COVID-19 Diagnostics using Image Classification

1st Mohammad Bahrami
dept. of Mechanical Engineering
Stevens Institute of Technology
Hoboken, USA
mbahrami@stevens.edu

2nd Sivankit Bhanot
dept. Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, USA
sbhanot@stevens.edu

3rd Sneha Kawle
dept. of Mathematical Science,
Computer Science and Electrical Engineering
Stevens Institute of Technology
Hoboken, USA
skawle1@stevens.edu

***Abstract*—We have attempted to classify thoracic X-ray images into three classes of 'Normal', 'Pneumonia' and 'COVID-19'. We used off-the-shelf neural network architectures to extract features from these images and then applied various different techniques for classification including Decision Trees, Softmax Regression, and SVMs. For the initial dataset we observed good performance on the training and validation sets but ended up with very poor performance on the test set for all but one of the ML techniques applied. This raised concern about the reliability of the test set of our initial dataset and we decided to train on another dataset. It is difficult to find larger datasets for COVID-19 thoracic X-ray images due to the novel nature of the virus and the disease. For this reason, and also consulting other similar research in the area, we used transfer learning in all our machine learning methods. The neural networks used to extract features have been pre-trained on the ImageNet dataset and have weights from being trained on that dataset. For the second COVID-19 dataset, we observed a good performance on the test set (reaching a max of 100% accuracy, averaging at about 83.71% accuracy). Applying these methods gives us some more groundwork to cover for future in this area, such as applying all the methods in an ensemble and then using that to predict instances from the first dataset. Transfer learning could also be applied by first training on a large thoracic X-ray dataset instead of ImageNet, and using those weights to extract features.**

***Index Terms*—Image classification, feature extraction, Convolutional Neural Networks (CNN), COVID-19**

I. Introduction

With the outbreak of the novel coronavirus, called COVID-19, most, if not all, governments have been struggling to discover a solution to mitigate the spread of the virus as their first immediate response. This necessitates diagnosing affected people by the virus in the early stages which is a challenging task. Part of the difficulty has been associated with the shared symptoms of COVID-19 with those of other certain illnesses such as the common cold, flu, and pneumonia. In this project, we work on developing an image classifier for diagnosing COVID-19 from X-ray images of potential patients.

II. Related Work

As a promising approach of diagnosing COVID-19, there is a vast increase in the different machine leaning-based techniques used to classify X-ray images of a patient infected with COVID-19 [1]–[4]. In order to classify images we need to know some features that characterize the difference of images across different classes. As a result, a machine (algorithm) is able distinguish between different classes of images by measuring those features. As images varying in size, resolution, contrast, and so on, they are considered as complex and high-dimensional data for feature extraction. One effective approach is to preprocess data into a set of images with the same size and feed them into a pre-trained CCN to extract a set of features often from the penultimate layer of the network, known as transfer learning for feature extraction [4]. Certain attempts have made to employing Transfer Learning technique along with a convolutional neural network (CNN) architecture for extracting features associated with COVID-19 from X-ray images. In this technique, first, the convolutional layers of a neural network are trained on a larger dataset. Having trained the network which is capable for feature extraction, the dataset of X-ray images that we are working with (which is also much smaller compared to the larger one) is passed through the neural network for classification. As a results, the weights that are attained after training on the larger dataset can be useful in extracting the relevant features from our dataset of interest.

As Apostolopoulos et al. have mentioned in [1], transfer learning is particularly useful in the case where we have a smaller dataset of images. This technique helps in training our network over a larger dataset. Results in [1] also suggested VGG19 [5] and MobileNet [6], two common CNN architecture, as the best performing CNNs using transfer learning on their X-ray image dataset. Basu et al. in [7] identified greater benefit from the use of domain extension transfer learning. they showed that the neural network performs better if the dataset used for training is close to the dataset that we want to extract its features.

III. Our solution

In this project we consider the problem of classifying thoracic X-ray images to detect COVID-19 infections. We first extracts appropriate features using transfer learning. For this aim, we adopt a pre-trained VGG-16 network introduced in [5]. We also pre-process our dataset to to be in standard size of 224-by-224 for VGG-16. We next use the extract features as the input for different classifiers such as SVM, K-nearest neighbor, and so on compare the performance of the obtained results.

A. Dataset Prepossessing and Analysis

We use two datasets and the results of both will be presented. This is because first, we observed that dataset 1 contains some flawed and mislabeled images, specifically in test dataset, giving rise to a very poor accuracy in classification. Second, we investigated the sensitivity of our classifier to new features and data by considering another dataset. Dataset 1, available in [8], contains 950 X-ray images labeled via three classes of COVID-19, Pneumonia, and No finding (or healthy). Table II presents the dataset split into different sets. Also,

TABLE I: Dataset 1 Information

	COVID-19	**Normal**	**Pneumonia**	**Total**
Training Set	200	280	280	760
Validation Set	25	35	35	95
Test Set	25	35	35	95
Total	250	350	350	950

Dataset 2, available in [2], contains 380 X-ray images labeled via three classes of COVID-19, Pneumonia, and Normal (or healthy). Table I presents the dataset split into different sets.

TABLE II: Dataset 2 Information

	COVID-19	**Normal**	**Pneumonia**	**Total**
Training Set	114	83	107	304 (80%)
Test Set	26	17	33	76 (20%)
Total	140	100	140	380

It is necessary to note that these datasets do not obey a consistent standard or format in its images. This necessitates pre-processing the images to obtain a set of images with the same size and resolution, which may adversely affect the performance and accuracy of final results. To select a uniform size for all the images, we follow the standards dictated by the CNN architectures. For example VGG16's default input-size is 224-by-224 [5].

B. Feature Extraction

As we are dealing with image data, we will be exploring the use of state-of-the-art CNNs in extracting features and thereby classifying the images. In fact, because image data, specifically X-ray images associated with COVID-19, is complex and has large dimension, it is difficult, in not impossible, for human to assign correct features to images. Therefore, we take advantage of CNNs for performing all the operations to find a set of features. CNNs have been known to perform very well on image classification for medical applications [9]. In this project, we consider using VGG16 and VGG19 architectures [5] as their performance, as an example, on a large X-ray dataset was promising in [1] (97.82% accuracy in transfer learning). As mentioned before, we will use a pre-trained network for transfer learning [2] which gives us the features. Also, we may use other architectures, such as GoogLeNet [10] and ResNet [11], in future to compare the results.

While we will be employing the use of CNNs, it is noteworthy that the network can be used for two purposes. first, using the network for an end-to-end classification and second, it can be used to only extract some features out of the image dataset. In the latter case, different machine learning algorithms should be used for performing classification which will be discussed in Section III-C.

C. Machine learning for classification

Having extracted the features by using a pre-trained CNN, we next investigate the performance of different classification algorithm in classifying our dataset.

At this stage, we aim to employ Decision Tree, Softmax, and a combination of Principal Component Analysis (PCA) and Support Vector Machine (SVM) for classification. We investigate the accuracy of these methods in classification and thus diagnosing COVID-19 cases. In what follows we provide reasons for selecting the mentioned algorithms.

Decision Tree learning (selected by Sneha Kawle to be implemented). Decision Tree learning is a widely used and practical method for classification and regression. This inductive learning algorithm is a Supervised classification technique which can train samples and build the decision tree and then use it in the data classification process. Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. Decision Trees based classification is most suited for problems wherein the training dataset consists of attribute-value pairs with a large number of features, the target function has discrete output values and the dataset might be prone to errors or contain missing attribute values. With respect to the Chest X-ray dataset, we can observe that it meets all these conditions. The training dataset is divided into three categories – COVID-19, No-Findings, Pneumonia. Post feature extraction, we have a large dataset with each instance of dimensions $7 \times 7 \times 512$, along with the three categories as the target labels $(0, 1, 2)$ which are discrete output values. We should also consider that the dataset of X-ray images could potentially contain errors caused by manual intervention or technical glitches. As such, Decision Tree classification seems to be appropriately suited for the given problem of image classification.

VGG16 and VGG19 with Softmax (selected by Sivankit Bhanot to be implemented). Convolutional Neural Networks were selected as the approach to extracting features and then applying classification to them as these have had a good run in recent years with image classification challenges. While they have been around, their popularity kicked off with the AlexNet paper in 2012 [12]. This architecture was presented as one of the contenders for the 2012 ILSVRC Competition on the ImageNet dataset, which involved classifying images into 1000 different categories. Since then, there have been numerous successors to the AlexNet architecture. Among them are also VGG 16 and VGG 19. These are both networks designed by researchers in the Visual Geometry Group at University of Oxford. They were both contenders in the 2014 ILSVRC Competition. The pre-trained and un-trained version of both of these networks are available in Keras. We have used them as both have been known to perform well in image classification problems [5] [1]. To illustrate the importance of transfer learning, we tried to first train the complete VGG 16 and VGG 19 neural networks from scratch using just our dataset (consisting of less than 1000 instances). We quickly observed in both cases that within less than 10 epochs, the accuracy of the training set and validation set settled to low values. These are shown in the table below:

TABLE III: Accuracy Evaluation of CNN models without Transfer Learning

Accuracy	**VGG16**	**VGG19**
Train	0.6229	0.6229
Validation	0.2558	0.2558

Large convolutional neural networks are designed to be trained on larger datasets and the dataset we are working with is clearly not large enough to train this network from scratch. Therefore, we pursued transfer learning.

SVM as well as NN classifier with ReLU activation function (selected by Mohammad Bahrami to be implemented). We consider SVM as one candidate classifier in this project of some of the its inherent features that we speculate they might helpful for classifying (diagnosing) COVID-19 cases. First, being effective in dealing with a large feature space [13]. In fact, the complexity and difficulty of diagnosing COVID-19 from X-ray images may cause having a large feature space that are required to distinguish between COVID-19 and other illness which causes very similar symptoms. Second, the kernel trick and its functionality [14]. The kernel tick allows us to transform the problem into a higher dimensional feature space in which data is linearly separable and thus classification might be performed with a higher accuracy. Also, as another algorithm we replace the last fully-connected layers of VGG16 with two fully-connected layers of our choice having ReLU activation function as classifier. The reason for this choice is investigating the effect of different NN classifiers and activation functions in the accuracy of classification.

D. Implementation Process

1) Feature extraction using transfer learning: For extracting features from the data, we have first used the VGG16 network [5]. Using this network a feature extraction function has been designed and used. This function can be used in the future with any network of our choice.

Having a small dataset, we will be using transfer learning to get accurate results. This involves using a pre-trained CNN (or training it on another dataset before using it to extract features on our dataset). Keras has built in functions for the VGG16 and VGG19 networks. One of the arguments is that of weights. The weights that would be obtained from training the network on ImageNet (for which both these networks were initially used) can be passed into the function by assigning a simple value to the "weights" parameter. This way, we are able to get the pre-trained convolutional neural network. As we are only concerned with feature extraction, we do not need the fully connected layers.

It is necessary to note that VGG network has the default input size of 224-by224 [5], and in order to comply with this requirement, we pre-processed our data which includes scaling. Table VII shows the number of parameters in each layer of VGG16 architectures associated with features we extract from them. In both cases we end up with an array of instances each having dimensions $7 \times 7 \times 512$ as features. Figure 1 depicts some of the extracted features.

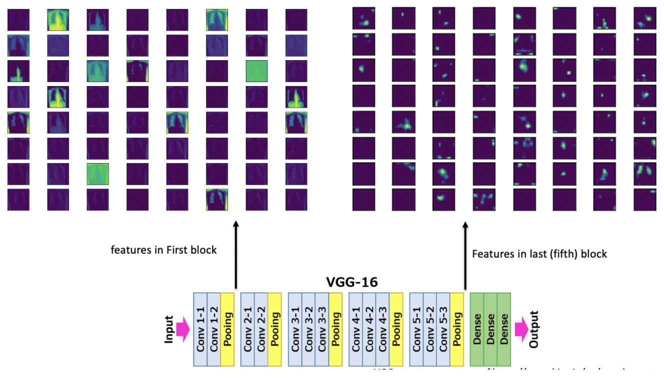


Fig. 1: Features extracted from the first and the fifth (last) block of VGG16 CNN. VGG16 architecture is pre-trained by IMAGENET dataset, and the corresponding weights were applied, as filters, to a randomly selected X-ray image of dataset 1 to show some of the features that are extracted for classification.

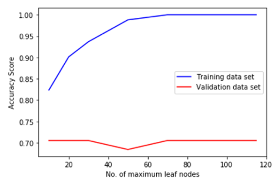
2) Decision Tree with GridSearch: Post feature extraction process, we obtain three matrices of containing features of dimensions 7x7x512, for the training, validation and testing set of images. In order to train a decision tree model, we first need to flatten these matrices into 1-Dimensional arrays.

In order to obtain the best hyper-parameters, a gridsearch on decision tree model was run with Maximum leaf nodes set to [10,20,30,50,70,90,100,110,115]. Running the first grid search, we can observe the results in table IV with maximum leaf nodes of 20 obtaining the highest rank in terms of mean test scores.

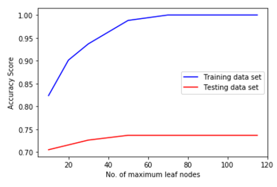
TABLE IV: DataSet 1: GridSearch 1 Results

Max_leaf_nodes	**Mean_test_score**	**Rank_test_score**
10	0.671053	3
20	0.690789	1
30	0.675	2
50	0.669737	4
70	0.669737	4
90	0.669737	4
100	0.669737	4
110	0.669737	4
115	0.669737	4

The accuracy measures on the validation and testing datasets were plotted against the different values of maximum leaf nodes, as shown in the Figure 2. We see that the the accuracy scores start to flatten out with higher number of leaf nodes.



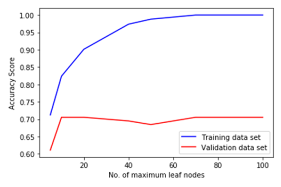
(a) Validation data Accuracy score



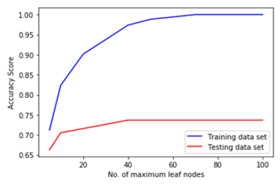
(b) Testing data Accuracy score

Fig. 2: GridSearch 1 with Accuracy Measures over Max Leaf nodes on dataset 1.

The first grid search results generate the best parameter of 20 max leaf nodes. We also ran a second grid search with max leaf nodes set to [None,5,10,20,40,50,70,90,100]. The second grid search results also generated 20 as the best parameter for max leaf nodes. The accuracy scores obtained for the second grid search are as shown in Figure 3.



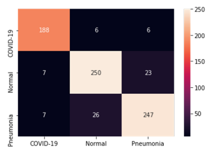
(a) Validation data Accuracy score



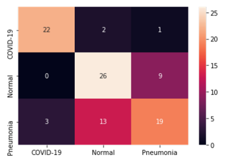
(b) Testing data Accuracy score

Fig. 3: GridSearch 2 with Accuracy Measures over Max Leaf nodes on dataset 1.

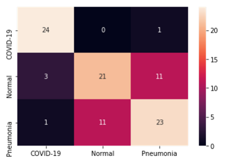
We then plotted the confusion matrices and generated the classification reports for both, the validation and testing datasets, in order to evaluate the performance of the decision tree. From Figures 4 and 5 we observe an accuracy score of 90% on the training dataset and approximately 71% on the validation and testing datasets.



(a) Training Confusion Matrix

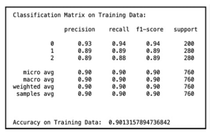


(b) Validation Confusion Matrix



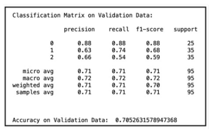
(c) Testing Confusion Matrix

Fig. 4: Confusion Matrices of GridSearch 1 and 2 with 20 leaf nodes on dataset 1.



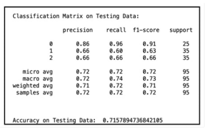
```
Classification Matrix on Training Data:

              precision    recall  f1-score   support

           0       0.93      0.94      0.94       200
           1       0.89      0.89      0.89       280
           2       0.89      0.88      0.89       280

   micro avg       0.90      0.90      0.90       760
   macro avg       0.90      0.90      0.90       760
weighted avg       0.90      0.90      0.90       760
 samples avg       0.90      0.90      0.90       760


Accuracy on Training Data:  0.9013157894736842
```

(a) Training Classification Report



```
Classification Matrix on Validation Data:

              precision    recall  f1-score   support

           0       0.88      0.88      0.88        25
           1       0.63      0.74      0.68        35
           2       0.66      0.54      0.59        35

   micro avg       0.71      0.71      0.71        95
   macro avg       0.72      0.72      0.72        95
weighted avg       0.71      0.71      0.70        95
 samples avg       0.71      0.71      0.71        95


Accuracy on Validation Data:  0.7052631578947368
```

(b) Validation Classification Report



```
Classification Matrix on Testing Data:

              precision    recall  f1-score   support

           0       0.86      0.96      0.91        25
           1       0.66      0.60      0.63        35
           2       0.66      0.66      0.66        35

   micro avg       0.72      0.72      0.72        95
   macro avg       0.72      0.74      0.73        95
weighted avg       0.71      0.72      0.71        95
 samples avg       0.72      0.72      0.72        95


Accuracy on Testing Data:  0.7157894736842105
```

(c) Test Classification Report

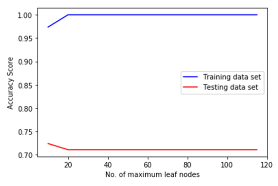
Fig. 5: Classification Reports of GridSearch 1 and 2 with 20 leaf nodes on dataset 1.

Given the testing results obtained from the other algorithms, we evaluated a second dataset of chest x-ray images and repeated the process. A GridSearch on decision tree model was run with Maximum leaf nodes set to [10,20,30,50,70,90,100,110,115]. Running the first grid search, we can observe the results in Table V with maximum leaf nodes of 20 obtaining the highest rank in terms of mean test scores.

TABLE V: DataSet 2: GridSearch 1 Results

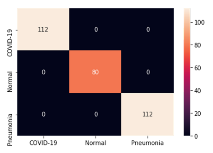
Max_leaf_nodes	Mean_test_score	Rank_test_score
10	0.848689	9
20	0.851967	1
30	0.851967	1
50	0.851967	1
70	0.851967	1
90	0.851967	1
100	0.851967	1
110	0.851967	1
115	0.851967	1

The plot of the accuracy scores against the leaf nodes for the testing dataset can be observed from the Figure 6. Confusion matrices and Classification reports were generated for the decision tree trained with 20 as the max leaf nodes. From the Figures 7 and 8, we observe an accuracy score of 100% on the training dataset whereas the testing dataset generated an accuracy score of 71%.

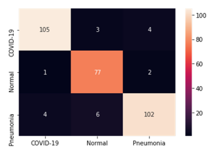


(a) Testing data Accuracy score

Fig. 6: GridSearch 1 with Accuracy Measures over Max Leaf nodes on dataset 2.



(a) Training Confusion Matrix

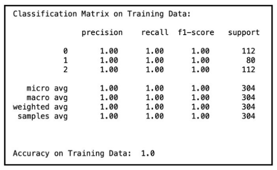


(b) Testing Confusion Matrix

Fig. 7: Confusion Matrices of GridSearch 1 with 20 leaf nodes on dataset 2.

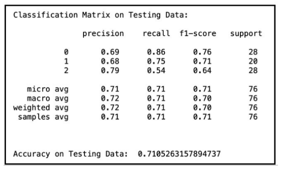
A second gridsearch was run on the decision tree with max leaf nodes set to [None,2,4,6,8,10,12,14,16] on dataset 2. This generated best parameter of 6 as max leaf nodes. The results of the second gridsearch are as shown in Table VI.

Confusion matrices and Classification reports were generated for the decision tree trained with 6 as the max leaf nodes. From the Figures 9 and 10, we observe an accuracy score of 93% on the training dataset whereas the testing dataset generated an accuracy score of 72%.



```
Classification Matrix on Training Data:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       112
           1       1.00      1.00      1.00        80
           2       1.00      1.00      1.00       112

   micro avg       1.00      1.00      1.00       304
   macro avg       1.00      1.00      1.00       304
weighted avg       1.00      1.00      1.00       304
 samples avg       1.00      1.00      1.00       304


Accuracy on Training Data:  1.0
```

(a) Training Classification Report



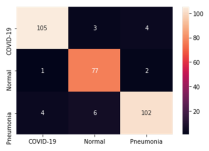
```
Classification Matrix on Testing Data:

              precision    recall  f1-score   support

           0       0.69      0.86      0.76        28
           1       0.68      0.75      0.71        20
           2       0.79      0.54      0.64        28

   micro avg       0.71      0.71      0.71        76
   macro avg       0.72      0.71      0.70        76
weighted avg       0.72      0.71      0.70        76
 samples avg       0.71      0.71      0.71        76


Accuracy on Testing Data:  0.7105263157894737
```

(b) Testing Classification Report

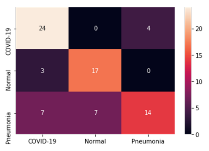
Fig. 8: Classification Reports of GridSearch 1 with 20 leaf nodes on dataset 2.

TABLE VI: DataSet 2: GridSearch 2 Results

Max_leaf_nodes	Mean_test_score	Rank_test_score
None	0.83224	12
2	0.351639	13
4	0.852077	4
6	0.868525	1
8	0.861913	2
10	0.848689	11
12	0.855246	3
14	0.851967	5
16	0.851967	5

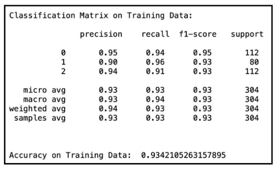


(a) Training Confusion Matrix



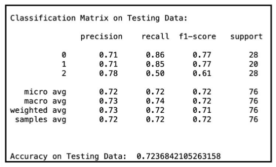
(b) Testing Confusion Matrix

Fig. 9: Confusion Matrices of GridSearch 2 with 6 leaf nodes on dataset 2.



```
Classification Matrix on Training Data:

              precision    recall  f1-score   support

           0       0.95      0.94      0.95       112
           1       0.90      0.96      0.93        80
           2       0.94      0.91      0.93       112

   micro avg       0.93      0.93      0.93       304
   macro avg       0.93      0.94      0.93       304
weighted avg       0.94      0.93      0.93       304
 samples avg       0.93      0.93      0.93       304


Accuracy on Training Data:  0.9342105263157895
```

(a) Training Classification Report



```
Classification Matrix on Testing Data:

              precision    recall  f1-score   support

           0       0.71      0.86      0.77        28
           1       0.71      0.85      0.77        20
           2       0.78      0.50      0.61        28

   micro avg       0.72      0.72      0.72        76
   macro avg       0.73      0.74      0.72        76
weighted avg       0.73      0.72      0.71        76
 samples avg       0.72      0.72      0.72        76


Accuracy on Testing Data:  0.7236842105263158
```

(b) Testing Classification Report

Fig. 10: Classification Reports of GridSearch 2 with 6 leaf nodes on dataset 2.

From the results obtained, we can observe that decision tree with 20 leaf nodes has the best performance at around 71% accuracy. Given the constraints on computation power, we were only able to run the grid search on one hyperparameter i.e. max leaf nodes. We would be able to obtain a higher accuracy score by running the gridsearch on other hyperparameters such as sample size, max depth, criterion etc. Increasing the size of the dataset could also contribute towards improving the accuracy scores over the testing dataset.

3) VGG16 and VGG19 with Softmax: As mentioned earlier, VGG16 was used to extract features, prior to applying different classification algorithms. This is the transfer learning approach. When using transfer learning, we are concerned at first with the convolutional layers or just the base of the network. For these layers, we had the option to use weights from the pre-trained network. These are weights from the network being pre-trained on the ImageNet dataset.

After attaining the features, some fully connected layers needed to be applied. These are the layers that are trained using our dataset. Since we are dealing with multi-category (more than 2) classification, softmax regression is applied by using the softmax activation function in the last fully connected layer.

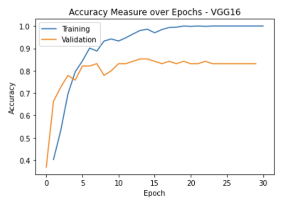
VGG16 consists of 5 convolutional blocks, with the first two blocks consisting of two convolutional layers each, and the latter three blocks containing three layers each. Each convolutional block is followed by a max-pooling process. The VGG 19 consists of 5 convolutional blocks also with the first two blocks containing two convolutional layers each, but the last three blocks have 4 layers each. Again all the blocks are followed by a max-pooling process. For both networks, the ending layers consist of flattening the feature matrix and three fully connected (dense) layers with the last one having a softmax activation function to apply softmax classification. When performing transfer learning, we are initially concerned with just training the convolutional blocks on a larger dataset like ImageNet. The dense layers are later on added to the network and then the network is trained on the target dataset. In our case, we have used the same dimensions and parameters for the last four layers (consisting of a flattening layer and three dense layers). The one difference applied is to the last prediction dense layer where the output shape is made (None, 3) as we are concerned with classifying among three classes rather than 1000. The summary showing all the layers and the blocks are displayed in Tables VII and VIII.

TABLE VII: VGG16 Architecture

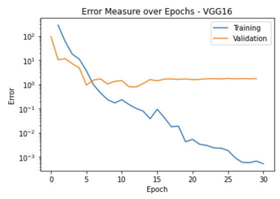
Layer (type)	Output Shape of each instance
input_1 (InputLayer)	$224 \times 224 \times 3$
block1_conv1 (Conv2D)	$224 \times 224 \times 64$
block1_conv2 (Conv2D)	$224 \times 224 \times 64$
block1_pool (MaxPooling2D)	$112 \times 112 \times 64$
block2_conv1 (Conv2D)	$112 \times 112 \times 128$
block2_conv2 (Conv2D)	$112 \times 112 \times 128$
block2_pool (MaxPooling2D)	$56 \times 56 \times 128$
block3_conv1 (Conv2D)	$56 \times 56 \times 256$
block3_conv2 (Conv2D)	$56 \times 56 \times 256$
block3_conv3 (Conv2D)	$56 \times 56 \times 256$
block3_pool (MaxPooling2D)	$28 \times 28 \times 256$
block4_conv1 (Conv2D)	$28 \times 28 \times 512$
block4_conv2 (Conv2D)	$28 \times 28 \times 512$
block4_conv3 (Conv2D)	$28 \times 28 \times 512$
block4_pool (MaxPooling2D)	$14 \times 14 \times 512$
block5_conv1 (Conv2D)	$14 \times 14 \times 512$
block5_conv2 (Conv2D)	$14 \times 14 \times 512$
block5_conv3 (Conv2D)	$14 \times 14 \times 512$
block5_pool (MaxPooling2D)	$7 \times 7 \times 512$
flatten (Flatten)	1×25088
fc1 (Dense)	1×4096
fc2 (Dense)	1×4096
predictions (Dense)	1×3

The neural network with the structure described above was trained on our initial dataset for a total of 30 epochs and a batch size of 30. The accuracy and error measure on the training and validation sets was recorded and plotted. These are shown in Figure 11.

The performance of the pre-trained VGG16 on the training, validation and test sets is also illustrated by the confusion matrices in Figure 12. Figure 12a shows how all the values lie within the primary diagonal of the matrix signifying that

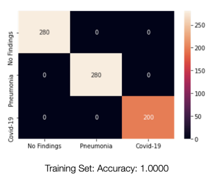


(a) VGG16 Training and Validation Accuracy

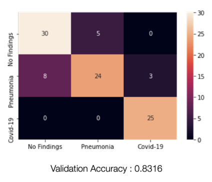


(b) VGG16 Training and Validation Error

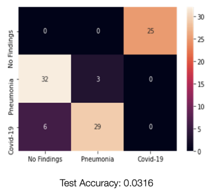
Fig. 11: Accuracy and Error Measures over 30 epochs for VGG16 on dataset 1.



(a) Training set



(b) Validation set



(c) Testing set

Fig. 12: Confusion Matrices of VGG16 on dataset 1.

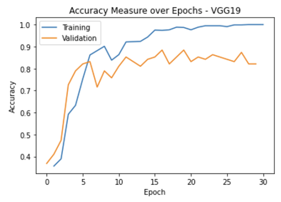
all instance have been classified correctly as their predicted and actual classes are the same. While we see in Figure 14b that there are some misclassified instances, the majority of the values still lie on the primary diagonal. This is contrasted with the matrix for the test set in Figure 14c where there are barely any values in this primary diagonal showing the very low accuracy of just above three percent. Before moving to judge the dataset, we tried the same with the VGG19 network.

Carrying out similar process for VGG19, we again ran 30 epochs with a batch size of 50 and obtained accuracy measure of 1.0000 on training set and 0.8211 on validation set. The plots are displayed in Figure 13. While the accuracy does not seem to converge even after 30 epochs, we observe that the validation set error again bottoms out at the 15th epoch. Figure 14 shows the confusion matrices for VGG19 performance on the three subsets.

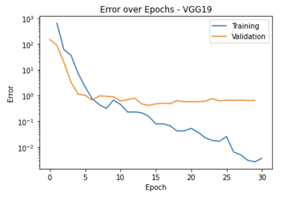
We observed very poor performance of both VGG16 and VGG19 on the test subset of our initial dataset. This raised concern regarding the legitimacy of the test set. In order to rule out any issues with our network architecture and feature extraction method, we decided to test the performance on another similar data-set which also has thoracic X-ray images to be classified into three categories.

TABLE VIII: VGG19 Architecture

Layer (type)	Output Shape of each instance
input_2 (InputLayer)	$224 \times 224 \times 3$
block1_conv1 (Conv2D)	$224 \times 224 \times 64$
block1_conv2 (Conv2D)	$224 \times 224 \times 64$
block1_pool (MaxPooling2D)	$112 \times 112 \times 64$
block2_conv1 (Conv2D)	$112 \times 112 \times 128$
block2_conv2 (Conv2D)	$112 \times 112 \times 128$
block2_pool (MaxPooling2D)	$56 \times 56 \times 128$
block3_conv1 (Conv2D)	$56 \times 56 \times 256$
block3_conv2 (Conv2D)	$56 \times 56 \times 256$
block3_conv3 (Conv2D)	$56 \times 56 \times 256$
block3_conv4 (Conv2D)	$56 \times 56 \times 256$
block3_pool (MaxPooling2D)	$28 \times 28 \times 256$
block4_conv1 (Conv2D)	$28 \times 28 \times 512$
block4_conv2 (Conv2D)	$28 \times 28 \times 512$
block4_conv3 (Conv2D)	$28 \times 28 \times 512$
block4_conv4 (Conv2D)	$28 \times 28 \times 512$
block4_pool (MaxPooling2D)	$14 \times 14 \times 512$
block5_conv1 (Conv2D)	$14 \times 14 \times 512$
block5_conv2 (Conv2D)	$14 \times 14 \times 512$
block5_conv3 (Conv2D)	$14 \times 14 \times 512$
block5_conv4 (Conv2D)	$14 \times 14 \times 512$
block5_pool (MaxPooling2D)	$7 \times 7 \times 512$
flatten (Flatten)	1×25088
fc1 (Dense)	1×4096
fc2 (Dense)	1×4096
predictions (Dense)	1×3



(a) VGG19 Training and Validation Accuracy



(b) VGG19 Training and Validation Error

Fig. 13: Accuracy and Error Measures over 30 epochs for VGG19 on dataset 1.

Prior to training our network on this new dataset, we also attempted to swap the initial test and validation subsets in dataset 1 and try to see how a network trains by using the original test subset as a validation subset. We observed that in doing so the new 'validation' subset converges to an accuracy of just above 2 percent in less than 10 epochs, leading us to believe that the test subset in dataset 1 is much different to the validation and training subsets. Table IX shows the overall performance of the two networks on our first dataset.

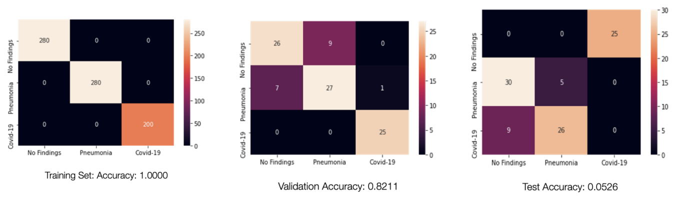
TABLE IX: Accuracy Evaluation of CNN models on dataset 1

Accuracy after 50 epochs	**VGG16**	**VGG19**
Train	1.0000	1.0000
Validation	0.8316	0.8211
Test	0.0316	0.0526

Running similar process on dataset 2 yielded us with increased accuracy measures for the test set for both VGG16 and VGG19. These are shown in Table X. The second dataset is smaller and hence a batch size of 30 was used and 20 epochs were run. The accuracy and error plots achieved after training VGG16 with Softmax on the second dataset are shown in Figure 15.

We see from the confusion matrices for the VGG16 performance on dataset 2 that the training subset again has no mis-classified instances. In the test subset, we have 10 mis-classified instances. As Figure 16b shows there are 9 instances that are actually Pneumonia cases but have been classified as Normal. There is also 1 instance of Normal class that has been predicted as belonging to Pneumonia class. In the case of COVID-19 we do not have mis-classified instances. All instances actually belonging to COVID-19 category have been predicted as such.

If we compare this performance with that of the VGG19 network, we see that there are some mis-classifications in



(a) Training set (b) Validation set (c) Test set

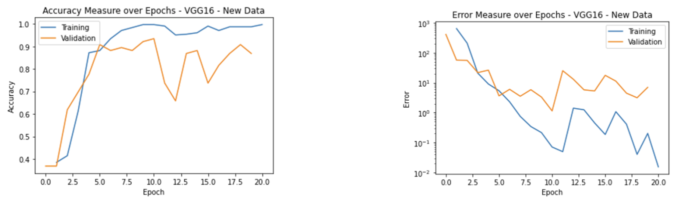
Fig. 14: Confusion Matrices of VGG19 on dataset 1.

relation to the COVID-19 category as well, as shown in Figure 18b. There are 4 instances in the test sub set that have been mis-classified into the COVID-19 category. 2 are instances that actually belong to Normal category and 2 instances that actually belong to Pneumonia category. These four being classified incorrectly as COVID-19 means that we have four cases of false positives. We have no cases of false negatives.

When dealing with diagnosis of an illness like COVID-19 and in the case of an emergent pandemic where the aim is to restrain and prevent spread of the disease and virus, we want to minimize and have no cases of false negatives. False positives are acceptable as they only lead to a more cautious approach, but with false negatives we will not be able to keep track of all those that are actually infected and correctly diagnosed.

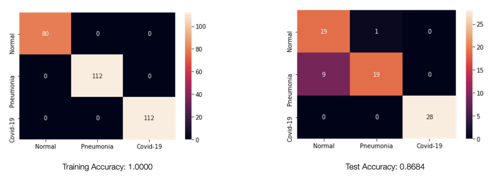
TABLE X: Accuracy Evaluation of CNN models on dataset 2

Accuracy after 20 epochs	VGG16	VGG19
Train	1.0000	1.0000
Test	0.8684	0.8816



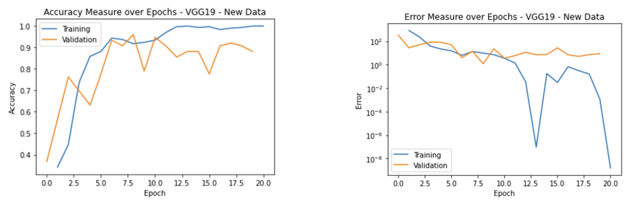
(a) Training and Test Accuracy (b) Training and Test Error

Fig. 15: Accuracy and Error Measures over 20 epochs for VGG16 on dataset 2.



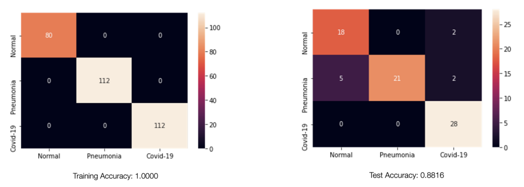
(a) Training set (b) Testing set

Fig. 16: Confusion Matrices of VGG16 on dataset 2.



(a) Training and Test Accuracy (b) Training and Test Error

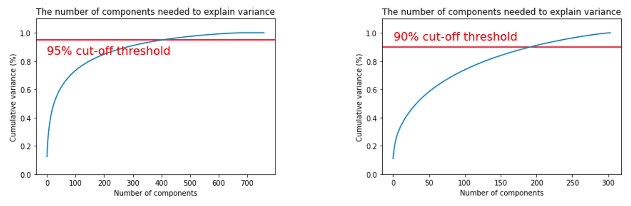
Fig. 17: Accuracy and Error Measures over 20 epochs for VGG19 on dataset 2.



(a) Training set (b) Testing set

Fig. 18: Confusion Matrices of VGG19 on dataset 2.

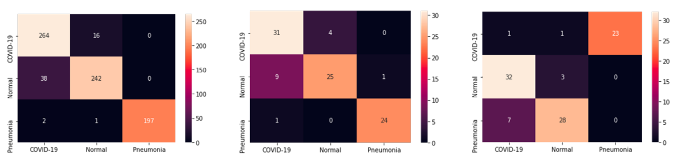
4) PCA and SVM: a grid search with 3-fold cross validation: The reasons for the candidacy of SVM as a classifier in this problem mentioned in Section III-C. Here, we first address the curse of dimensionality using PCA analysis. Results in Figure 19 shows the cumulative variance analysis performed on both datasets to determine the number of principal components that are sufficient to represent 95% and 90% features of dataset 1 and dataset 2, respectively.



(a) dataset 1 (b) dataset 2

Fig. 19: PCA results for demensionality reduction.

We next use these components in SVM classification. We perform a grid search over a set of SVM kernels (*linear and polynomial*) with their respective optimization parameters to find the best classifier. Also, the grid search uses a 3-fold cross validation which help avoid over-fitting in cases with a small dataset. In both cases of dataset 1 and 2 the best classier had a linear kernel. Figure 20 presents the accuracy of classification of the dataset 1 in confusion matrices form.



(a) Training set = 97% (b) Validation set = 84% (c) Testing set = 4%

Fig. 20: Confusion matrices associated with results for a SVM classifier applied to dataset 1.

As results, specifically in Figure 23d, show the classifiers do not perform good enough for new data although accuracy in validation set rules out the possibility of over-fitting as a contributing factor of low accuracy in classifying new data. We next investigate the problem in a different dataset, dataset 2, and elaborate on underlying reasons for low accuracy in Sections IV and VI.

Figure 21 presents the accuracy of classification performed on the dataset 2 in confusion matrices form. The very good accuracy on both training and testing sets suggests a high sensitivity of classifier to features extracted from this specific dataset. This implies we would expect low accuracy in classifying new data that contains also new features other than what have been extracted before. This problem arises because of having an small set and not having goof feature extraction.

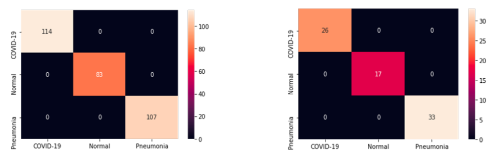
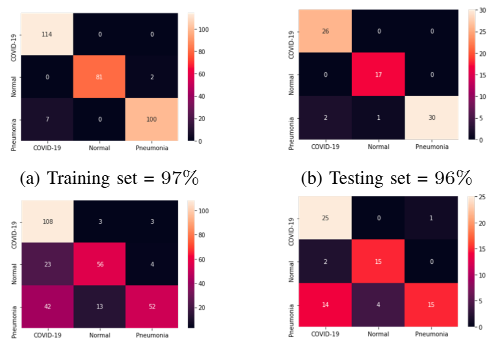

(a) Training set = 100% (b) Testing set = 100%

Fig. 21: Confusion matrices associated with results for a SVM classifier applied to dataset 2.

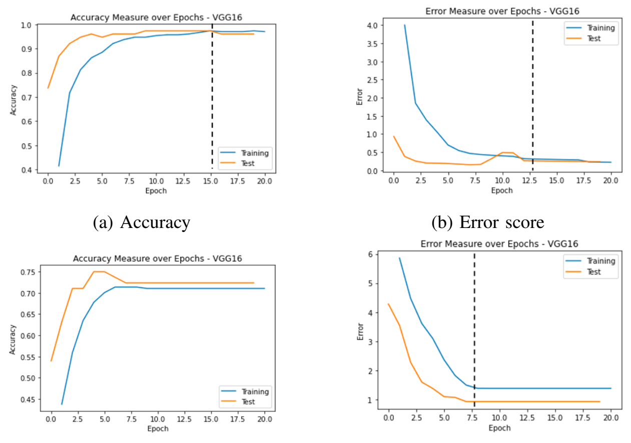
PCA and a two-layer neural network classifier with ReLU activation function: as another algorithm for classification, we replaced the last fully-connected layers of VGG16 with two fully-connected layers (with 512 and 3 neurons) of our choice having ReLU activation as classifier. The reason for the last 3 neurons is having a three classes of *COVID-19*, *Normal*, and *Pneumonia*.


(a) Training set = 97% (b) Testing set = 96%

(c) Training set = 71% (d) Testing set = 72%

Fig. 22: Confusion matrices associated with results for a 2-layer NN classifier applied to dataset 2. Cases a) and b) are results of using *adam* optimizer that is a stochastic gradient descent method, and cases c) and d) are results of using *rmsprop* optimizer that is uses momentum approach to maintain the gradient sufficiently large.

Figures 22 and 23 presents confusion matrices and error scores per epochs for 2-layer NN classifier. It shows the selected classifier a with ReLU activation function has a better performance than that of VGG16 and VGG19 with softmax activation function. Moreover, the momentum approach does not improve the the accuracy of classification that suggests we need to tune its corresponding parameter.


(a) Accuracy (b) Error score

(c) Accuracy (d) Error score

Fig. 23: Accuracy and Error scores for both training and testing sets obtained from a 2-layer NN classifier that was applied to dataset 2. Cases a) and b) are results of using *adam* optimizer that is a stochastic gradient descent method, and cases c) and d) are results of using *rmsprop* optimizer that is uses momentum approach to maintain the gradient sufficiently large.

IV. Comparison

In this section we compare our results and draw some conclusions of them. Table XI presents a summary of our results for dataset 1 and some results from literature. Moreover, Table XII presents a summary of our results for dataset 2 and results from literature.

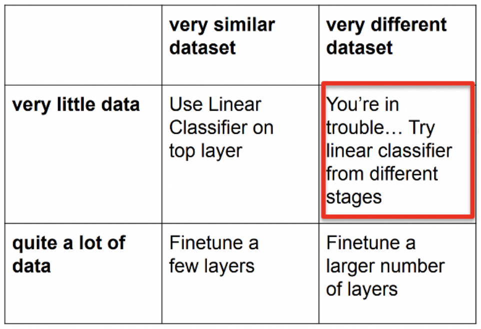
TABLE XI: **Comparison of Algorithms Performance on Dataset 1:** For the Decision Tree, MLN refers to the 'max_leaf_nodes' hyperparameter. In the PCA methods the features refer to 420 out of 760 principal components. The final two entries show performance of related work in industry and academia on the dataset. The multi-layer NN consists of two hidden layers of 128 and 16 neurons.

	Accuracy(%)			
Classifier	**Train**	**Val**	**Test**	**Features**
Decision Tree (MLN=20)	90	71	71	25088
VGG16 + Softmax	100	83	3	25088
VGG19 + Softmax	100	82	5	25088
PCA + SVM	92	84	4	420/760
Mutli-layer NN [2]	-	94	-	-
DarkCovidNet [3]	-	98	-	-

TABLE XII: **Comparison of Algorithms Performance on Dataset 2:** In our PCA methods the features refer to 200 out of 304 principal components. Also, for prior work named Multi-Layer NN, the features refer to using 200 out of 252 general features.

	Accuracy(%)		
Classifier	**Train**	**Test**	**Features**
Decision Tree (MLN=20)	100	71	25088
Decision Tree (MLN=6)	93	72	25088
VGG16 + Softmax	100	87	25088
VGG19 + Softmax	100	88	25088
PCA + SVM	100	100	200/304
PCA + 2 FC NN ('adam')	97	96	200/304
PCA + 2 FC NN ('rmsprop')	71	72	200/304
Multi-layer NN [2]	-	94	200/252
DarkCovidNet [3]	-	98	-

As results in Table XI suggest the classifiers do not perform good enough for new inquires. Surprisingly enough, accuracy of validation set rules out the over-fitting as a contributing factor to low accuracy in classifying new data. The only exception is the Decision tree method that shows robustness with respect to this issue. The first reason is that we observed that dataset 1 is not standard because it contains some copied and mislabeled images in all training, validation and testing sets. To address this problem, we used a similar but different dataset whose results presented in Table XII. More importantly, We speculate the problem of low accuracy is due to over-fitting with respect to the extracted features. Indeed, given a small dataset, there is no guarantee that the extracted features captures all necessary features that are required for a good generalization of the classifier. Figure 24 accounts for this problem in details. In fact, in a transfer learning scenario, that a pre-trained CNN is used for feature extraction, two main contributing factors to achieve a good classifier are 1) the similarity of the dataset that has been used to train the CNN and 2) richness of dataset that is used for training the classifier i.e. the last layers of CNN. In our cases, both of theses factors are in a low level which arises low accuracy classification for new data. It is necessary to note that in this case, common methods such as cross-validation and data augmentation are not able to significantly improve the low accuracy problem.



	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Fig. 24: Size-similarity matrix. Courtesy of Stanford course CS231n, Lec. 7

Furthermore, results of dataset 2 suggest that using a ReLU activation function can effectively address the diffusion of gradient which in turn provides a higher accuracy score than softmax activation function in VGG16 and VGG19.

V. Future work

We have used transfer learning to extract features and then classify the thoracic X-ray images in two different datasets. The neural networks used to extract features were trained on ImageNet which consists of a wide variety of images. However, as depicted in Figure 24, in our problem that we are concerned with classifying X-ray images it could be more beneficial to us to use a network pre-trained on X-ray images. This method would be domain extension transfer learning [7]. For this, we would also require using a large X-ray image dataset to train the CNN on. One such dataset is provided by National Institute of Health. It would also be worth exploring if the weights of the pre-trained network could be attained. This would save time in not having to train the network on the large dataset and simply use the weights as we have done with ImageNet. The weights used from a neural network previously trained on a large X-ray image dataset would make the network architecture accustomed to picking out those features which are more relevant to X-ray images.

As a relevant approach suggested in Figure 24, a different pre-trained CNN architecture such as googLeNet that provides multiple outputs in different stages can be used as feature extractors. This improves the accuracy because we are able to analyze the accuracy of classification obtained by using features of different layers. However, training a CNN with large datasets such as ImageNet requires a quite amount of time and a good hardware.

Promising results of using ReLU activation function with different dense layers of that in VGG16, presented in Table XII, suggest, for future iterations, trying a varied number of dense layers followed by different activation functions at the end of the pre-trained CNN architecture as different classifiers. These serve as hyperparameters to the network architecture and could influence the performance. A larger selection of popular architectures could also be tried to perform feature extraction and classification.

Furthermore, it would strengthen the validity of our methods by cross-examining their performance on the two datasets. The methods used to classify the second dataset, for example, could be evaluated on the first dataset (and vice versa), as the first dataset is unseen by the methods and will act as new data. This would help in evaluating the variance of performance of the methods used, and give an idea as to where they stand in the bias-variance trade-off.

The Decision Tree algorithm was trained with a GridSearch being performed only on one hyper-parameter of Maximum leaf nodes. It would be good to explore other hyper-parameters as well in order to arrive at the optimum combination which would further improve the performance of the algorithm in terms of classification accuracy. Other hyper-parameters which could impact the performance would be Criterion (Gini or Entropy), Maximum depth, Minimum Samples split, Minimum samples leaf, Maximum features and so on. The objective is to obtain a decision tree trained with the optimum hyper-parameters obtained via GridSearch.

The above tuning method is computationally expensive as we would need to define the range of values to use for the hyper-parameters. We could also explore other methods like Random Search and Bayesian Optimization which provide smarter, less-expensive tuning methods.

VI. Conclusions

While we observed a generally good performance on training and validation sets from the first dataset, the performance for most algorithms on the test set of this dataset was very poor. Barring the decision tree algorithm, we observed an accuracy lower than 10%. Switching the test and validation sub sets from this first dataset also did not seem to show much difference as the 'new' validation sub set still had very low accuracy levels as the CNN (VGG19) with dense layers was trained again. This highlighted that the test subset of dataset 1 was a to some degree different from the training and validation subsets.

For dataset 2 we were able to attain better accuracy on the test subsets for all algorithms and in some cases these were comparable to prior results in [2] [3]. For the decision tree algorithm, the accuracy of the training data increased for the second dataset from 90% for dataset 1 to 100% for dataset 2. However, the test sub set showed an accuracy around 71% for both datasets.

When using VGG16 and VGG19 with softmax on the first dataset, we could see from the accuracy and error plots that usually the validation error seemed to settle after or around the fifteenth epoch. In the case of the VGG19, the accuracy was still fluctuating quite a bit but the error seemed to bottom out. However, using both these convolutional nets on the second dataset, there were no signs of a settling accuracy or error level on either the training or validation set, even after 20 epochs. This also means that perhaps more epochs would need to be run. Overall the performance on the test set was better on the second dataset for both CNNs(with softmax as the last layer). For VGG16, the validation performance was 83% and test performance was 3% on the first dataset and the test performance was 87% on the second dataset. For VGG 19, the performance on the first dataset consisted of 82% on validation set and 5% on test set, and for the second dataset the performance on test set was 88%. Therefore, we observed an increase in accuracy performance with both CNNs.

Principle component analysis was adopted and used effectively for dimensionality reduction with the SVM algorithm. The results suggest that the extracted features for dataset 2 are completely linearly separable.

We observed from our work on datasets 1 and 2 that a small dataset does not allow us to generalize the obtained classifier which perform effectively on new data. Also, the results in the literature is consistent with this observation. In most of the cases of training classifiers to distinguish COVID-19 from other disease using X-ray images (see Table 4 in [3]), the accuracy score for validation set is more than 80%. Although this is a relatively good accuracy, because of limited amount of available data, there is no guarantee that the extracted features are sufficient to capture those exist in new data. Consequently, having low accuracy score for new data cannot be superficially ruled out.

Supplementary material

We have attached the code for pre-processing and feature extraction.

References

[1] I. D. Apostolopoulos and T. A. Mpesiana, "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks," *Physical and Engineering Sciences in Medicine*, p. 1, 2020.

[2] A. Z. Khuzani, M. Heidari, and S. A. Shariati, "Covid-classifier: An automated machine learning model to assist in the diagnosis of covid-19 infection in chest x-ray images," *medRxiv*, 2020.

[3] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. R. Acharya, "Automated detection of covid-19 cases using deep neural networks with x-ray images," *Computers in Biology and Medicine*, p. 103792, 2020.

[4] A. T. Sahlol, D. Yousri, A. A. Ewees, M. A. Al-Qaness, R. Damasevicius, and M. Abd Elaziz, "Covid-19 image classification using deep features and fractional-order marine predators algorithm," *Scientific Reports*, vol. 10, no. 1, pp. 1–15, 2020.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[7] S. Basu and S. Mitra, "Deep learning for screening covid-19 using chest x-ray images," *arXiv preprint arXiv:2004.10507*, 2020.

[8] P. Kumar. Covid x-rays dataset. [Online]. Available: https://www.kaggle.com/pranjallk1995/covid-xrays

[9] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE, 2014, pp. 844–848.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[13] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: Fast feature extraction and svm training," in *CVPR 2011*. IEEE, 2011, pp. 1689–1696.

[14] A. F. Agarap, "An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification," *arXiv preprint arXiv:1712.03541*, 2017.