COMS 4771 Machine Learning (2020 Spring)
Problem Set #Covid-19 Kaggle Project

Leyi Mai - lm3504@columbia.edu

Early prediction of COVID-19 cases is crucial in lowering the spread of the virus to other people. While Chest X-rays(CXR) was utilized by some physicians to diagnose COVID-19 early in the pandemic, it can sometimes be difficult to distinguish COVID-19 from ordinary pneumonia. In this report, I will explore different machine learning models to establish a multi-class classifier, which can classify any CXR image into one of the four classes: healthy, viral pheumonia, bacterial pneumonia and COVID-19.

Specifically, this report will discuss in details about the model exploration, data preprocessing, model training and evaluation, interpretation and reflection on results.

1. Model Exploration & Selection

Firstly, I explored different traditional machine learning algorithms learnt from this course. Considering the given 1127-image dataset, it is a relatively small dataset, and each image could be translated into a large number of features representing each pixel. In this sense, applying simple machine learning methods like Naive Bayes, KNN or Decision Tree will not be sufficient. Therefore, I experimented with Random Forest and Support Vector Machine, due to its ability to deal with a greater number of features and good interpretability. However, the model fails to achieve satisfactory accuracy on validation dataset, only achieving approximately an accuracy of around 55% to 65% after fine tuning and cross validation. It indicates that I might need some deeper models to adapt to the complexity of this task.

Therefore, I went further to explore deep learning models. From research I learned that Deep Convolutional Neural Networks(CNN) have performed well in image classification, which could learn the characteristics of an image by processing the input through a series of convolutional layers, pooling layers and fully-connected layers. I tried to implement self-built baseline CNN as well as existing architecture like VGG16, ResNet50 with Keras library.

After initial exploration, the 5-layer baseline CNN model is not complex enough to deal with this task and has failed to reach a higher accuracy. As for VGG16 CNN model, although it is of good potential to reach high accuracy, it can be too time-consuming to train due to its depth and number of fully-connected nodes and can sometimes grow too complex that results in accuracy degrade . Instead, the ResNet50 architecture has not only enabled deeper learning levels but also is relatively easier to train, since it allows shortcuts between layers which help significantly reduce the number of parameters while preventing distortion. Therefore, I decided to use the transfer learning based approach and implement ResNet50-based CNN in the training of this 1128-CXR dataset.

Reference 1: Convolutional Neural Network for Visual Recognition
Reference 2: Densely Connected Convolutional Networks

2. Image Data Preprocessing

Data preprocessing on these CXR image data will be necessary to adapt them into the ResNet50 CNN model and achieve better performance during training. Hence, the following steps of data preprocessing have been implemented.

1. Resizing and Cropping

   Since the given CXRs are drawn from different sources and are of different size and quality, the first step is to resize and crop these images into a same size and same shape, so that it can be input into the ResNet50 model. By applying keras preprocessing image library, I resized the data into $224 \times 224 \times 3$ pixel values and stored them in arrays.

2. Normalization

   Next, I normalize the data by scaling the pixel values from 0-255 down to 0-1 by dividing the values by 255. This helps ensure each input feature has similar data distribution and accelerate convergence when training the network. If this step is not taken, the convergence will take longer time since distribution of different feature values will likely be different, which complicate the learning process of the network.

3. Onehot Encoding on Labels

   To build the multi-class classifier, I also applied the one-hot encoding on the label column to generate four separate label columns each representing one of the four classes 'bacterial', 'viral', 'normal' and 'covid' . The last layer will use the activation function 'softmax' to output the class with the greatest probability. Without this step, the model will not be able to handle single-column four-class output layer.

4. Data Augmentation

   Lastly, I tried implementing the data augmentation with the ImageDataGenerator tool in Keras, by randomly rotating the image within 20 degrees and shifting for 10% in width and height. This is in an attempt to expose the network to a wider variations so that the model is less prone to undesired characteristics, for example, the angle of the CXR taken, and thus is less likely to be overfitting. This data augmentation has helped stablized the classifier, which has lowered training accuracy increase rate during epochs but improved the accuracy on testing test for 3%.

In figure 1, the resized representative chest X-ray images for bacterial pneumonia, covid-19, normal, viral pneumonia are displayed respectively.

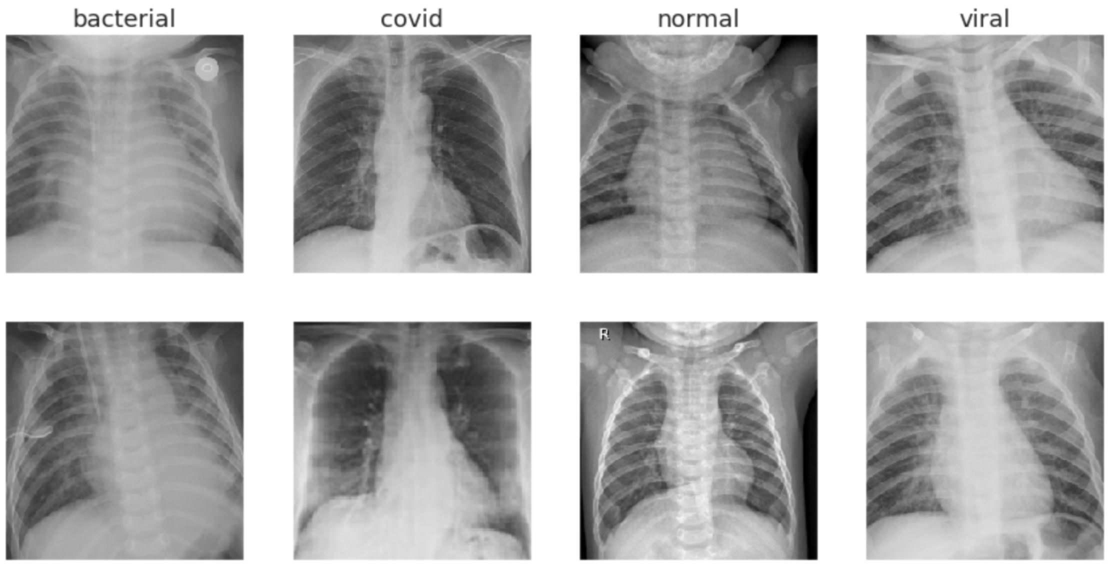


Figure 1: Representative chest X-ray images

Reference 3: Image Data Pre-Processing for Neural Networks

3. Model Training & Error Analysis

Based on the ResNet50 architecture, the model was modified to include a total of 53 convolutional layers in total, the first using a filter of size 7 × 7 and the latter using a filter of size 3 × 3. Batch normalization layers and spatial dropout layers were implemented intermittently to prevent overfitting.

The next step was to tune the hyperparameters of the model. To see which sets of hyperparamters lead to the most stable best result, I tried different combinations of crucial hyperparameters, including batch size, number of epochs, dropout rate and optimizer learning rate. To evaluate the performances of each model, I used 10% of data as validation set and plotted the validation accuracy curve and loss curve. As a result, the model accuracy was improved from 70% to 77% by setting the batch size as 32, number of epochs as 100, dropout rate as 0.2 and learning rate as 0.0005.

As is shown in Figure 2, the model has reached accuracy of approximately 73% on both the training and testing set, with the the loss on both sets converged nicely at the later epochs. In figure 2, the learning curves of validation accuracy and loss are shown.

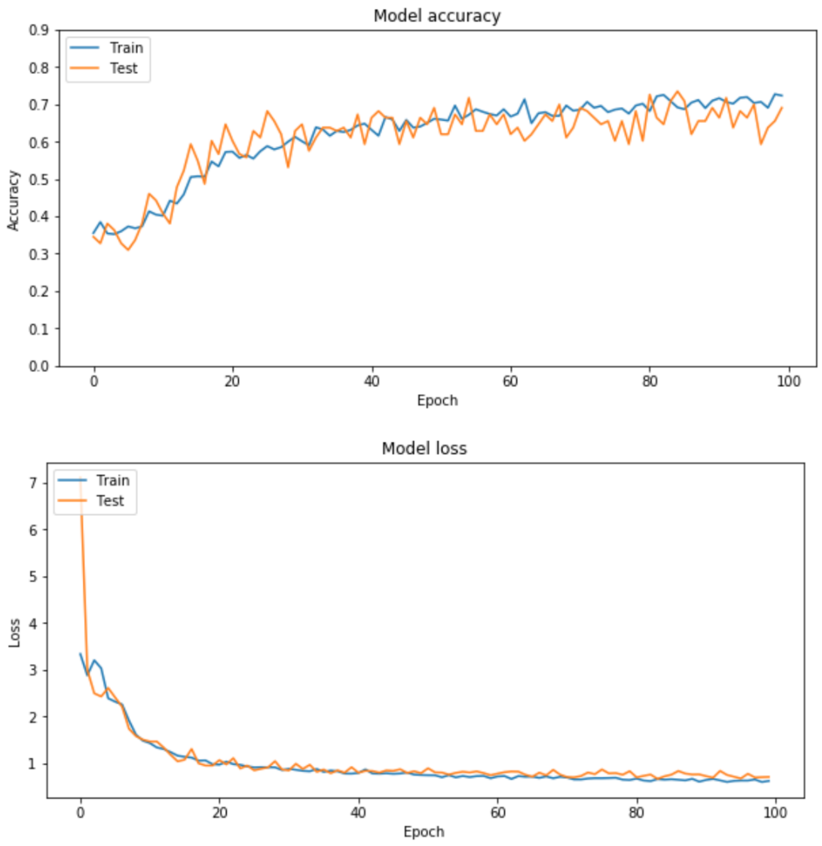
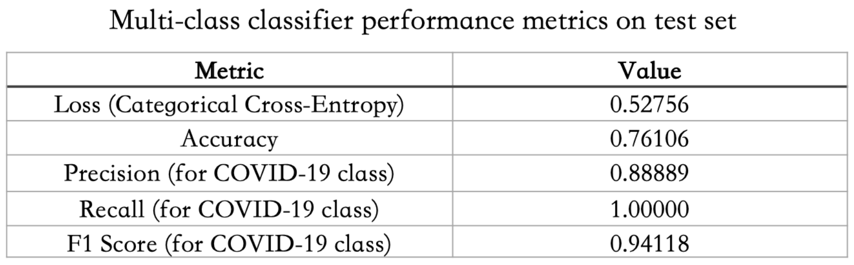


Figure 2: Learning Curves of Best Model

Conducting a detailed error Analysis would also be helpful and meaningful in evaluating the model performance and its implication for clinical setting. The model evaluation metrics, confusion matrix and performance curves of this particular model are shown in Figure 3, Figure 4 and Figure 5, respectively.



Multi-class classifier performance metrics on test set

Metric	Value
Loss (Categorical Cross-Entropy)	0.52756
Accuracy	0.76106
Precision (for COVID-19 class)	0.88889
Recall (for COVID-19 class)	1.00000
F1 Score (for COVID-19 class)	0.94118

Figure 3: Model Evaluation Metrics on Test Set

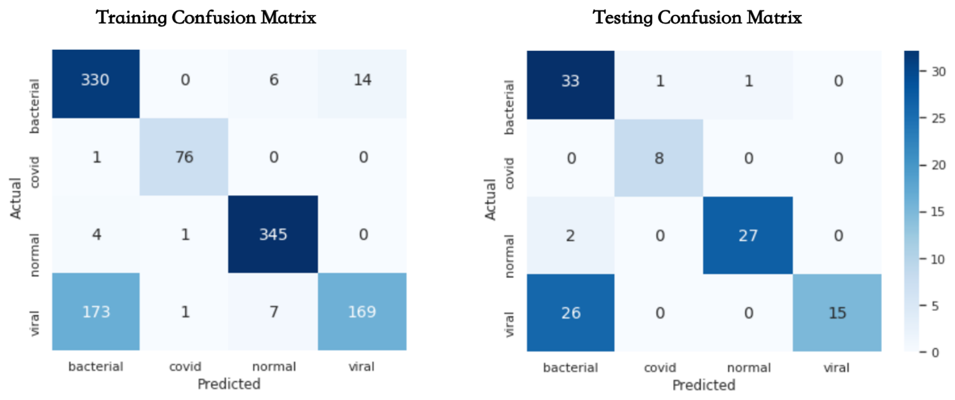


Figure 4: Confusion Matrix

As is shown in these results, the model achieves a 76% accuracy on the test set and an almost 100% recall rate (100% on test set and 98.7% on training set). From the confusion matrix, we can see that viral and viral pneumonia are sometimes confused with each other, and normal cases are sometimes classified as bacterial pneumonia or covid.

Considering the covid background, this is a relatively satisfactory result, since failing to identify a covid-19 case is dangerous for the fast spread of covid-19 virus. Although we do not want to falsely identify negative covid-19 case to be positive which would worsen the medical resource shortage, identifying true positive is so important and crucial that we can accept a certain level of false positive cases. The point is that if the recall rate of this classifier is trained to be high enough, it could assist physicians in the diagnosis of COVID-19 with the help of Chest X-rays, especially when the testing kit supply is falling short of the demand.

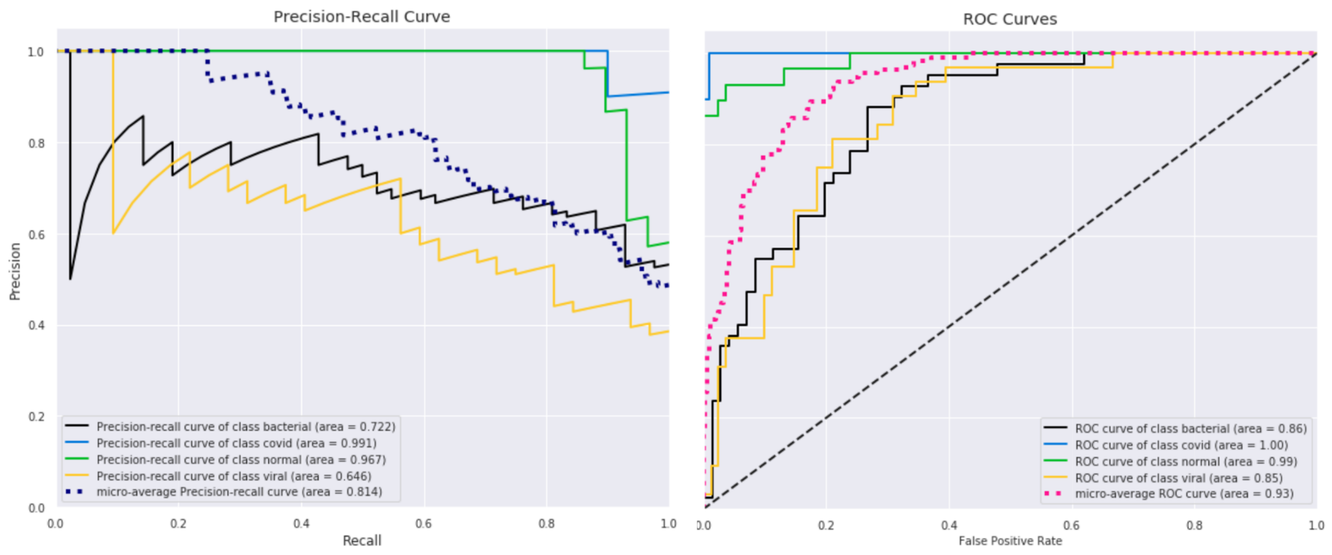


Figure 5: Presion-recall curve and ROC curve

Therefore, in this COVID-19 setting, we would prefer a model with higher recall or true positive rate, even with a slight drop in precision and slight increase in false positive rate in the tradeoff between recall and precision, as well as between true positive rate and false positive rate.

Reference 4: Correlation of Chest CT and RT-PCR Testing in Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases

4. Model Interpretability

While aiming for higher accuracy and recall is one of the main goals, the model's interpretability is also important, which would provide insight on what features are guiding the model's prediction in classifying an CXR image. Of the models that I explored, Random forest and Support Vector Machine are generally models with good interpretability. The idea of using multiple decision trees to vote or finding the optimal boundaries of these models make them easier to interpret, with feature importance being more comparable and the decision process being more transparent and understandable.

In comparison, neural network is relatively hard to interpret. The hidden inter-connected layers are like a black box. Nevertheless, there are some tools that we could explore to visualize the convolutional neural network so that we can better interpret what are the features extracted and used to distinguish between different classes.

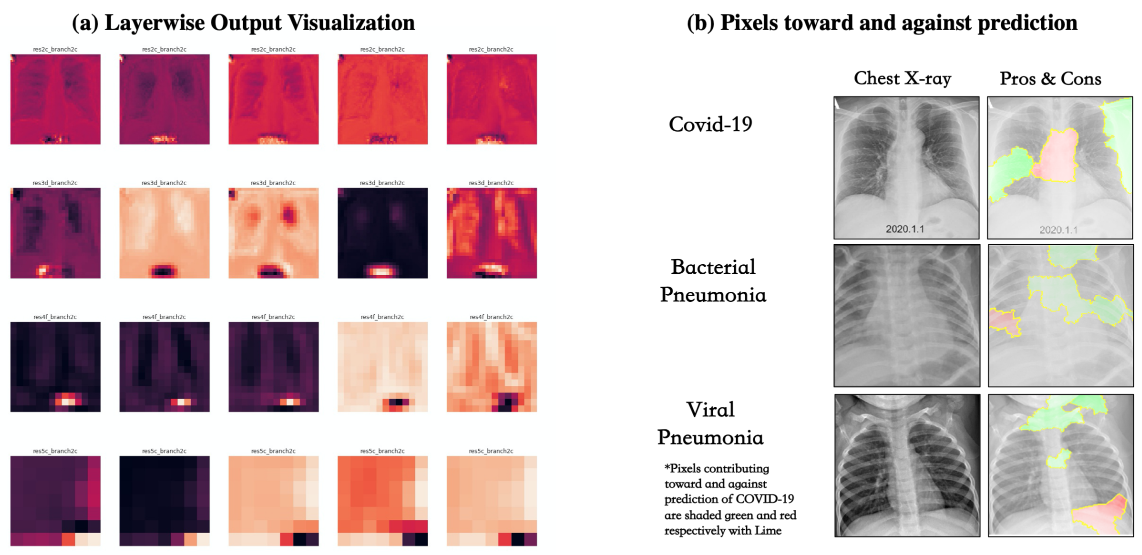

(a) Layerwise Output Visualization
res2c_branch2c
res3d_branch2c
res4f_branch2c
res5c_branch2c
(b) Pixels toward and against prediction
Chest X-ray
Pros & Cons
Covid-19
Bacterial Pneumonia
Viral Pneumonia
*Pixels contributing toward and against prediction of COVID-19 are shaded green and red respectively with Lime
2020.1.1

Figure 6: Visualization of the Model

In Figure 6a, the output of four main blcoks are visualized, which shows different features that are extracted at different layers. We can see that the starting layers look at the low-level features including the edges and shade of the lung , whereas the later layers look at higher-level features like some specific positions inside lungs. The visualization of the layerwise output could help us understand the hidden processing of the model to see which pixel areas are crucial in predicting the classes

In Figure 6b, I used Local Interpretable Model-Agnostic Explanations (LIME) to explain the predictions of the Convolutional Neural Network classifier that I trained. Lime works by perturbing the features in an example and fitting a linear model to determine which features were most contributory to the model's prediction for that example. As is seen in Figure 6b, superpixels in green indicate regions that were most contributory toward the predicted class, while superpixels coloured red indicate regions that were most contributory against the predicted class. This methods provide a very clear insight for physicians which areas inside the lung they should pay special attention to in the diagnosis of COVID-19.

Reference 5: Understanding Convolutional Neural Networks (CNNs) using Visualization
Reference 6: Investigation of Explainable Predictions of COVID-19 Infection from Chest X-rays with Machine Learning

5. Reflection

During the training, I have encountered difficulties in making breakthroughs in accuracy improvement, dealing with overfitting issue and increasing the speed of training.

At early stage of model exploration when I implemented the traditional machine learning algorithms like Random Forest, I found the accuracy was stuck at around 60% regardless of fine tuning and cross-validation. Then I looked through some literature and found out that ResNet50-based or VGG16-based CNN would more suitable models in image recognition, especially for complex tasks like recognizing COVID-19 from CXRs which might require deeper models and some pre-trained architectures. So the first lesson learned was that whenever I were to train a machine learning model on a specific dataset, I should research and look for a suitable model that have worked for similar problems and try to build on existing architectures or domain-specific models and then customize on the basis of that.

When I implemented the ResNet50 CNN model, the accuracy was improved to around 70% and it become difficult to make futher breakthroughs. Therefore, hyperparameter tuning would be necessary to further increase the accuracy. I selected the crucial hyperparameters, which included number of epochs, batch size, dropout rate, dropout layers and the learning rate. I conducted grid search and cross validation in finding the optimal hyperparameters. The accuracy has been improved to about 77% after explorations. For future exploration, customizing the last couple of layers might be also helpful in adapting the model to achieve higher accuracy on this dataset.

Furthermore, considering that this dataset was unbalanced with only 77 covid cases among 1127 total cases while the most important task was to identify covid cases, it might also be helpful to apply class weights in model training by giving the underrepresented class - covid - a higher weight to put more emphasis on a covid, or customize loss function by excessively penalizing false negative cases of covid-19. I tried applying the class weight method yet the performance decreased, potentially because the it might lead to decreasing accuracy in the other three classes which consist of most of the dataset. Customizing the loss function would be worth for further exploration.

Overfitting was another issue that was challenging to tackle due to the relatively small size of this dataset. When I first implemented the ResNet50 model with 90% of the data being training set and the other 10% being validation set, it achieved approximately 95% accuracy in the training set but only 75% in the holdout validation set and 70% in the testing data on kaggle. It indicated the overfitting issue, i.e., learning too much noise and detail in the training set that it negatively impacted the performance of the model on unseen data. This could be common for deep models and the way I handled it was through regularization and data augmentation. I added spatial dropout layers and batch normalization layers after every convolution layers, which resulted in a lower training accuracy growth rate and yet higher final testing accuracy. From this I learned that applying normalization might be helpful in stabilizing the learning of deep neural networks, which would lead to a more stablized better result when predicting on the unseen data.

For future exploration, I would also try to mitigate the overfitting issue by collecting more covid-19 CXR images and refine the model based on the enlarged dataset. And I would also like to try building multiple stable models and let the majority of vote by these classifiers be the final predictions, to further enhance the model and improve prediction accuracy.

Reference 7: Tips for training deep neural networks