COVID Kaggle Competition (Spring 2020)

Menglin Shao
Uni: ms5959
Columbia University
COMS 4771 Project

1 Abstract

In this study, three CNN models are proposed to learn the classifying approach for the patients' current health conditions based on their chest X-ray images. One is my own defined CNN model, and the other two are popular transfer learning models, InceptionV3, and ResNet50. Data augmentation is also applied to deal with the lack of sample data, so as to support deep neural network models. The overfitting problem is solved by freezing some redundant layers in the deep network. Performance results show that pre-trained ResNet50 yields the highest accuracy among all the models on both training set and validation set, and also get 82% accuracy on the Kaggle test set.

Keywords: Image classification; CNN models; Transfer learning method.

2 Data processing

The data set is consisting chest X-ray images and a set of multiclass labels indicating whether each patient was healthy or diagnosed with bacterial pneumonia, viral pneumonia, or COVID-19.

Firstly, every image is imported and convert to an array. All images in this dataset are resized to $256 \times 256 \times 3$ pixel size and are normalized by dividing 225. Then, they are split into 4 groups corresponding to the image's actual lung situation. From figure 1 below, it represents that we have 350 patients who are normal, and the same number as for bacterial pneumonia, viral pneumonia; but we only have 77 images that belong to Covid-19. Because of the unbalance in the dataset, a fixed percentage of

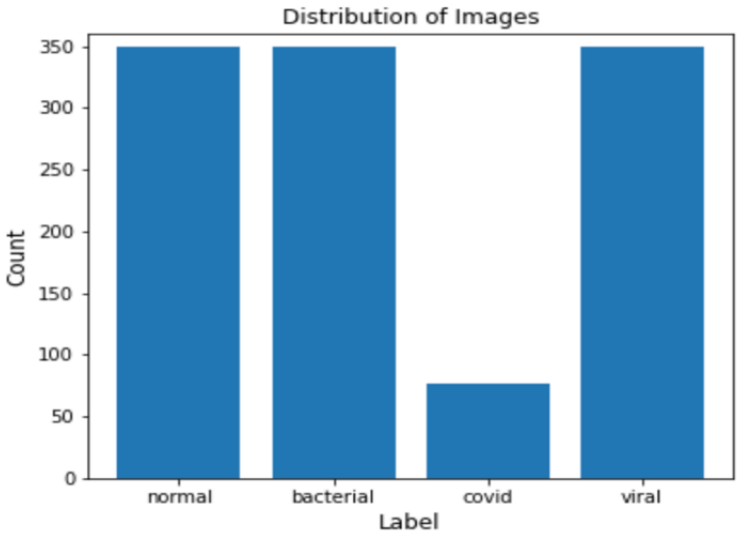


Figure 1: Distribution of image classes

70% are used for every classification during the train set and validation set are split. Moreover, since the number of samples in the dataset is 1127 which is not sufficient to conduct deep learning models, data preparation and augmentation is applied to expand the dataset. In order to handle different rotations and off-center shifts, or other possible transformation, such as flip or redundancy in the scene, Keras image augmentation API is utilized to enriching the possible cases in datasets and avoid noise in the test set. The augmentation part is loop 3 times, and thus, the model is trained on 3168 samples and validated on 335 samples.

3 Model Implementation

Deep learning models are successfully used in the classification area, especially for image classification. Convolutional Neural Networks (CNNs) is the most popular neural network model being used for image classification problems. The practical benefit for CNNs is that having fewer parameters greatly improves the time it takes to learn as well as reduces the amount of data required to train the model. This benefit can directly solve one of the biggest difficulties in the analysis of physical image data, which is the insufficiency in the sample data size.

Therefore, a CNN model is firstly implemented (model summary in Appendix 1). It consists of batch normalization and dropout layer with 0.3 dropout rate, then, Relu is used as the activation function, and SGD is applied for optimization. However, the slow convergence rate in the loss function is a problem for this CNN model. As the convergence results are shown in figure 2, in the accuracy plot, this model achieves 100% accuracy on the training set, but to go from approach 100% accuracy in the training set to fluctuations around 72% in the validation part, it demonstrates a clear problem with overfitting. However, this is still more of an art than a science and it is very hard to calculate exactly how to reduce overfitting without trial-and-error experimentation.

In order to solve this overfitting problem which is probably because of the lack of image sample, the transfer learning method is then applied. With the transfer learning method, the weights information obtained by the pre-trained model on a large dataset (ImageNet) is transferred to the model to be trained.

In this project, I applied ResNet50 (model summary in Appendix 2) and InceptionV3 (model summary in Appendix 3) models for the multi-classification for chest X-ray images as the pre-trained models. Then, training phases are added. A global spatial average pooling layer is added, and a fully-connected layer with activation function Relu, as well as the output layer with Softmax as the activation function. In addition, to avoid the overfitting problem, a dropout layer is included with a dropout rate equal to 0.5. The input to the model is an array of image data, which have a dimension of 256*256*3. It split into two sets, with 70% is for the training process, and the other 30% is used for validation evaluation.

The output is an array of predicted class possibilities. The model evaluation functions are also involved in the training process, where loss, accuracy, f1-score, and recall are calculated on both training set and validation set as a compare. All experiments were performed using Python on Google Colaboratory. The batch size, learning rate, and the number of epochs were experimentally set to 32, 0.0001, and 30, respectively for all experiments.

4 Results Evaluation

4.1 CNN

In figure 2, the training accuracy approach 1 quickly, while the loss slowly approaches to 0. This is also true on the validation set, with accuracy achieves around 70% after less than 5 epochs but on the other side, the loss is continuously decreasing and not converge. It may because the sample size is not enough to support CNN models, and thus, resulting in an overfitting problem. Since the accuracy is not idle, other transfer learning methods are applied to make up the lack of image samples.

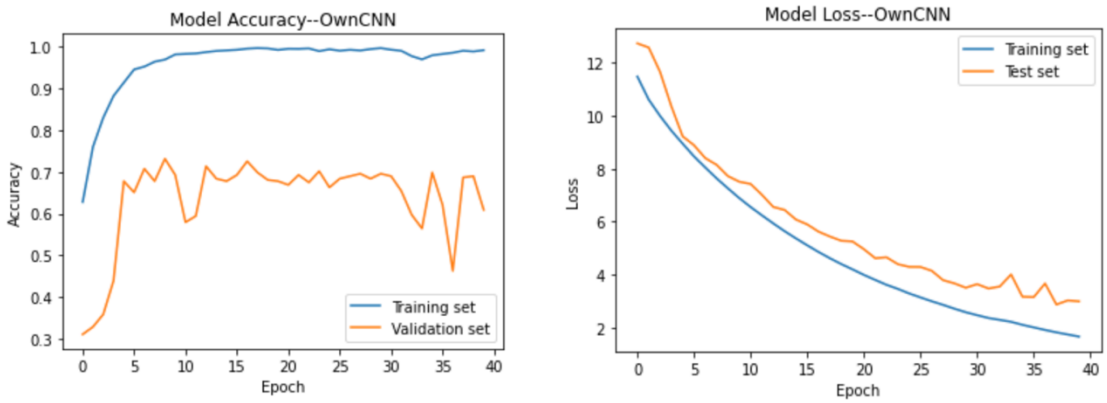


Figure 2: Model performance for CNN

4.2 InceptionV3

From figure 3 below, the training loss is decreasing and accuracy is increasing to approach 1. However, the validation results are not that idle. There is no significant improvement for both accuracy and loss on the validation set. Therefore, it may be a result of overfitting. I have tried several methods to prevent overfitting, such as enlarging the data size when doing data augmentation, or freeze some layers in InceptionV3 to avoid too complicated model, but did not explore any better idea to fix this fluctuation without growth trend. In this case, ResNet50 is used since it has fewer layers then InceptionV3.

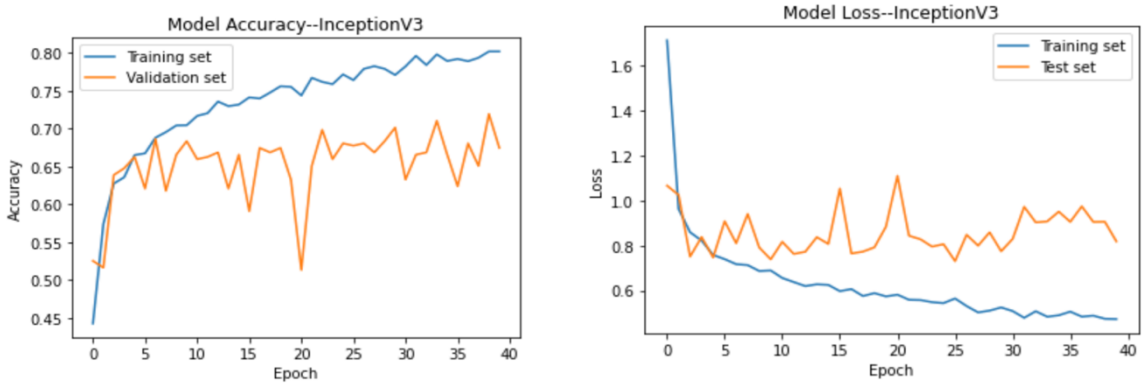


Figure 3: Model performance for InceptionV3

4.3 ResNet50

It can be seen that, in figure 4, ResNet50 shows the highest accuracy on the validation part, and the fastest loss decreasing process than other models tried before. Thus, ResNet50 is relatively stable, and to a certain extent solves the problem of overfitting encountered by the previous two models.

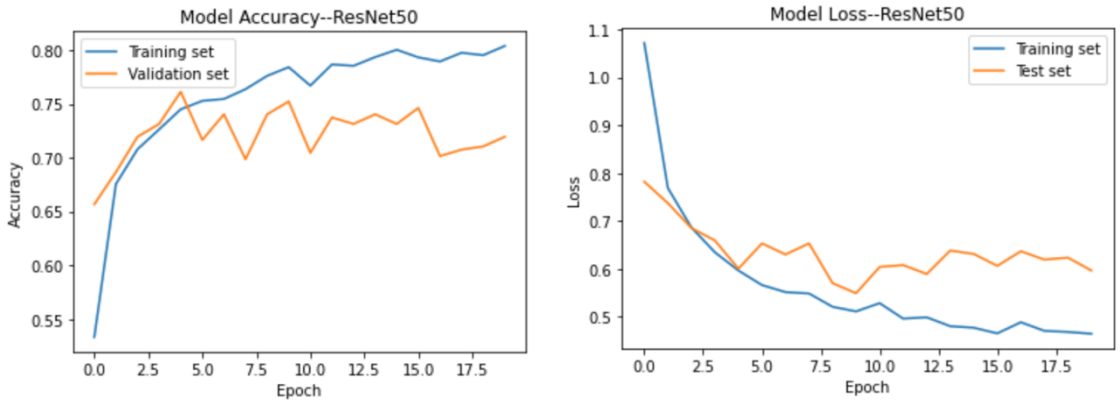


Figure 4: Model performance for ResNet50

4.3.1 Confusion Matrix for ResNet50

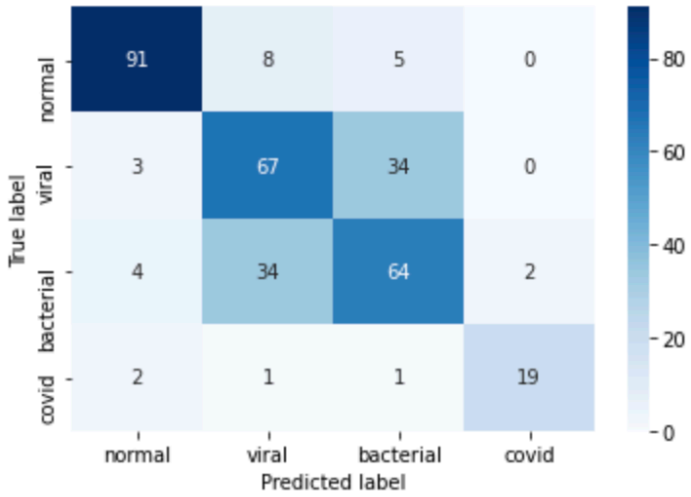


Figure 5: Confusion matrix for ResNet50

In this figure 5 above, we have two sections to be outlined. The diagonal entries are corresponding to Ture Positive where the predictions label belongs to a particular class was correctly classified as the said class. While the off-diagonal entries are the ones that predict incorrectly. Then, from the heatmap below, the diagonal elements have significantly higher values than the other parts, which indicates our classifier has certain accuracy. It's very interesting that the confusion matrix shows the accuracy of viral and bacterial is very low. It may indicate that the similarity between viral and bacterial is high on the x-ray chest image.

4.4 Model Comparison

In another detailed performance, comparisons of three models using the training data and validation set are shown in Table 1. As for the training set, around 99% of accuracy is obtained on the training set for both the CNN model and ResNet50. However, we have obtained the best performance as an accuracy of 98%, recall of 96%, and specificity value of 100% for ResNet50 pre-trained model. The lowest performance values have been yielded an accuracy of 67%, recall of 66%, and a f1-score of 68% for our CNN model. As a result, the ResNet50 model provides superiority over the other two models both training and validation stage.

Table 1. Model Comparison

Model	set	loss	accuracy	f1-score	precision	recall
CNN	train	1.6726	0.9917	0.9916	0.9916	0.9916
	validation	3.0008	0.6090	0.6100	0.6132	0.6068
InceptionV3	train	0.4728	0.8020	0.8007	0.8151	0.7874
	validation	0.8183	0.6746	0.6815	0.7036	0.6612
ResNet50	train	0.0034	0.9990	0.9990	0.9990	0.9990
	validation	1.6260	0.7761	0.7660	0.7672	0.7648

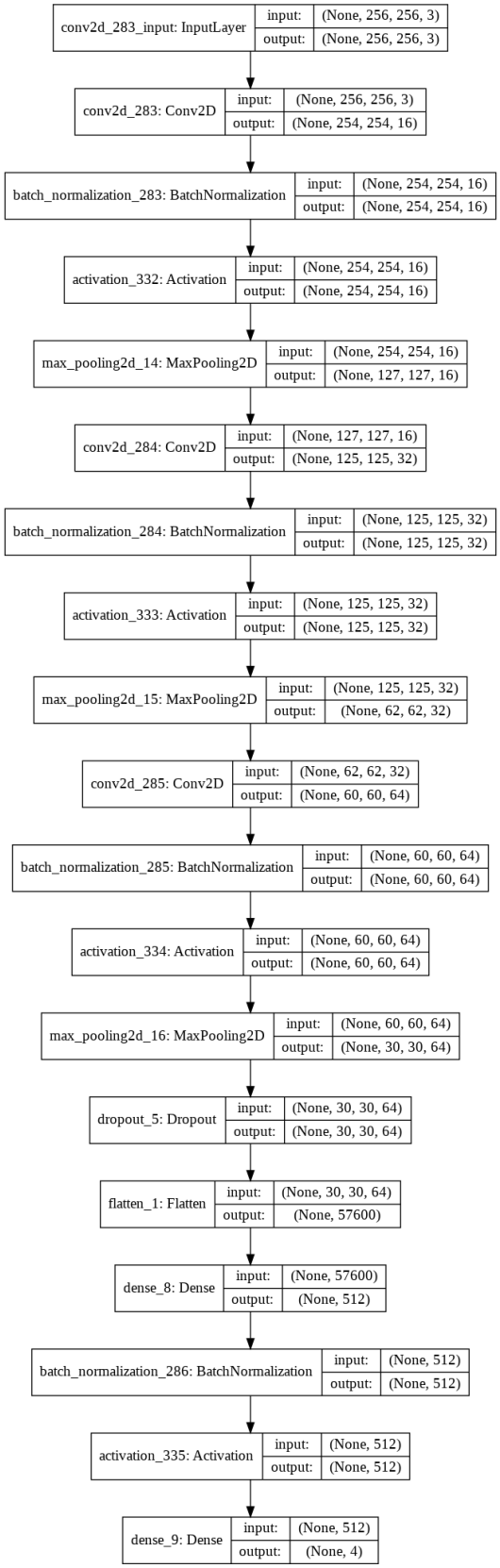
4.5 Model Interpretation

It is often said that a deep neural network is a black box, and it is very difficult to understand how the model makes predictions. However, the interpretability of models is very important to trust the model, and therefore it is very important to understand how the model predicts. After doing some research online, I found that people use Gradient-weighted Class Activation Mapping (Grad-CAM), which uses the class-specific gradient information flowing into the final convolutional layer of a CNN to produce a coarse localization map of the important regions in the image. Other researchers also aim to visualize the intermediate convolution layer outputs. It will help us understand how the input is transformed by the layers and how different filters are learned by the network.

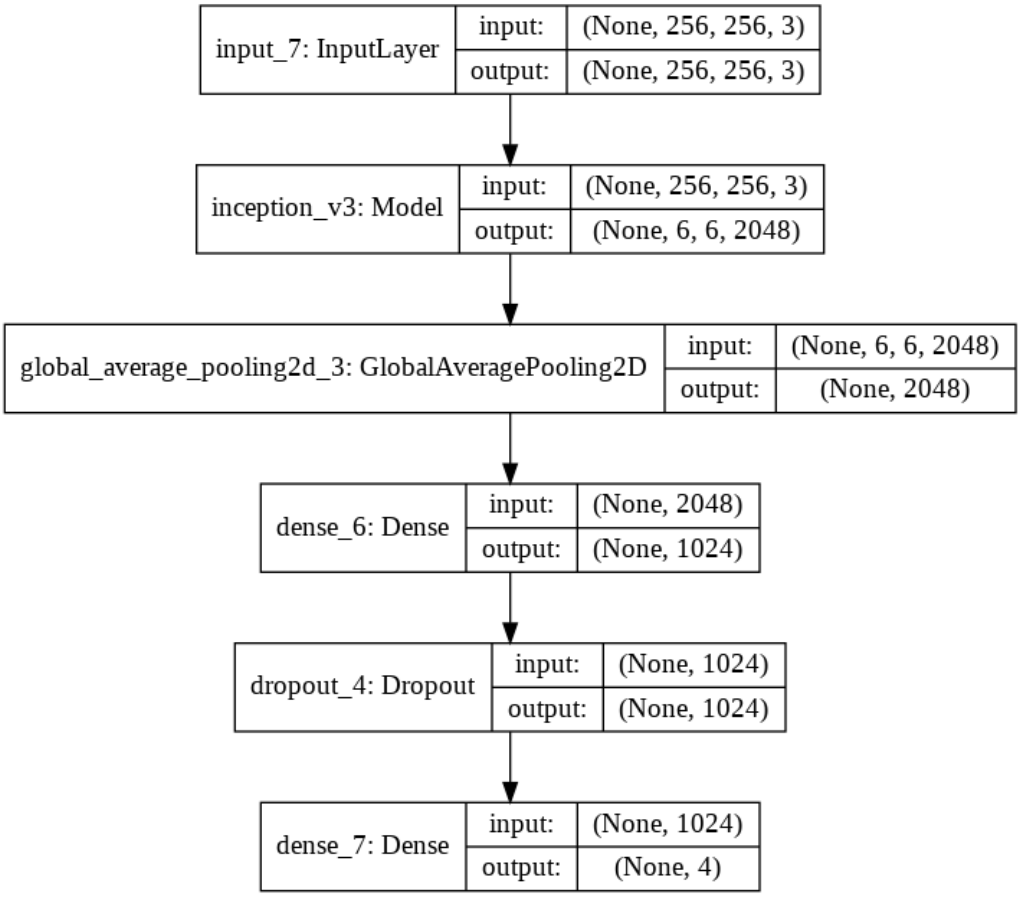
5 Conclusion

When I was implementing this project, I first tried to use some very simple linear classification models, like SVM. However, it did not achieve relatively high accuracy as the CNN models I used later. Thus linear classifiers, despite being a powerful tool and can extract interpretable features, fail to differentiate between extremely close classes, and thus, cannot accurately classify the chest X-ray images. While, when I tried CNN models, I was facing the problem of insufficient data, since deep learning models always require large datasets to support deep network mining. In this case, I utilized data augmentation to enlarge the dataset and also perform transfer learning methods by using pre-trained models on lmageNet. These two approaches successfully improve the accuracy to 99% on training set, but due to the fact of overfitting, we only achieve around 75% accuracy on the test set. Therefore, I reduced the number of layers of the network and froze some redundant layers, and finally got 82% accuracy using ResNet50 on the test set.

Appendix 1: Model Summary for own defined CNN



Appendix 2: Model Summary for InceptionV3



Appendix 3: Model Summary for ResNet50