc: 0.6250

Epoch 19/20
----------
train Loss: 0.2049 Acc: 0.9193
val Loss: 1.1303 Acc: 0.5625

Epoch 20/20
----------
train Loss: 0.2147 Acc: 0.9133
val Loss: 1.1517 Acc: 0.6250

Training complete in 32m 29s
Best val Acc: 0.750000
```

Figure13. Loss and Accuracy Resnet50 no pretrained

```
Epoch 18/20
----------
train Loss: 0.1279 Acc: 0.9584
val Loss: 0.7237 Acc: 0.6875

Epoch 19/20
----------
train Loss: 0.1273 Acc: 0.9588
val Loss: 0.7077 Acc: 0.6875

Epoch 20/20
----------
train Loss: 0.1320 Acc: 0.9561
val Loss: 0.6872 Acc: 0.6875

Training complete in 32m 15s
Best val Acc: 0.750000
```

Figure14. Loss and Accuracy. Resnet50 pretrained

Then we moved to see the results of Resnet50 model. We also used a learning rate of 0.001 with decay of 0.1 every 7 epochs. In general, we found resnet 50 to perform better than vgg16 anyway. For resnet50, we have pretrained model and a non-pretrained model. Both with similar overall accuracy and false positive rates. For un-pretrained model, we have more universal accuracy for both classes, and from the loss curve, we saw the learning rate and momentum we set up is good for training. However, the loss curve of pretrained model shows that we may have a too large learning rate. To increase performance of pretrained model, we need to adjust parameters to make it fit the data better. From the train loss and validation loss, we also find the model overfitting the data.

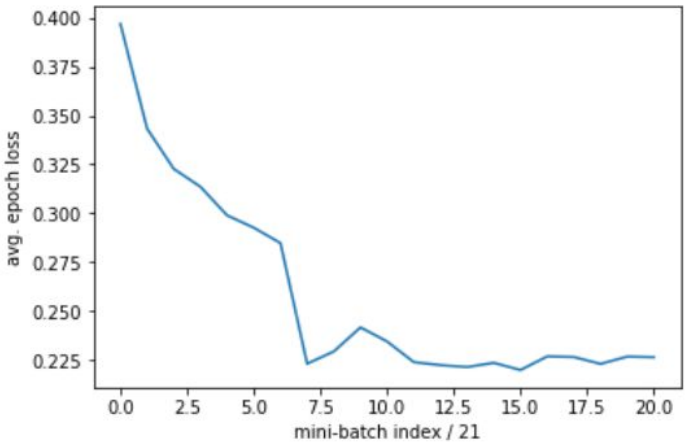

Figure 15. Training loss curve of Densenet

```
Epoch 18/20
----------
train Loss: 0.2231 Acc: 0.9187
val Loss: 1.1565 Acc: 0.6250

Epoch 19/20
----------
train Loss: 0.2268 Acc: 0.9114
val Loss: 1.3785 Acc: 0.5625

Epoch 20/20
----------
train Loss: 0.2265 Acc: 0.9112
val Loss: 1.4448 Acc: 0.5625

Training complete in 82m 58s
Best val Acc: 0.687500
```

Figure 16. Loss and Accuracy of Densenet

Last but not least, we looked at denseNet. It is a new architecture which extends from ResNet. It is supposed to be deeper, more accurate and more efficient with shorter connection between layers. We chose the same learning rate and tuning in order to compare with Resnet model. It has very high accuracy but it didn't appear to perform better than resnet50. From figure 16 we could see it has the same problem as vgg16 which training loss is way smaller than validation loss, which shows overfitting. So the validation accuracy keep decreasing over epoches. Comparing to resnet model, it has significantly increased running time, however, unfortunately we didn't get it best working so we can't compare accuracy with resnet.

Comparing all models, with the current setting, we find resnet50 model to be both time efficient, stable and accurate. Although we didn't have time to tune the model to perform best, we learned sever things from this experiment. First of all, all our model is over trained. To continue further, we would decrease the learning rate or increase drop out in all our models. We should also increase the size of validation dataset. Rightnow we don't have 30% validation vs. 70% training. So our validation accuracy while training may be affected. For our dataset, to successful diagnosis pneumonia, we should tune the hyperparameters to make it better recognize patterns of pneumonia or maybe different kinds of pneumonia. Right now we found resnet model to be superior than vgg, and denseNet which is an extension of renset is expected to perform better with more tuning. For our model, besides fixing the overfitting problem, we should edit the hyperparameters to decrease false positive rates. We believe it has the potential of performing better at diagnosing pneumonia problems

Bonus Points

We chose the option 1 as final projects. Besides comparing performance of 5 CNN architecture and many models, we also tested the difference of many hyperparameters like what difference would it make for different kinds of data augmentation, different layers, relx vs max pooling, with or without batch normalization, pretrained model vs un-pretrained, sigmoid vs softmax and so on. We also tested on 4 other datasets to consolidate our results. This paper has a word count of 4377 words.

Last but not least, we created our own CNN model and its performance is similar to vgg16 on this dataset.

Work Distribution

Huixuan Tan:
Worked on AlexNet, DenseNet, and our CNN model. Compared differences due to changes in number of convolution layers, pooling layer, activation function, and also the number of classes within a dataset.
Consolidation of different Dataset
Analyzing results of different hyperparameters

Jianing Zhang:
worked on VGG16 architecture, two models: with or without Batch Normalization
Worked on Resnet50 architecture, two models: pretrained vs un-pretrained.
Helped testing data augmentation method.
Analyzing results of resnet50, vgg16, denseNet and our Model

References

O. Er, N. Yumusak, and F. Temurtas, "Chest diseases diagnosis using artificial neural networks," Expert Systems with Applications, vol. 37, no. 12, pp. 7648–7655, 2010.

Abiyev, R. H., & Ma'aitah, M. (2018). Deep Convolutional Neural Networks for Chest Diseases Detection. *Journal of healthcare engineering*, *2018*, 4168538. doi:10.1155/2018/4168538

Mooney, Paul. (2018). *Chest X-Ray Images(Pneumonia)* [Data file]. Available from Kaggle Web site: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Migdał, Piotr. (2018). *Alien vs. Predator images* [Data file]. Available from Kaggle Web site: https://www.kaggle.com/pmigdal/alien-vs-predator-images

Wiggins, Walter. (2019). *Hello_World_Deep_Learning_SIIM* [Data file]. Available from Kaggle Web site: https://www.kaggle.com/wfwiggins203/hello-world-deep-learning-siim

cchangcs. (2018). *Garbage classification* [Data file]. Available from Kaggle Web site: https://www.kaggle.com/asdasdasasdas/garbage-classification#one-indexed-files-notrash_test.txt

Mooney, Paul. (2018). *Blood Cell Images*[Data file]. Available from Kaggle Web site: https://www.kaggle.com/paultimothymooney/blood-cells#dataset2-master.zip

Simonyan, K., Zisserman A. (2015, Apr 10 Published). *Very Deep Convolutional Networks For Large-Scale Image Recognition*. Retrieved from https://arxiv.org/pdf/1409.1556.pdf

Huang, G., Liu Z., Maaten, L. v., Weinberger, Kilian Q. (2018, Jan 28 Published). *Densely Connected Convolutional Networks*. Retrieved from https://arxiv.org/pdf/1608.06993.pdf

Krizhevsky, A. (2014, Apr 26 Published). *One weird trick for parallelizing convolutional neural networks*. Retrieved from https://arxiv.org/pdf/1404.5997.pdf

He, K., Zhang X., Ren S., Sun J. (2015, Dec 10 Published). *Deep Residual Learning Image Recognition*. Retrieved from https://arxiv.org/pdf/1512.03385.pdf