Düzce University
Journal of Science & Technology

Research Article

COVID-19 Prediction from Chest X-Ray Images using Transfer Learning

Kaan BIÇAKCI [a] , Volkan TUNALI [a,*]

[a] *Department of Software Engineering, Faculty of Engineering and Natural Sciences, Maltepe University, Istanbul, TURKEY*
** Corresponding author's e-mail address:* volkan.tunali@gmail.com
DOI: 10.29130/dubited.878779

ABSTRACT

The COVID-19 pandemic has been affecting our lives in many ways, not only the healthcare systems in the countries but the whole societies worldwide. Meantime, a considerable number of studies have been conducted and lots of medical techniques have been tried to overcome the pandemic. In this work, making use of real-world images, we applied Convolutional Neural Networks to chest X-ray images to predict whether a patient has the COVID-19 virus or not. Initially, we used transfer learning to fine tune a number of pre-trained ResNet, VGG, and Xception models, which are very well-known architectures due to their success in image processing tasks. While the achieved performance with these models was encouraging, we ensembled three models to obtain more accurate and reliable results. Finally, our ensemble model outperformed all other models with an F-Score of 97%.

Keywords: *Chest X-Ray, COVID-19, Viral Pneumonia, Deep Learning, Transfer Learning, Ensemble Learning*

Transfer Öğrenme Kullanarak Göğüs Röntgeni Görüntülerinden COVID-19 Tahmini

ÖZ

COVID-19 salgını, sadece ülkelerdeki sağlık sistemlerini değil, dünya çapında tüm toplumları birçok şekilde etkilemektedir. Bu süreçte, pandeminin üstesinden gelmek için önemli sayıda çalışma yapılmış ve birçok tıbbi teknik denenmiştir. Bu çalışmada, gerçek görüntülerden yararlanarak, bir hastada COVID-19 virüsünün olup olmadığını tahmin etmek için Evrişimsel Sinir Ağlarını göğüs röntgeni görüntülerine uyguladık. Başlangıçta, görüntü işleme alanındaki başarıları nedeniyle çok iyi bilinen mimariler olan bir dizi önceden eğitilmiş ResNet, VGG, ve Xception modellerini elimizdeki probleme uygun olarak yeniden eğitmek üzere Transfer Öğrenme kullandık. Bu modellerle ulaşılan performans tatmin edici olsa da daha isabetli ve güvenilir sonuçlar elde etmek amacıyla üç ayrı modeli bir araya getiren bir topluluk modeli oluşturduk. Son olarak, topluluk modelimiz %97'lik bir F-Skoru ile diğer tüm modellerden daha iyi performans gösterdi.

Anahtar Kelimeler: *Göğüs Röntgeni, COVID-19, Viral Pnömoni, Derin Öğrenme, Transfer Öğrenme, Topluluk Öğrenmesi*

Received: 11/02/2021, Revised: 05/05/2021, Accepted: 12/05/2021

I. INTRODUCTION

As of February 10th, 2021, there are 107,639,175 coronavirus cases which have resulted in the death of 2,358,080 people [1]. Since December 2019, the coronavirus has started to become a major problem worldwide as a pandemic. The COVID-19 pandemic has been affecting our lives in many ways, not only the healthcare systems in the countries but the whole societies worldwide. Meantime, a considerable number of studies have been conducted and lots of medical techniques have been tried to overcome the pandemic. Some studies have particularly focused on the diagnosis of COVID-19 from medical images like X-rays and CT scans.

There are several Deep Learning systems that were developed to identify and detect patients who had COVID-19 using their chest X-ray images. Minaee et al. tested some popular deep learning architectures such as ResNet [2] and DenseNet [3]. Their dataset was imbalanced due to the difficulty in finding X-rays images of COVID-19 patients. Therefore, in order to increase the size of COVID-19 X-rays, they used image augmentation in their study [4].

In a study by Khan et al., a model based on Xception architecture [5] was able to give promising results [6]. They also used ImageNet [7] weights of Xception model, fine tuned last layers, and achieved 90% accuracy in 3-class classification.

In another research, Apostolopoulos et al. used pre-trained models from architectures like MobileNet [8], Xception, and VGG [9] for the detection of COVID-19 using chest X-ray images [10]. They applied these models to both binary and multi-class classification and achieved 95% accuracy at best.

Ozturk et al. built a CNN model for COVID-19 detection using chest X-rays [11]. Their model achieved 87% and 98% accuracies for multi and binary class classification, respectively.

In this study, we applied CNN models to chest X-ray images to predict whether a patient has the COVID-19 virus or not. We used transfer learning to fine tune a number of pre-trained ResNet, VGG, and Xception models, which are very well-known architectures due to their success in image processing tasks.

The studies mentioned above were similar to our study. However, their datasets were generally very small when compared to ours; that is, we trained our models on a relatively larger dataset. Some of the studies consistently reported that their datasets were imbalanced in terms of class distribution having small number of COVID-19 X-ray images while having much greater number of images from normal patients. Our dataset, on the other hand, had a comparatively balanced class distribution. Additionally, some of the studies applied image augmentation techniques to increase the size and variety of their datasets as we also took a similar approach. In our study, we also used ensemble learning to enhance the classification accuracy. Different from the previous studies, we were able to get better results using a larger dataset and ensemble learning.

This paper is organized as follows: Section II explains the details of the datasets and deep learning models that we experimented with. In Section III, we present our results and discuss about them. Section IV concludes the study and provides some future directions for further research.

II. MATERIAL AND METHOD

A. DATASET

The dataset used in this study were provided by Chowdhury et al. [12]. The original dataset contains 3,886 Chest X-Ray images. The images in the dataset are divided into three categories as COVID-19, Normal, and Viral Pneumonia according to their diagnosis, and their category distribution is 1,200, 1,341, and 1,345, respectively. Some sample images from the dataset are given in Figure 1.

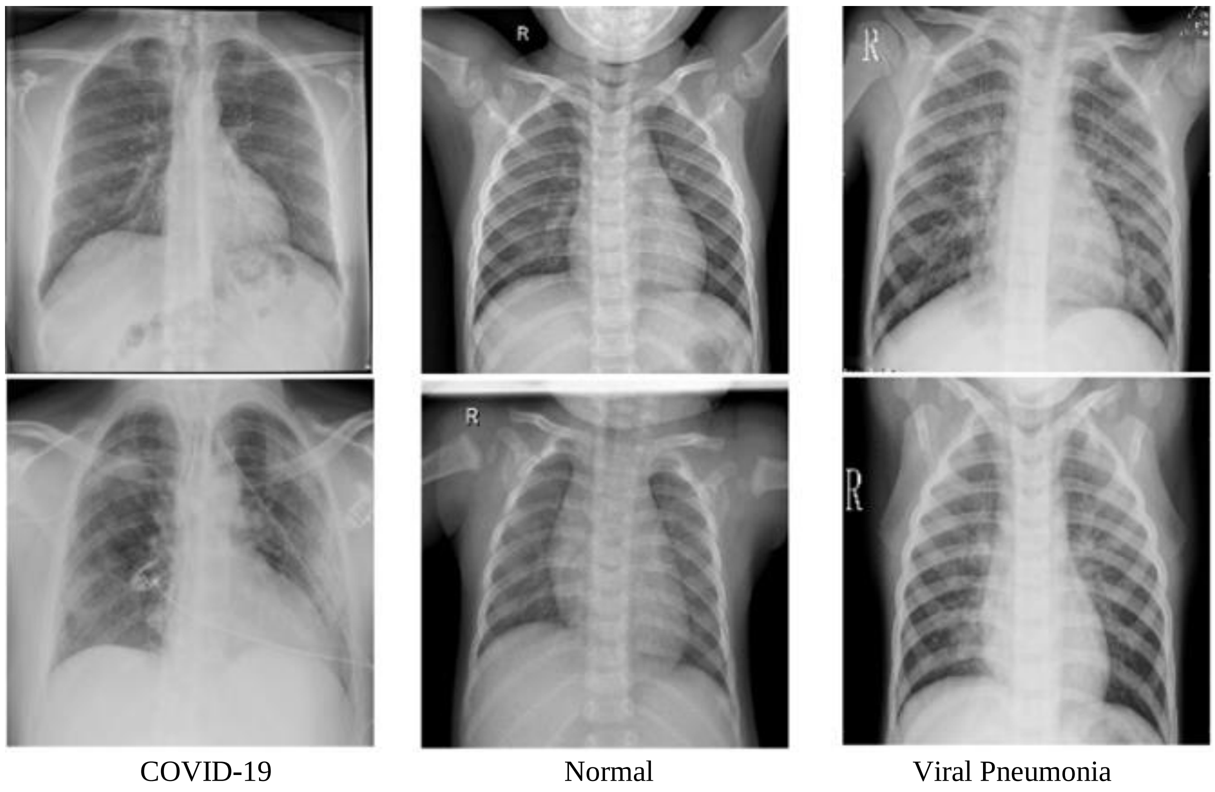


Figure 1. *Some sample images from the dataset [12].*

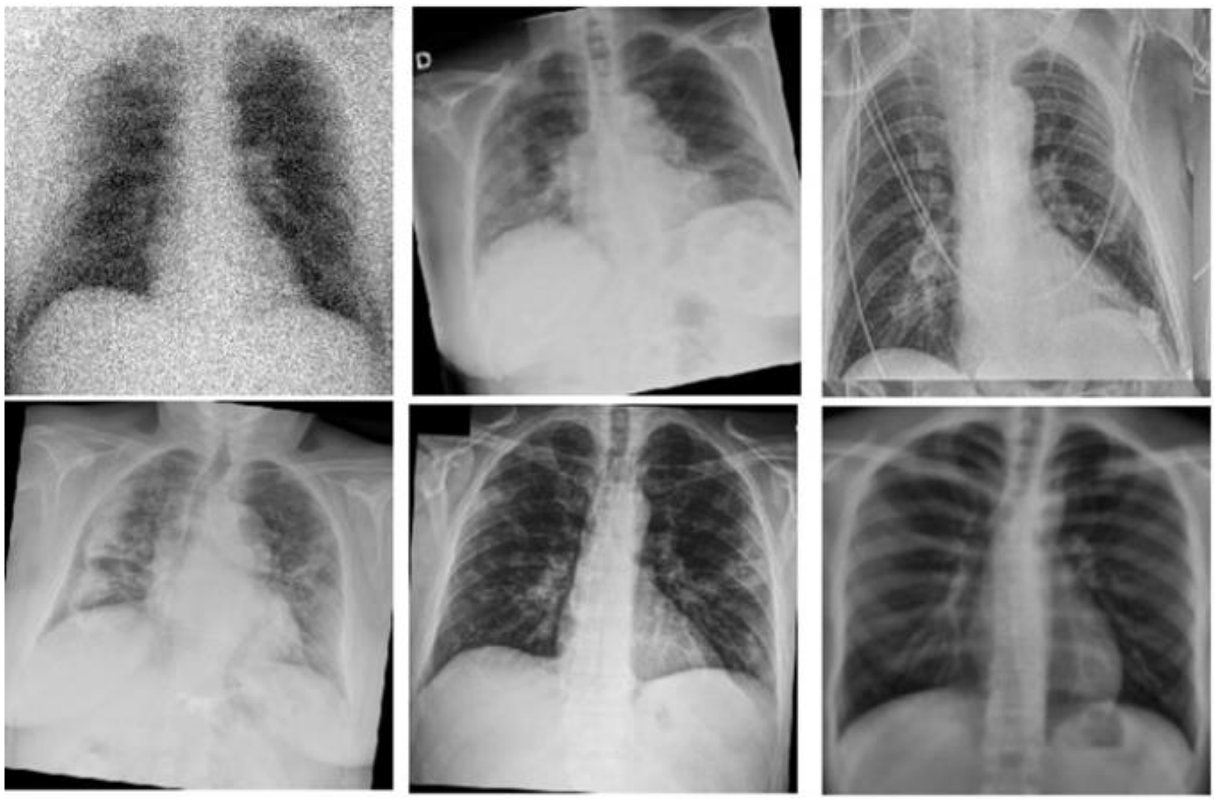


Figure 2. *Some sample images we used to augment the COVID-19 category.*

When we first examined the images in the original dataset, we observed that there were lots of duplicate images, especially in COVID-19 category. Thus, we identified the duplicates using a hashing method, and then removed them.

After the duplicate removal process, the dataset became more imbalanced than the original with 917 COVID-19, 1,339 Normal, and 1,338 Viral Pneumonia images. In order to make the dataset balanced, we augmented the COVID-19 category by warp shifting, rotating clockwise, and adding gaussian blur to 349 images that we picked at random. To achieve that, we used the image processing functions in OpenCV [13] and scikit-image [14] libraries. We flipped the images neither horizontally nor vertically because as data augmentation is performed, category labels of the images should stay the same. However, horizontal flip would make images of Normal category to have the heart on the right hand side of the chest which is a disease called “dextrocardia” [15]. Some sample images we used to augment the COVID-19 category are given in Figure 2.

It is worth mentioning that the dataset contained images in variety of resolutions such as 331x331, 1024x1024, and so on. Following the common practice in deep learning studies, we resized all images in the dataset to 224x224.

Once we obtained the final dataset, we split the dataset into train, validation, and test sets. We used 65% of the images for training, 15% for validation, and 20% for testing, as seen in Table 1. We need to state that we used the augmented images in COVID-19 category only for training. That is, those images did not appear in validation and test sets.

Table 1. *Distribution of image categories in the dataset over train, validation, and test sets.*

Split	**COVID-19**	**Normal**	**Viral Pneumonia**
Train	874	875	826
Validation	184	224	176
Test	208	240	336
Total	**1,266**	**1,339**	**1,338**

B. DEEP LEARNING METHODS

B. 1. Deep Learning

Deep learning is a subset of machine learning where algorithms are inspired by the connectivity patterns of human brain called Artificial Neural Networks (ANN). There are different deep learning architectures that have been applied to fields such as computer vision, speech recognition, natural language processing, and so on. Convolutional Neural Networks (CNN) is an architecture that has been used for image feature extraction. One other architecture is Recurrent Neural Networks (RNN) which has connections between its layers as a form of directed graph, allowing the information carried in layers to remember. Long Short Team Memories (LSTM) are special type of an RNN, capable of remembering long-term dependencies [16].

RNNs and LSTMs are more suitable for sequential data such as text, time series, financial data, speech, audio, video, and so on. Therefore, they are commonly used for tasks such as natural language processing and time series processing. CNNs, on the other hand, are best suitable to work with spatial structures like images.

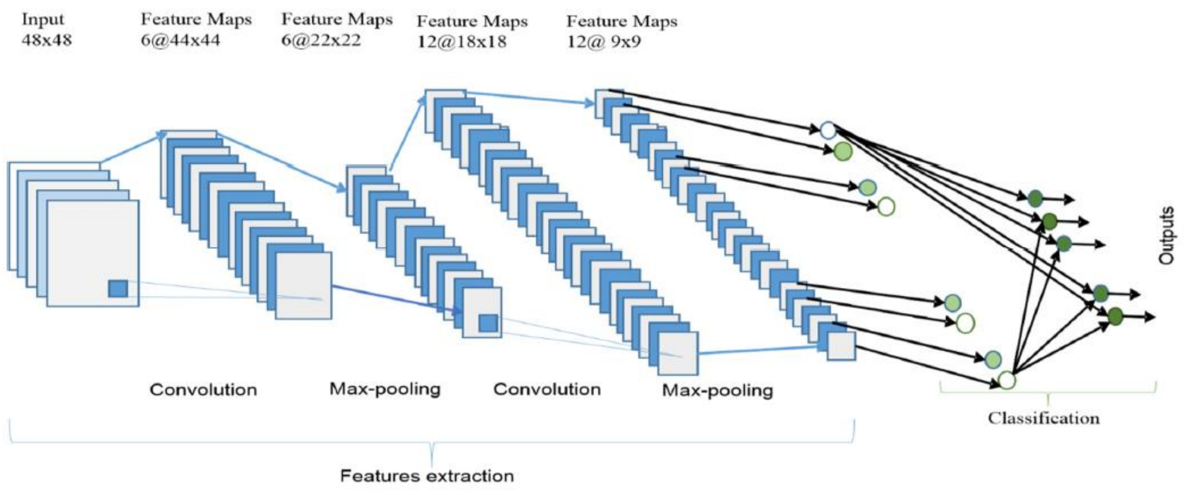


Figure 3. *The overall architecture of the generic Convolutional Neural Network (CNN) [17].*

The overall architecture of the generic CNN is given in Figure 3. Most crucial components of this architecture are the convolution and pooling operations. Convolution represents the direct application of any mathematical filter to a given input that results in an activation. Repeating the same process with the same filter results in a map of activations that is called feature maps, indicating the locations and strength of a detected characteristic of the given input. Pooling is used for reducing the spatial dimensions of mapped feature maps. Pooling layer operates on each feature map independently [16].

B. 2. ResNet Models

ResNet models are one of the most popular CNN (Convolutional Neural Network) models. In theory as neural networks become deeper, the expected performance should increase. However, in practice, performance degrades. In order to overcome this issue, ResNet architecture was introduced [2]. The key idea in this architecture is skipping connections while providing identity shortcuts for the networks. This makes gradient updates much easier for deeper layers. All ResNet models follow the same logic; the only difference is the number of layers in the network.

B. 3. VGG Blocks

The VGG CNN architecture was a significant milestone in deep learning and computer vision. VGG blocks consist of small filter sized convolutions followed by max pooling layers. The VGG blocks start with two convolutional layers which have 64 and 128 filters respectively. Then, the third layer contains 256 filters. In the ordinary usage of VGG blocks, filters are increased with the depth of the network [9].

B. 4. Xception Model

Xception model consists of three main parts named as entry flow, middle flow, and exit flow respectively. The entry flow has two convolution layers followed by a layer of ReLU activation function [5]. In this model Separable convolutional layers are used as a main difference from the previous similar architectures. There are additional “skip” connections, where two tensors are added to merge. Similarly, the Middle flow and the Exit flow are constructed with the same principles.

B. 5. Transfer Learning

Transfer learning is a machine learning technique to re-use pre-trained models for new objectives. That is, layers and weights of a pretrained model are used as a starting point in model creation [16]. Transfer learning is usually used when there are insufficient number of samples to train a model from scratch. In this way, we make use of the information readily available in the parameters of a previously trained model. In addition, training of a new model takes considerably less amount of time. For instance, obtaining a model which was already trained on ImageNet dataset [7] which contains millions of images can be a good starting point for any image classification task. This kind of a model can also be used for COVID-19 detection purposes. An illustration of transfer learning can be seen in Figure 4.

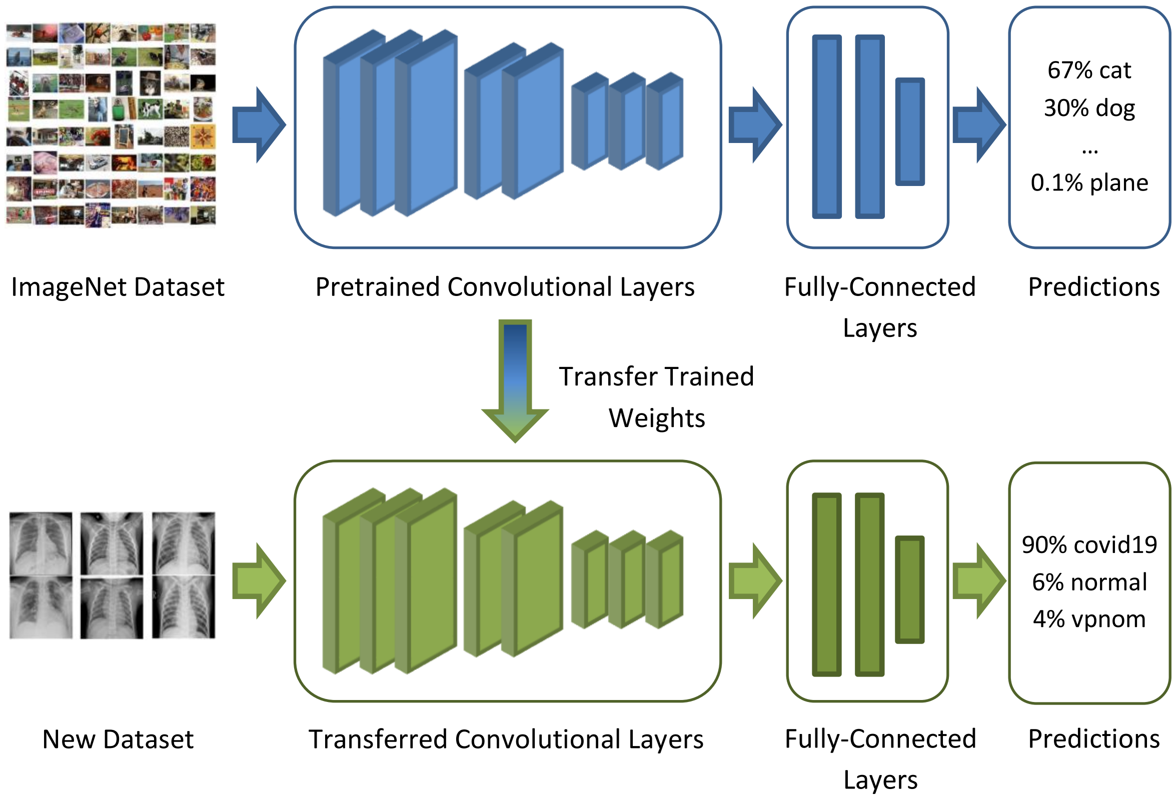


Figure 4. *Illustration of transfer learning: a CNN is pretrained on ImageNet and subsequently trained on X-ray images for our research.*

As the size of our dataset was small, we used transfer learning to fine tune popular pre-trained deep learning architectures in our study. Specifically, we used pre-trained models of ResNet18, ResNet50, ResNet101, ResNet152, VGG16, VGG19, and Xception architectures.

B. 6. Ensemble Models

A special type of classifier, called Ensemble Classifier, combines multiple base classifiers in order to improve the total accuracy of all base classifiers for different circumstances [18]. A result acquired from combination of several machine learning models can be more accurate than a single classifier. The main issue here is how to generate different base classifiers that complement each other. In addition, how to combine the outputs of base classifier for maximum accuracy is another problem [19]. There are several ensemble classification techniques developed for combining multiple classifiers. Well known and widely used ensemble classification techniques are Bagging [20] and Boosting [21, 22].

In this study, we took a bagging approach to combine different deep learning models to improve the classification accuracy. That is, we particularly created two different ensemble models to combine different deep learning models. Every output of each individual model was combined and averaged to get better classification accuracy. We called the first model we created ENS1, which was a combination of VGG16, ResNet101, and Xception models. The second one, ENS2, was based on VGG19, ResNet50, and Xception models. Our choice of base models was based on the performances of the models on the COVID-19 category. General architecture of our ensemble models can be seen in Figure 5.

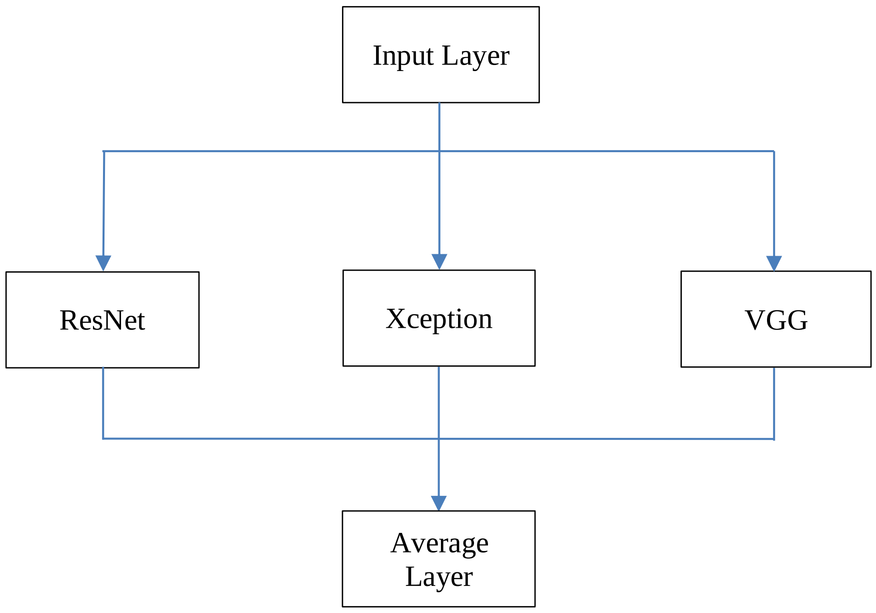


Figure 5. *General architecture of our ensemble models.*

C. MODEL TRAINING

We used GeForce GTX 1660 Ti and Tesla K80 GPUs for training all the models. Models and architectures were generated using TensorFlow 2.5.0 [23] and Keras 2.5.0 [24] in Python 3.8.3 environment. During the training process we used random image processing feature of Keras such as *zoom_range* and *width_shift_range* to prevent overfitting, which is done internally by Keras.

We trained all the models with slightly low learning rates. Categorical cross entropy loss was used to minimize the score of distance between predictions and ground truth values. As pre-trained models include various regularization techniques, we only added dropout layers with a rate of 0.4 between the layers that we tried to fine tune and train. Models and used optimizers to minimize the loss are given in Table 2.

Table 2. *Models and their optimizer parameters for the training process.*

Model	Optimizer	Momentum	Nesterov	Learning Rate
VGG16	SGD	0.9	True	0.00009
VGG19	SGD	0.9	True	0.00009
ResNet18	SGD	0.9	True	0.00009
ResNet50	SGD	0.9	True	0.00009
ResNet101	SGD	0.9	True	0.00005
ResNet152	SGD	0.9	True	0.00005
Xception	SGD	0.9	True	0.00005
ENS1	Adam	-	-	0.00003
ENS2	Adam	-	-	0.0001

Unlike for the other models, we used Adam optimizer for our ensemble models rather than SGD optimizer. Adam is an adaptive optimizer and adaptive optimizers are known to be better than SGD for especially large-scale models and they are prefferred due to their faster convergence [25]. In addition, during our experimental studies, we obtained slightly better performance with Adam than SGD, and thus, we settled on Adam.

The learning rate controls how fast the gradients are updated, in other words, it controls how the model adapts itself to the problem. Higher learning rates make the model converge faster, however, they can cause the model to diverge. Using a low learning rate means the model is taking small steps towards minimum point of loss function. As it takes small steps, more epochs are required. Very low learning rate also can cause model to stuck, which means no learning takes place [16].

For our dataset, using a higher learning rate was giving rise to oscillation of validation loss as well as the training loss. Thus, we picked reduced learning rates with momentum. Additionally, we used three Keras callbacks to control the training process, namely, ReduceLROnPlateu, ModelCheckpoint, and EarlyStopping. ReduceLROnPlateu reduces the learning rate when the given monitoring metric stopped improving. We used ReduceLROnPlateu with the following parameters: *factor* = 0.01, *patience* = 4, *minimum learning rate* = 5×10^{-8} [26]. ModelCheckpoint saves the model to a specific directory when monitoring metric is improved at the end of every epoch. We monitored the validation loss with the parameter *save_best_only* = True. Thus, we overwritten the previous saved model [24]. Early Stopping is a callback we used as a preventive measure to overfitting. Overfitting is a common problem when the training dataset is small in which the model memorizes the training data, and it performs quite well with seen data but performs very poorly with unseen data. In order to prevent this and make sure that our models were not overfitting, we used early stopping with a parameter *patience* = 16 while monitoring the validation loss [27].

D. MODEL EVALUATION METRICS

In order to measure the classification performance of the selected models, we used Accuracy, Precision, Recall, and F-Score metrics. For each of these metrics, the higher the metric value, the higher the performance of a classifier is. We used scikit-learn [28] to generate desired metrics.

When we test a classifier, we obtain four different counts as True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). Using these counts, it is possible to compute the above metrics as given in Eqn. 1, 2, 3, and 4, respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F\text{-}Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

III. EXPERIMENTAL RESULTS AND DISCUSSION

A. MODEL PERFORMANCE COMPARISON

We first obtained classification performance metrics for each model with respect to each one of the three classes COVID-19, Normal, and Viral Pneumonia. In this study, our focus was particularly on the successful detection of COVID-19 cases. Therefore, we first present our experimental results for the COVID-19 class in Table 3.

Table 3. *Model performance comparison for COVID-19 class.*

Model	Precision	Recall	F-Score
VGG16	0.99	0.85	0.92
VGG19	0.98	0.87	0.92
ResNet18	0.94	0.84	0.89
ResNet50	**1.00**	0.93	0.96
ResNet101	0.99	0.91	0.95
ResNet152	0.98	0.88	0.93
Xception	0.99	0.93	0.96
ENS1	**1.00**	0.92	0.96
ENS2	0.99	**0.97**	**0.98**

Most of the models have a high precision but lower recall rates compared to each other. If a model has a high precision rate, then it predicts the given image as positive when it is really positive. It can be trusted. However, with a low recall rate, if it predicts the given image as negative, it cannot be trusted as much as the predictions of positive. There is still some probability that given sample's label is positive. Therefore, for a reliable classification, both high precision and high recall are expected from the models.

When we examine the individual model performances in Table 3, we observe that VGG models almost performed the same, as the number of layers used in the models were close to each other. ResNet50 performed better than ResNet152, which may indicate that there was a high mismatch between the high model complexity of ResNet152 and our dataset. Xception model, on the other hand, performed closely or better than ResNet and VGG models as expected. ENS1 model was better than ENS2 in terms of precision. That was expected, as ENS1 included the models that had the highest precision score. When we constructed the ENS2 model, we chose the models that obtained the highest Recall scores. As a result, our ENS2 model outperformed all other models when it comes to the detection of whether a patient has the COVID-19 infection or not.

As this is a multiclass classification, it is better to compare the performances of models with respect to all classes. In Table 4, we present a comparison of model performances for all classes in terms of F-Score.

Table 4. *Model performance comparison for all classes.*

Model	COVID-19	Normal	Viral Pneumonia	Weighted Average
VGG16	0.92	0.86	0.91	0.89
VGG19	0.92	0.91	0.90	0.91
ResNet18	0.89	0.92	0.91	0.89
ResNet50	0.96	0.92	0.94	0.94
ResNet101	0.95	0.95	0.95	0.95
ResNet152	0.93	0.91	0.94	0.93
Xception	0.96	0.90	0.91	0.92
ENS1	0.96	0.93	0.94	0.94
ENS2	**0.98**	**0.96**	**0.97**	**0.97**

Whereas ENS1 was one of the best-performing models in COVID-19 prediction, it showed only an average performance among the other models in terms of F-Score. On the other hand, ENS2 was the best in overall prediction performance as well as in COVID-19 prediction only.

In order to demonstrate the generalization capabilities of our models, we present the accuracies of the models on train, validation, and test splits in Table 5. When we analyze these accuracy scores, we see that ENS2 achieved the highest level of generalization as well as the highest prediction performance.

Table 5. *Model accuracies for train, validation, and test splits.*

Model	**Train**	**Validation**	**Test**
VGG16	0.91	0.89	0.88
VGG19	0.91	0.91	0.91
ResNet18	0.90	0.88	0.89
ResNet50	0.96	0.95	0.94
ResNet101	0.95	0.94	0.95
ResNet152	0.94	0.92	0.93
Xception	0.94	0.93	0.92
ENS1	0.96	0.95	0.94
ENS2	**0.97**	**0.97**	**0.97**

B. PERFORMANCE DETAILS OF OUR ENSEMBLE MODEL ENS2

Our ENS2 ensemble model is made up of VGG19, ResNet50, and Xception models. The purpose of ensembling these models was to combine the variety of models as every one of them follows a different architecture. We picked ResNet50 instead of ResNet101 because performance scores of ResNet50 on COVID-19 prediction were better. Prediction of each model in the ensemble is combined and averaged to get more reliable results. Keras implementation of ENS2 model is given in Figure 6.

```
models = [model_xception, model_resnet, model_vgg]
model_input = tf.keras.Input(shape=(224, 224, 3))
model_outputs = [model(model_input) for model in models]
ensemble_output = tf.keras.layers.Average()(model_outputs)
ensemble_model = tf.keras.models.Model(inputs=model_input, \
                        outputs=ensemble_output, name='ensemble')
```

Figure 6. *Ensembling models in Keras.*

In Table 6, we present performance scores of ENS2 model for each class in more detail. In addition, we present the confusion matrix obtained from our experiments in Table 7.

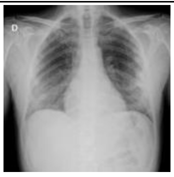
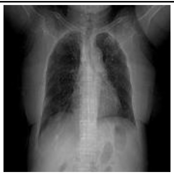
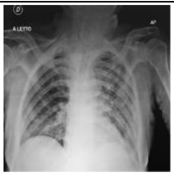
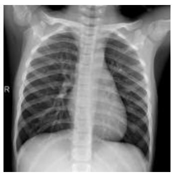
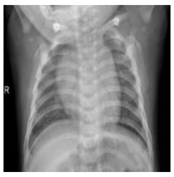
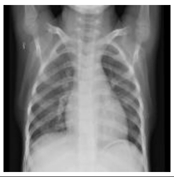
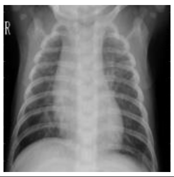
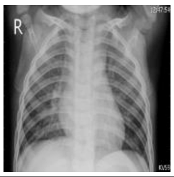
Table 6. *Performance scores of ENS2 model for each class.*

	Accuracy	**Precision**	**Recall**	**F1-Score**
COVID-19	**0.97**	**0.99**	0.97	**0.98**
Normal	0.99	0.93	**0.99**	0.96
Viral Pneumonia	0.95	0.99	0.95	0.97

Table 7. *Confusion matrix of ENS2 model.*

		Predicted Classes			
		COVID-19	Normal	Viral Pneumonia	Total
Actual Classes	COVID-19	**202**	4	2	208
	Normal	0	**238**	2	240
	Viral Pneumonia	3	14	**319**	336
	Total	205	256	323	784

Table 8. *Confusion matrix of test samples as predicted by ENS2 model.*

		Predicted Classes		
		COVID-19	Normal	Viral Pneumonia
Actual Classes	COVID-19			
	Normal	None		
	Viral Pneumonia			

ENS2 model obtained a very high precision, recall, and as a result, a very high F-Score when predicting COVID-19 class. On the other hand, precision of Normal category is not as high as other metrics. When we examine the confusion matrix, we see that our ENS2 model misclassified some Viral Pneumonia images as Normal. This might be because, in some cases, the images from Normal class resemble the images from Viral Pneumonia class as seen in Figure 1. Additionally, we provide sample images from the test set as they were predicted by ENS2 model in Table 8 in a confusion matrix form.

IV. CONCLUSION AND FUTURE DIRECTIONS

In this study, we proposed several CNN models to predict whether a patient has the COVID-19 virus or not from chest X-ray images. We used transfer learning to fine tune a number of pre-trained ResNet, VGG, and Xception models, which are very well-known architectures due to their success in image processing tasks. We trained two ensemble learning models to include a variety of the base models in order to improve the classification accuracy. The best performing ensemble model was our ENS2 model that outperformed the other models with 97% accuracy on the three-class classification. We also show in this study that transfer learning is a highly effective way of creating deep learning models on new tasks.

A major limitation of our study is that we had to use a limited number of X-ray images that depict COVID-19 infection. As a future work, we plan to make our model more accurate and reliable by collecting more images from both local and global public data sources.

Entire system proposed in this study was based purely on medical images of patients. Despite the apparent success proved by our experiments, a better and more reliable system for diagnosing COVID-19 and similar infections can possibly be developed by incorporating some other medical data of patients into the model.

V. REFERENCES

[1] Worldometers.info. (2021, Feb. 10). *Coronavirus Update (Live) from COVID-19 Virus Pandemic* [Online]. Available: https://www.worldometers.info/coronavirus/

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.

[3] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261-2269.

[4] S. Minaee, R. Kafieh, M. Sonka, S. Yazdani, and G. Jamalipour Soufi, "Deep-COVID: Predicting COVID-19 from chest X-ray images using deep transfer learning," *Medical Image Analysis,* vol. 65, p. 101794, 2020.

[5] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800-1807.

[6] A. I. Khan, J. L. Shah, and M. M. Bhat, "CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images," *Computer Methods and Programs in Biomedicine,* vol. 196, p. 105581, 2020.

[7] J. Deng, W. Dong, R. Socher, L. Li, L. Kai, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248-255.

[8] A. 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,* vol. abs/1704.04861, 2017.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR,* vol. abs/1409.1556, 2015.

[10] 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,* vol. 43, pp. 635-640, 2020.

[11] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. Rajendra Acharya, "Automated detection of COVID-19 cases using deep neural networks with X-ray images," *Computers in Biology and Medicine,* vol. 121, p. 103792, 2020.

[12] M. E. H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. A. Emadi, M. B. I. Reaz, and M. T. Islam, "Can AI Help in Screening Viral and COVID-19 Pneumonia?," *IEEE Access,* vol. 8, pp. 132665-132676, 2020.

[13] G. Bradski, "The OpenCV library," *Dr Dobb's J. Software Tools,* vol. 25, pp. 120-125, 2000 2000.

[14] S. v. d. Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and t. s.-i. contributors, "scikit-image: image processing in Python," *PeerJ,* vol. 2, p. e453, 2014.

[15] S. W. Yusuf, J. B. Durand, D. J. Lenihan, and J. Swafford, "Dextrocardia: an incidental finding," *Texas Heart Institute journal,* vol. 36, pp. 358-359, 2009.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge: The MIT Press, 2016.

[17] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics,* vol. 8, p. 292, 2019.

[18] F. Huang, G. Xie, and R. Xiao, "Research on Ensemble Learning," in *2009 International Conference on Artificial Intelligence and Computational Intelligence*, 2009, pp. 249-252.

[19] E. Alpaydin, *Introduction to Machine Learning,* 2nd ed. London, England: The MIT Press, 2010.

[20] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, pp. 123-140, 1996.

[21] R. E. Schapire, "Theoretical Views of Boosting and Applications," in *10th International Conference on Algorithmic Learning Theory AL'99*, 1999, pp. 13-25.

[22] R. E. Schapire, "A Brief Introduction to Boosting," in *16th International Joint Conference on Artificial Intelligence IJCAI'99,* 1999, pp. 1401-1406.

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation,* Savannah, GA, USA, 2016, pp. 265–283.

[24] F. Chollet. (2021, Dec. 12). *Keras* [Online]. Available: https://keras.io

[25] N. Keskar and R. Socher, "Improving Generalization Performance by Switching from Adam to SGD," *ArXiv,* vol. abs/1712.07628, 2017.

[26] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 464-472.

[27] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping," in *Proceedings of the 13th International Conference on Neural Information Processing Systems,* Denver, CO, 2000, pp. 381–387.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.