In the Name of God

University of Tehran

Faculty of Electrical and Computer Engineering

Neural Networks and Deep Learning

Assignment 2

c

Assignment Details	
Provider	Mohammad Taha Majlesi
STUNumber	810101504

Assignment Questions

Question 1: Diagnosing COVID-19 Patients Using X-Ray Images

1.1 Introduction to the Paper and Exercise

During the COVID-19 pandemic, one of the major challenges for the medical community was the rapid and timely diagnosis of infected individuals. In some cases, the high volume of patients visiting medical centers was so overwhelming that it compromised the ability of physicians to accurately diagnose cases, or caused long waiting times for patients. This paper presents a research group's effort to design and evaluate a CNN model for high-accuracy, rapid diagnosis of COVID-19 by training on X-ray images of infected and healthy individuals. The original study used two datasets that are no longer accessible. Therefore, we will use a publicly available dataset from Kaggle. This dataset differs slightly from the original in terms of the number of classes (it includes three classes). The goal of this exercise is to implement the model proposed in the paper, adapt it to our new dataset, and explore the concept of Transfer Learning. We aim to understand why, despite the availability of powerful pre-trained models in Image Classification, researchers continue to design new models for specific applications.

1.2 Dataset Analysis (10 points)

When faced with a new dataset, it is crucial to gain a thorough understanding of it to prepare it for our model. This initial step involves exploring the data's characteristics.

- What classes are included in the dataset?
- What is the format of the data?
- What is the distribution of data across the classes?
- Plot histograms of the number of data points for each class in both the Train and Test sets.
- Explain the benefits of having a balanced dataset for neural networks.
- If the dataset is imbalanced, what strategies would you propose to address this? (Your suggestions should include specific tools and libraries).

1.3 Data Preprocessing (20 points)

In this section, we will prepare the data. After plotting the class distribution histograms, identify the class with the fewest samples in each set (train and test) and randomly sample an equal number of instances from the other classes to create a balanced dataset. You should then prepare this data for model training and evaluation. Subsequently, apply Data Augmentation.

- First, explain what Data Augmentation is and its purpose.
- Using data augmentation techniques, expand the dataset you prepared in the previous step by a factor of 4 to 6. Explain the methods you used and why.

- Save the newly generated dataset for subsequent steps.
- Select one sample from the original dataset and display the new samples generated from it using your different augmentation methods.

1.4 Model Preparation (20 points)

Refer to the paper and implement the proposed CNN model. The model's details are provided in Table 3 and Figure 4 of the paper. After implementing the model, generate a summary and verify it against the details in the paper. Adjust the input image size and the model's output layer according to your dataset. The paper does not specify a learning rate and uses a variable one. Research this approach and, if necessary, use it. Otherwise, if you use a fixed learning rate, justify your choice and explain the potential impact of other learning rates on training. (Justify your claims with loss and accuracy plots).

1.5 Training and Evaluation (30 points)

Train the model according to Table 3 for 15 to 100 epochs. You have autonomy over your system, but try to train for more epochs to reach a stable point of accuracy (30 epochs is sufficient, more is optional). You must determine the appropriate learning rate yourself. Use 35% of the training data for validation.

- Plot the loss and accuracy curves during training.
- Test the model on the test data and report its performance. (Do you have suggestions for improving it? Even if it requires changing the model architecture).
- Evaluate the model using Accuracy, Precision, Recall, and F1-score. Conceptually explain each metric and what insight it provides about the trained model.
- Generate and analyze the Confusion Matrix. Which classes are easier to classify, and which are harder?

It is essential to analyze all the outputs and provide a complete report. The quality of your analysis of the challenges faced during training is crucial. Since the dataset is different, you are not expected to replicate the paper's exact results, but your model's performance should be reasonable and demonstrate good generalization.

1.6 Transfer Learning (20 points)

In this section, we will explore the application of Transfer Learning. As discussed in the course, this technique involves using pre-trained models from different domains. Please explain this technique, specifying which parts of the pre-trained model are used and which are removed. Why? Then, you need to train two models, VGG16 and MobileNetV2, on the initial dataset created in the previous section and report their performance as before. Finally, in a comprehensive table, evaluate and analyze the performance of all three models. Discuss the pros and cons of each model and, specifically, explain why, despite the existence of such pre-trained models, researchers still need to design new models similar to the one in the paper.

Question 2: Implementing a Car Classification System using VGG16 and SVM

2.1 Introduction

With the rapid evolution of artificial intelligence, deep learning and machine learning techniques are widely used for image classification. Vehicle classification, in particular, has become a crucial application in intelligent transportation systems, automated toll collection, and security surveillance. However, traditional deep learning models often require large amounts of labeled data and significant computational power to achieve high accuracy. To address this, a hybrid approach combining deep feature extraction and machine learning classification can provide an effective solution. By increasing the diversity of car models, the need for accurate and efficient identification and classification methods has grown.

In this project, you will implement a hybrid model based on VGG16 and SVM for car classification. The VGG16 model will act as a feature extractor from the images, while the SVM will serve as an effective classifier to analyze and categorize the extracted data. The goal of this method is to increase accuracy and reduce the error rate in classifying images of Toyota brand cars. Finally, you will become familiar with models used in previous studies and, while implementing them, provide a comprehensive explanation of their performance and differences.

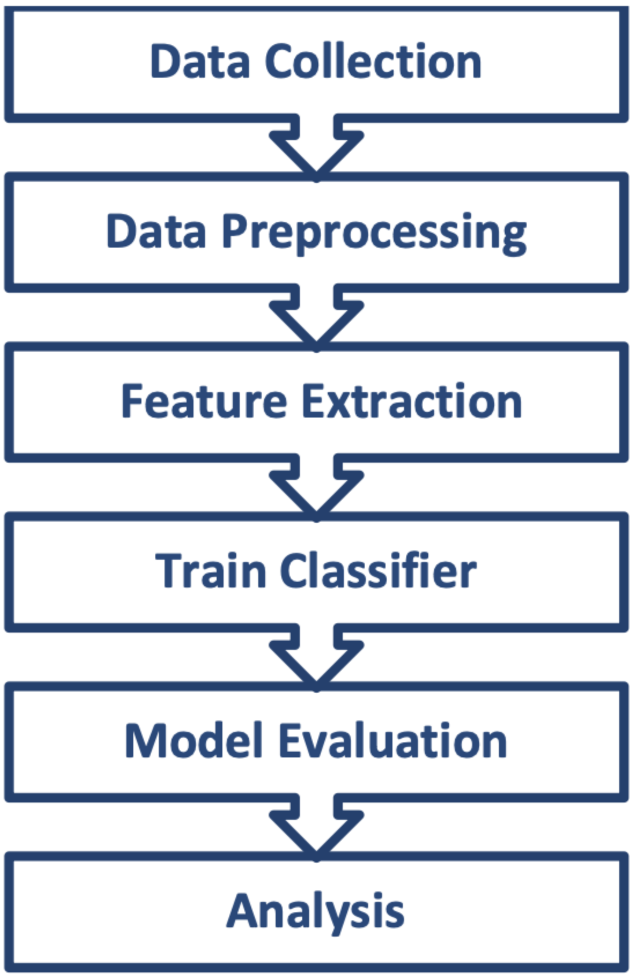


Figure 1: Implementation Steps.

2.2 Data Preprocessing (30 points)

First, download the Toyota car images from the provided link. You need to display the statistical frequency distribution of the dataset and, before proceeding with the preprocessing steps mentioned below, convert the car labels to numerical values.

- From the car classes of this brand, select 10 car models (10 classes) of your choice for the subsequent steps.
- According to the paper, resize the images to 224x224.
- After resizing, normalize the pixel values to be between 0 and 1.
- The given dataset has a limited number of images for each class. Based on the frequency plot you displayed, if the data is imbalanced, first propose a solution to balance it, and while mentioning the advantages of your chosen method over others, implement it. Also, as recommended by the paper, you can use data augmentation to improve the model's generalization.
- Finally, to start training, divide the data into training and testing sets with an 80-20 split and report the dimensions of each set.

2.3 Feature Extraction (30 points)

In this section, first load the pre-trained VGG16 model without its Fully Connected layers. Then, to extract features, pass the images through the last convolutional layer and save the features for each image. In the next step, flatten the extracted features into one-dimensional vectors. Perform the same steps for the AlexNet model and save its features as well.

2.4 Training and Evaluation (30 points)

1. Using the features extracted in the previous step, train the two models mentioned in the previous step and then evaluate their performance on the test data. The evaluation metrics are the same as those in Table 4 of the paper; you must explain the performance of each metric and report the obtained values.

2. State the main difference between AlexNet and VGGNet, regardless of the results.

3. In previous studies, the paper presented a CNN model for better and more comprehensive comparison. In this step, you will implement it and subsequently report the results.

4. The paper proposes a model combining VGG and SVM to improve the accuracy of previous models; in this approach, the SVM is trained linearly with the features extracted in the previous step and acts as a classifier. For this model, as with the previous models, test the results on the test data and report them.

Overall, in this section, you need to report the results for four models within the paper.

2.5 Results Analysis (10 points)

Display the calculated performance metrics from the previous step in a table and provide a comparison of the performance of each model. To facilitate comparison, present graphs similar to Figure 3 of the paper.

- Among the 10 selected classes, state which classes are better classified by each model and which classes each model performs weaker on.

- Now, considering that you have compared the four models, what overall view has been obtained? To improve the performance of the models, what solutions do you suggest to enhance the models' diagnostic power and overall performance?

2.6 Bonus (5 points)

1. Test the proposed model of the paper (VGG+SVM) with different kernels (e.g., Linear, RBF) and state their performance difference.

2. In this paper, before starting classification, feature extraction was performed. Discuss the impact of this action on the final result in classification problems, especially this question.

Student Report

Prepared by: Mohammad Taha Majlesi - 810101504

1 Question 1: Diagnosing COVID-19 Patients with X-Ray Images

1.1 Dataset Analysis

The dataset consists of three classes: COVID-19, PNEUMONIA, and NORMAL, representing chest X-ray images of patients with coronavirus, pneumonia, and healthy individuals, respectively. All images are in '.jpg' format. The training dataset includes 3418 samples of pneumonia, 460 samples of COVID-19, and 1266 samples of healthy individuals. The test set has a similar distribution, with 855, 116, and 317 samples for each class, respectively. The image dimensions are highly variable, ranging from a minimum of 127 pixels to a maximum of 5623 pixels.

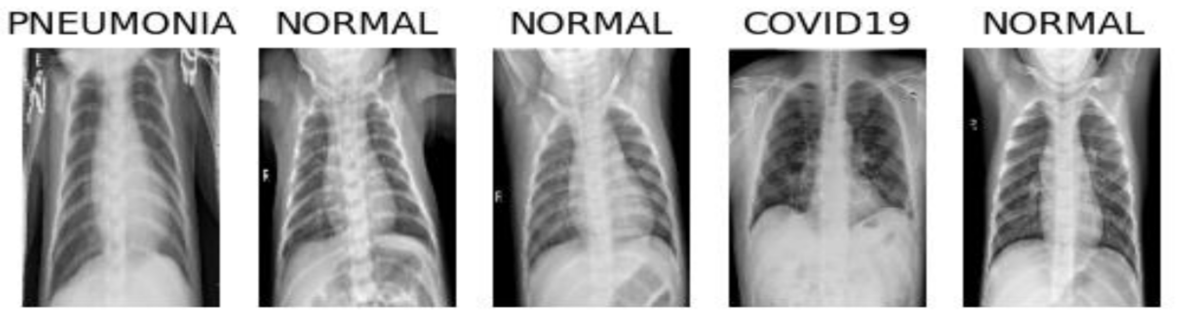


Figure 2: Sample images from the dataset.

1.1.1 Class Balance Analysis

As is evident, the class distribution is imbalanced. This can make the learning process more difficult because the model learns from the data it sees, and a discrepancy in the number of data points for different classes can lead to a bias in the class errors, causing overfitting or underfitting for certain classes. Furthermore, a model might achieve high accuracy by simply predicting the majority class, while failing to correctly identify minority class samples, which are often of higher importance.

One library that provides extensive tools for addressing class imbalance is 'imbalanced-learn'. Generally, some models are more sensitive to class balance than others, and thus, changing the model can improve performance. Additionally, some models allow for assigning higher weights to certain data points, which can be beneficial in such scenarios.

Using appropriate metrics for comparison and a suitable loss function for training models on imbalanced data is crucial. Metrics like 'precision' and 'recall', which are based on different types of model errors, can provide a better assessment based on the goal and importance of the classes. Metrics such as F1-score, ROC, and AUC, by considering different errors, can reflect poor performance even if the error in one class is low but high in another.

Generally, there are two common approaches to handle imbalanced data by modifying the dataset itself. It is important to note that, like most other learning processes, information from the test set should not be used. Thus, these methods should only be applied to the training data, or separately to the training and test sets.

- **Undersampling:** As the name suggests, this method involves removing samples from the majority classes to balance the number of data points across all classes. This approach is generally not recommended as removing data can lead to the loss of important information for learning, resulting in a drop in model performance. However, for datasets where samples are very similar and numerous, and we believe that removing some of them will not lead to a significant loss of useful information, this method can be used. One of the algorithms for this method is **Random Undersampling**, which randomly selects and removes data from the majority class. A better algorithm is **Tomek Links**, which finds pairs of samples from different classes that are very similar and removes the sample from the majority class.

- **Upsampling (Oversampling):** This category of methods, contrary to undersampling which removes data from the majority class, generates data for the minority class. Thus, no information is lost, and they can achieve better performance. One of the drawbacks of these methods is that by increasing the number of synthetic samples of a class, we might cause overfitting. The solution is to create variations between the generated data and the existing data. The **Random Upsampling** algorithm randomly selects samples from the minority class and duplicates them. This can be done with or without replacement of the selected samples. A better algorithm is **SMOTE**, which first selects a number of samples from the minority class and for each sample, considers its K nearest neighbors that are in the same class, and for each neighbor, generates a new sample by interpolating between these two samples.

1.2 Data Preprocessing

Sometimes, the data available for training a model is very limited, and in such cases, using **Data Augmentation** can, by increasing the number of data points, enable the training of a model with desirable performance. In general, with the help of data augmentation methods, we create changes in the data that are subtle enough that we can still assign the same label to the data generated after the changes. Thus, with these changes, we preserve the main features of the data and change features that might be different in real-world data.

The main advantages of this method are increasing the model's generalization power and reducing overfitting, making the model robust to the changes we have created. Also, with the help of these methods, the need for collecting a large amount of data is reduced, and thus, more complex models can be trained with less data.

One of the main applications of this method is for data that is difficult to collect or for which there are few available samples. For example, for predicting rare diseases, this method can be used. In general, this method is common in computer vision when dealing with image data because the ratio of image data to its features (which are dimensions) is very small, and we can generate images with the same label by making changes in light or geometric transformations, thus increasing the model's robustness to changes between images taken in different conditions.

One of the common changes is horizontal or vertical reflection of the image. For medical applications similar to this dataset, only reflection with respect to the vertical axis is usually applied because the human body is somewhat symmetric with respect to the vertical axis. In the reference paper, this type of reflection is also used, but in our opinion, since in all the imaging data, the imaging is done from the front, there is no need to increase robustness to reflection because such images will not exist in the dataset.

Changing the angle of the images is also common, and in the paper, changing the angle by multiples of 90 degrees is applied. For reasons similar to reflection, a large change in angle close to 90 degrees does not occur in real-world data, so we also apply a small change in angle.

Another transformation we use is translation, as the lung may not be in the center of the image, but this change is not applied in the main paper. In the next sections, we will train and evaluate models with a dataset augmented with methods similar to the paper.

In addition to geometric transformations, changes in light and contrast can also be performed, which we think might increase the model's error because X-ray images usually have specific standards, and changes in light and contrast might lead to misdiagnosis.

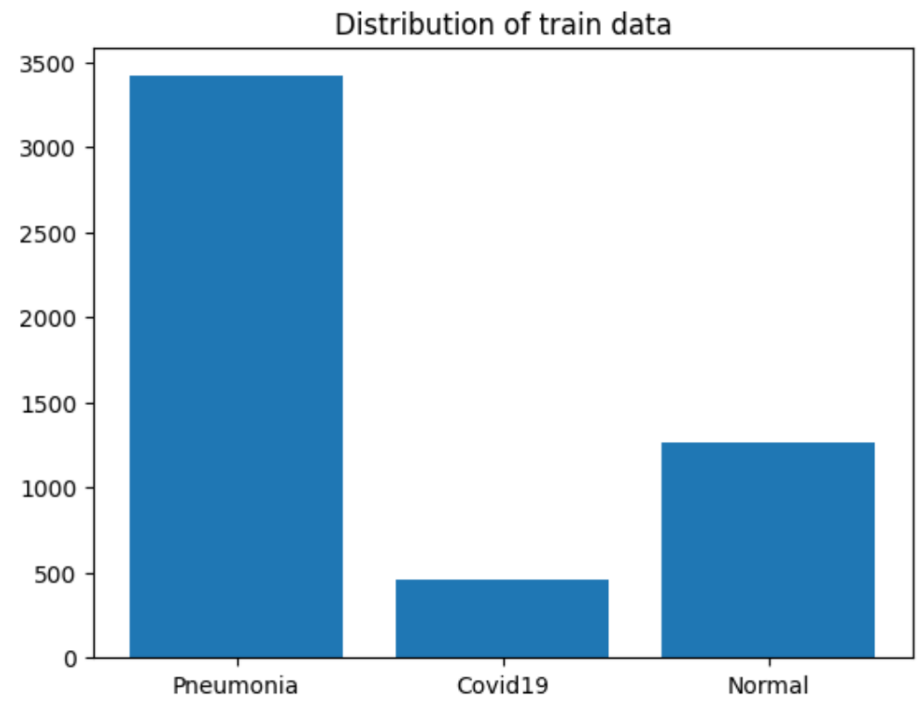


Figure 3: Sample of data augmentation.

1.3 Model Preparation

The paper states that the number of images has been increased fivefold. We also increase the number of images fivefold because by selecting a few individuals, we expect that without considering the original image itself, an equal number of positive and negative angles and spatial changes in each direction are present. It should be noted that we do not augment the test and validation data, and we separate them before augmenting the training data so that no information about them is leaked. Thus, 897 training data are converted to 4485 training data. We also have 483 validation data and 348 test data.

The details of the model are depicted in the paper as follows.

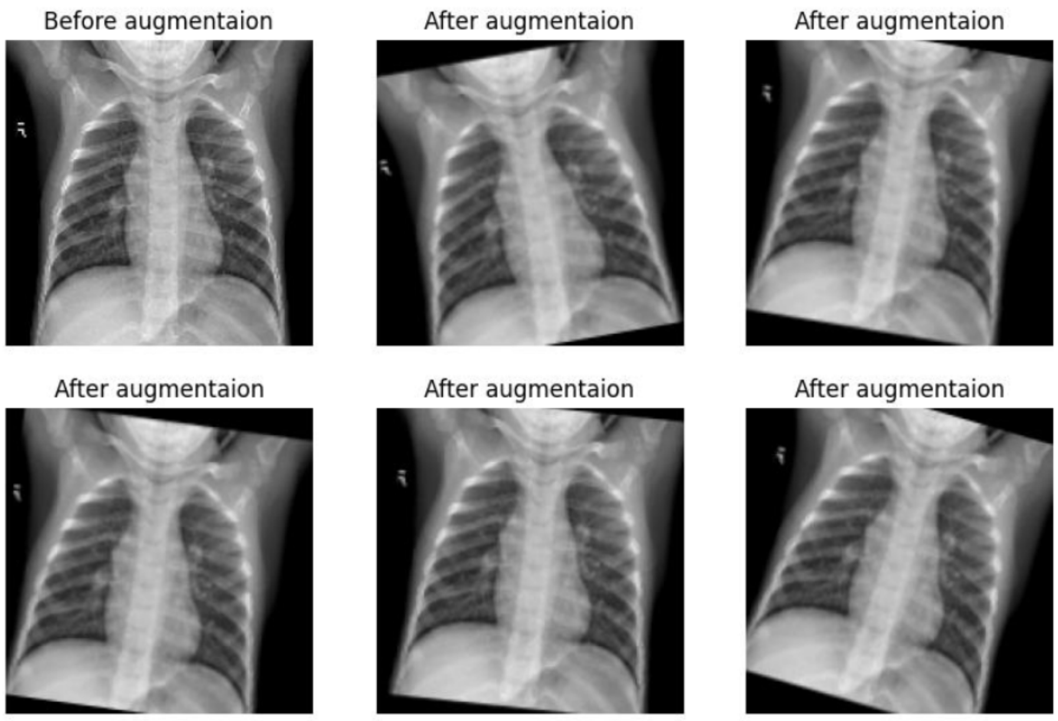


Figure 4: CNN architecture from Figure 4 of the paper.

Table 1: Model parameters from Table 3 of the paper.

Parameter	Value
Input dimension	(150, 150, 3)
Filters to learn	64, 128, 256
Max pooling	2x2
Batch normalization	Axis = -1
Activation functions	ReLU, sigmoid
Dropout rate	20%
Kernel size	3x3
Epochs	50
Optimizer	Adam
Loss function	binary_crossentropy

Given the difference in the dataset, we needed to make changes to the model. Since the paper's dataset has two labels, corona patient or healthy individual, the output of the model in the paper is one neuron, but in our dataset, there are three labels, and for this reason, we place three neurons in the last layer. For a similar reason, instead of using the sigmoid activation function in the last layer, we use the softmax activation function. Also, the loss function used is for a dataset with two classes, but for a three-class dataset that we use, 'sparse$_{c}ategorical_{c}rossentropy`willnotwork, soweuseit.$

After creating the model, we noticed that after passing the data through the last layer, the data dimensions become zero, and thus we receive an error. No explanation was given for this in the paper, and this is not a problem that arises due to the difference in the dataset because we have made the image dimensions similar to the paper. So, to solve this problem, for all the convolutional layers, we set 'padding="same"' so that rows and columns with zero values are added around the images, and thus, the dimensions do not become zero until the last layer.

Due to the large number of layers, only a part of the model's architecture is shown in the image below.

Table 2: Summary of the model architecture without Pooling, Dropout, BN layers.

Layer (type)	Output Shape	Param #
conv2d_204 (Conv2D)	(None, 150, 150, 64)	1,792
conv2d_205 (Conv2D)	(None, 75, 75, 64)	36,928
conv2d_206 (Conv2D)	(None, 37, 37, 128)	73,856
conv2d_207 (Conv2D)	(None, 18, 18, 128)	147,584
conv2d_208 (Conv2D)	(None, 9, 9, 256)	295,168
conv2d_209 (Conv2D)	(None, 4, 4, 256)	590,080
flatten_34 (Flatten)	(None, 4096)	0
dense_102 (Dense)	(None, 512)	2,097,664
dense_103 (Dense)	(None, 256)	131,328
dense_104 (Dense)	(None, 3)	771

1.4 Training and Evaluation

To prevent overfitting and underfitting of the model, we use early stopping. Thus, we train the model for 40 epochs and then continue training until the model's loss on the validation data does not decrease for 25 consecutive epochs. The reason for choosing these numbers is that according to the tests we conducted, models that were trained for more than 80 epochs had up to 5

For evaluating the models, we will use four well-known metrics. Usually, the metrics are defined for binary classification, but they can also be used for multi-class classification. Each of these metrics is described with the help of four simpler metrics: TP, TN, FP, FN. The two metrics FP and FN indicate incorrect classification, and the metrics TP and TN indicate correct classification. When a model classifies data as the class of interest, based on whether the classification is correct or not, the classification is considered part of TP or FP, and if it classifies it as another class, similarly, the classification is TN or FN. Now let's examine the other metrics.

The first metric is accuracy. This metric is the most common metric in machine learning and indicates the overall performance of the model. This metric is obtained from the following relation:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

From the above relation, we understand that from the perspective of accuracy, the value of all classes is the same, and to increase accuracy, it is sufficient for a model to increase the number of correct classifications. The main problem with this metric is in cases where the importance of the classes is not the same, which is one of the most common times when the number of data in the classes is imbalanced. For example, if a model diagnoses all individuals as healthy and only one percent of the individuals are sick, the accuracy is 99

To solve this problem, we turn to other metrics. The two metrics, precision and recall, usually, based on the importance of the positive or negative class, can lead to a better evaluation of the model. These two metrics are obtained from the following relations:

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

It can be seen that both models' values increase with an increase in the classification of the positive class, but the precision value decreases with the incorrect classification of data as the positive class, and in recall, with the incorrect classification of data as the negative class. If we consider the disease example, the high number of TNs does not increase these two metrics. However, in this example, the use of recall is recommended because, among the two possible errors, the reduction of FN is much more important because this error indicates that some patients have been incorrectly diagnosed as healthy, while the other error indicates that some healthy individuals have been diagnosed as sick, which is less harmful. As an example of where the use of precision is recommended, we can consider the detection of spam emails. In this case, the importance of the FP error is greater because it indicates that correct emails have been diagnosed as spam.

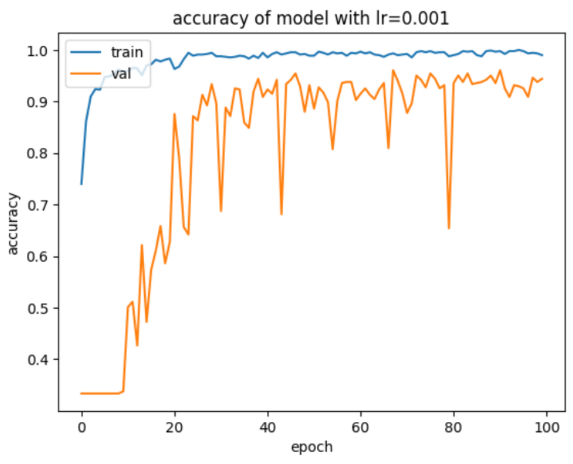
Usually, the increase of both precision and recall metrics simultaneously is not possible due to their different nature, and in cases where both metrics are of high importance, we can use the F1-score metric, which is the harmonic mean of these two metrics, and even if one of these two metrics has a low value, the F1-score will also have a low value.

$$\text{F1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

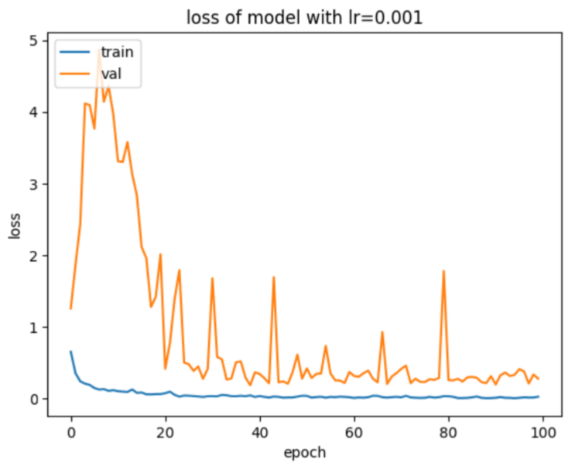
For this project, due to the balancing of the classes, the use of the accuracy metric can help us in comparing the models, but since the number of classes is equal, we expect the other metrics to have similar values. So we also consider accuracy as the main metric for this project, but we will also examine the other metrics, and we consider that in the time of corona, the importance of the recall metric for the corona class was greater.

1.4.1 Training Before Adding Augmented Data

In the paper's dataset, after balancing the classes and before increasing the number of data, the model's accuracy reached about 70 percent. We also, to examine the accuracy before augmentation, trained the model. The two figures below show the changes in loss and accuracy of the model on the training and test data in different epochs during training.



(a) Model loss on balanced data.



(b) Model accuracy on balanced data.

Figure 5: Training history on the balanced (but not augmented) dataset.

The training is stopped after 100 epochs. As expected, with the increase of epochs, the model's loss on both data decreases and the accuracy increases.

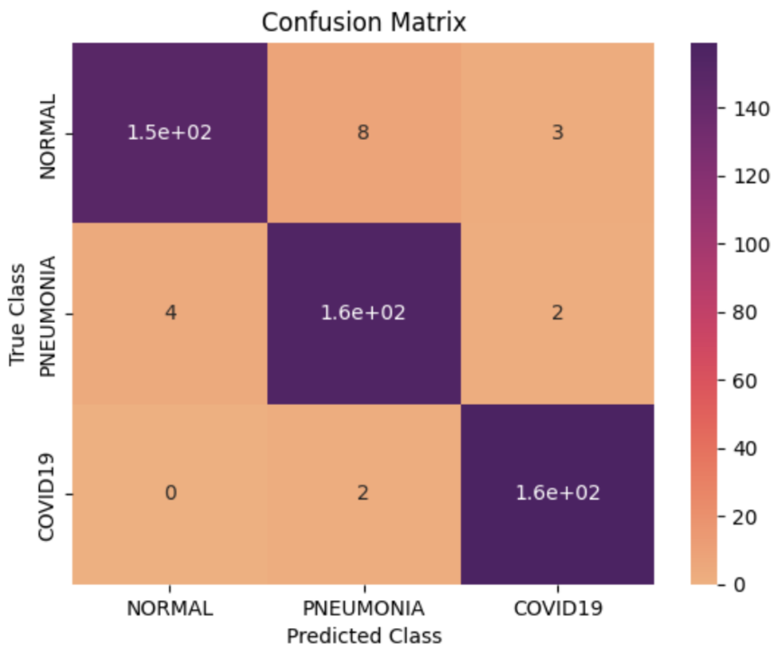


Figure 6: Confusion matrix of the model on balanced data.

Table 3: Evaluation metrics of the model on balanced data.

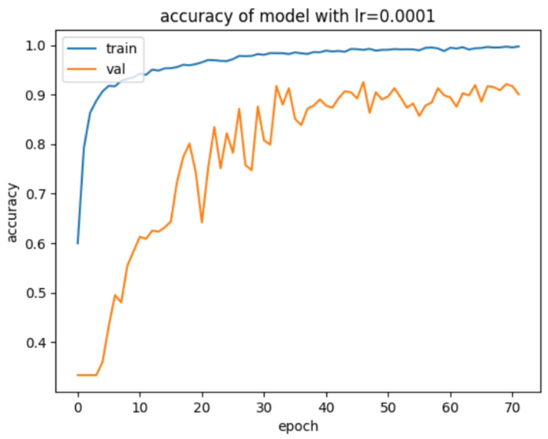
	NORMAL	PNEUMONIA	COVID19	Micro	Macro
Accuracy	93.17	96.27	98.76	96.07	96.07
Precision	97.40	93.94	96.95	96.07	96.10
Recall	93.17	96.27	98.76	96.07	96.07
F1-score	95.24	95.09	97.85	96.07	96.06

It can be seen that the model's performance on all metrics is above 93 percent, which is much higher than the 70 percent that the paper's model achieved. It can also be seen that most of the errors occurred in the diagnosis of patients with pneumonia, whereas the pneumonia class did not exist in the paper's data.

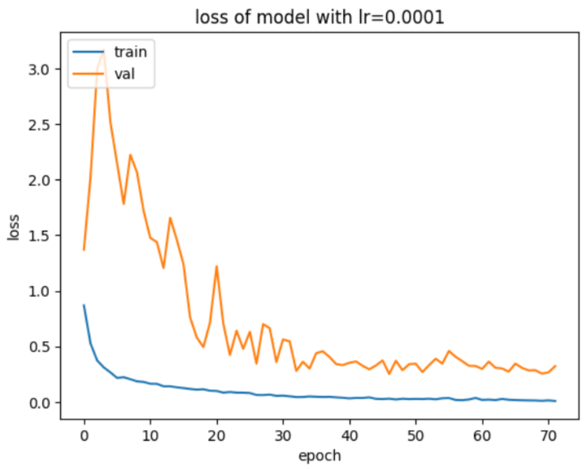
In our opinion, one of the reasons can be the difference in the data. In the paper's data, there are less than 150 images, which after augmentation reaches 900 images, but the number of our images without augmentation is 897. So, perhaps even by making changes in the model and selecting appropriate hyperparameters, the paper's accuracy or better can be achieved without augmentation. Also, in the paper, the data is described as low quality, and it is possible that the quality of the data we use is better, and this is one of the reasons for the model's performance.

1.4.2 Training with a Fixed Learning Rate

Now we train the model with three fixed but different learning rates on the images. First, we set the learning rate to 0.0001.



(a) Loss with LR=0.0001.



(b) Accuracy with LR=0.0001.

Figure 7: Training history on augmented data with LR=0.0001.

The training is stopped after 72 epochs. It can be seen that with the augmented data, the changes in loss and accuracy are much smoother and with less noise. The reason for this can be the larger number of data or the use of a lower learning rate.

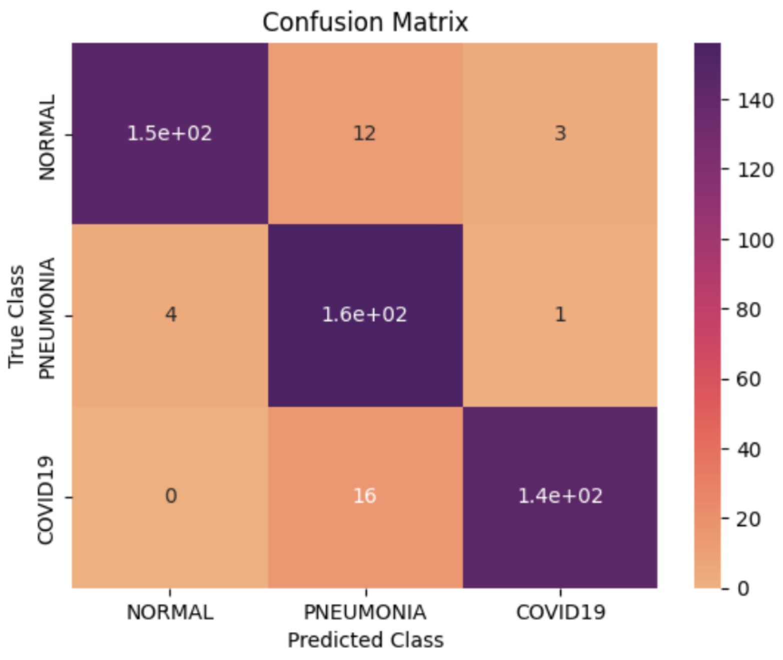
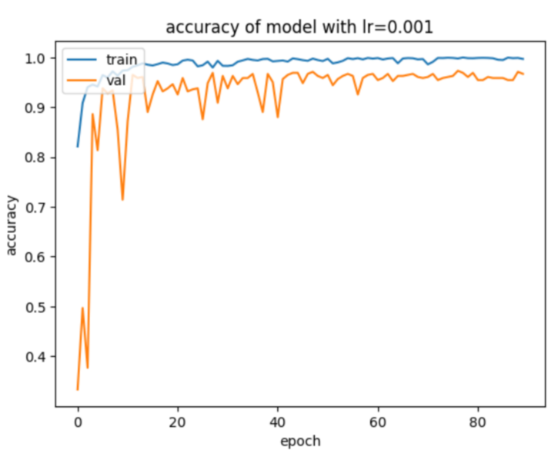


Figure 8: Confusion matrix on augmented data with LR=0.0001.

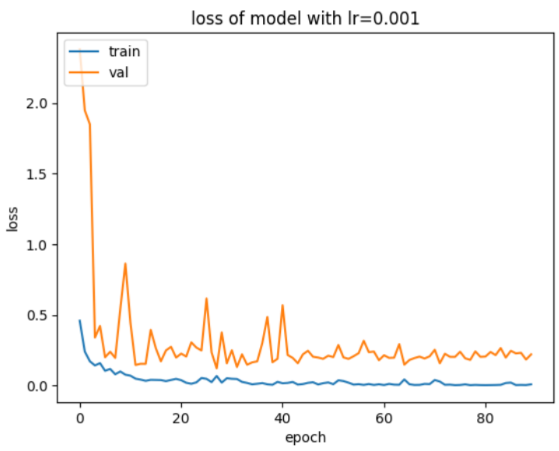
Table 4: Evaluation metrics on augmented data with LR=0.0001.

	NORMAL	PNEUMONIA	COVID19	Micro	Macro
Accuracy	90.68	96.89	90.06	92.55	92.55
Precision	97.33	84.78	97.32	92.55	93.14
Recall	90.68	96.89	90.06	92.55	92.55
F1-score	93.89	90.43	93.55	92.55	92.62

It can be seen that the evaluation metrics have decreased by about four percent with the addition of data. Of course, one of the reasons for the weaker performance can be the early stopping, because the previous model was trained for 100 epochs. We trained the models several times, and in general, it seems that there is not much change in the model's performance, and without the augmented data, it had a similar or stronger performance. Now we train a model with a learning rate of 0.001.



(a) Loss with LR=0.001.



(b) Accuracy with LR=0.001.

Figure 9: Training history on augmented data with LR=0.001.

The training is stopped after 90 epochs.

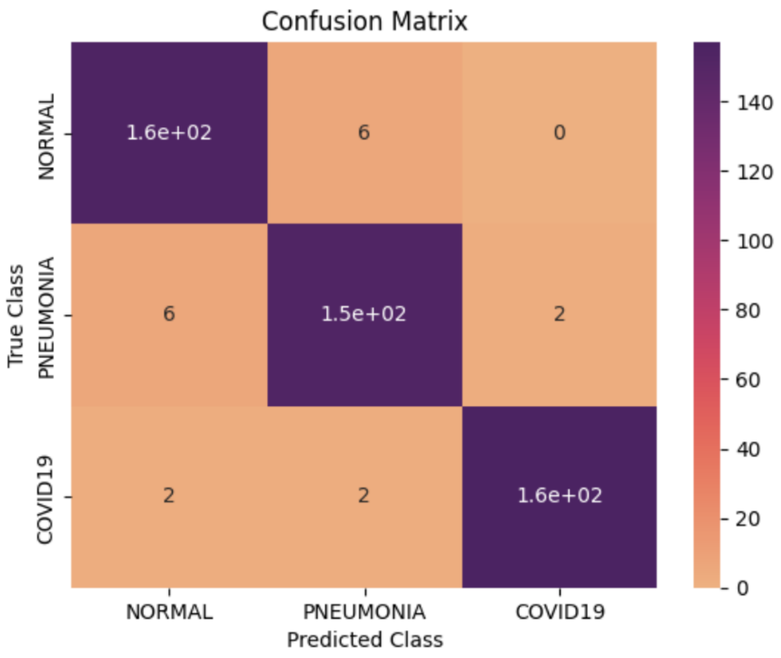
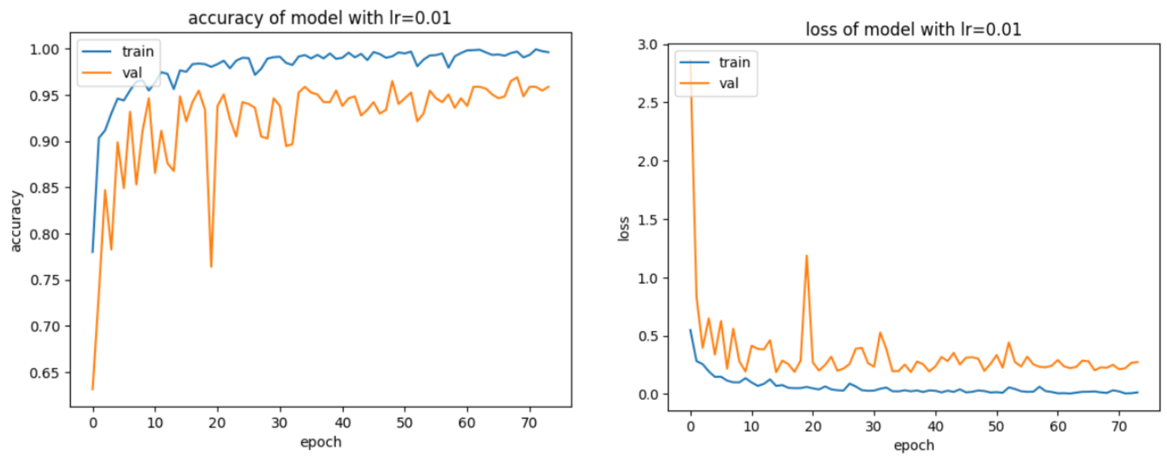


Figure 10: Confusion matrix on augmented data with LR=0.001.

Table 5: Evaluation metrics on augmented data with LR=0.001.

	NORMAL	PNEUMONIA	COVID19	Micro	Macro
Accuracy	96.27	95.03	97.52	96.27	96.27
Precision	95.09	95.03	98.74	96.27	96.29
Recall	96.27	95.03	97.52	96.27	96.27
F1-score	95.68	95.03	98.12	96.27	96.28

It can be seen that compared to the learning rate of 0.0001, the model's performance has improved slightly, which may be due to the slow training of the previous model or getting stuck in local optima, but still, the model's performance is not that different. Now we set the learning rate to 0.01 and train a model.



(a) Loss with LR=0.01. **(b)** Accuracy with LR=0.01.

Figure 11: Training history on augmented data with LR=0.01.

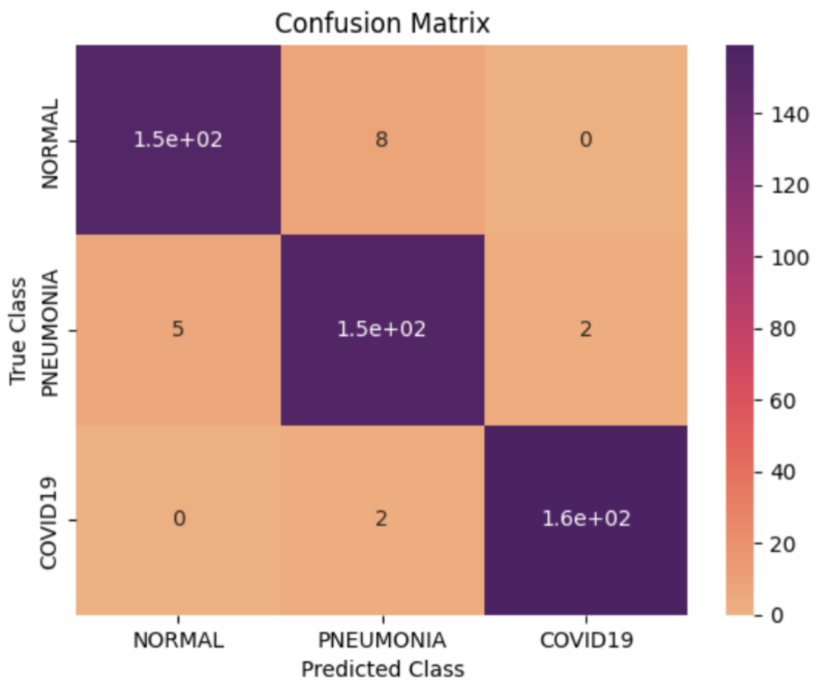


Figure 12: Confusion matrix on augmented data with LR=0.01.

Table 6: Evaluation metrics on augmented data with LR=0.01.

	NORMAL	**PNEUMONIA**	**COVID19**	**Micro**	**Macro**
Accuracy	95.03	95.65	98.76	96.48	96.48
Precision	96.84	93.90	98.76	96.48	96.50
Recall	95.03	95.65	98.76	96.48	96.48
F1-score	95.92	94.77	98.76	96.48	96.48

By examining these four models, conclusions can be drawn. First, it can be seen that since we balanced the classes, all the evaluation metrics have very similar values, and by measuring one of them, the model's performance can be assessed. Also, as we saw, by adding augmented data, the performance metrics did not change much and are still far from the paper's model performance. Of course, it should be noted that the class of pneumonia patients did not exist in the paper's data, and without considering the errors of this class, we reach a similar performance.

In general, the learning rate indicates the speed of change of the model's weights. If a large learning rate is chosen, the weights change with greater intensity, and by choosing a value that is too large, the model may become unstable. On the other hand, if the learning rate is very small, the learning is very slow, and the model may get stuck in local optima and not be able to get out and find better points. The three common learning rates are 0.01, 0.001, and 0.0001, which we tested all three. With the increase in the learning rate, the performance of the models increased, which could be due to the faster training speed, and it is possible that if we train the models with a lower learning rate for longer, they will reach a similar performance, provided they do not get stuck in local optima.

To be able to take advantage of both a high and low learning rate, various methods have been created that consider the learning rate as variable and update its value according to a specific function, which we will examine next.

1.4.3 Training with a Variable Learning Rate

In general, in most methods where the learning rate is variable, initially, a large learning rate is considered to be able to find the optimal points with greater speed without the model getting stuck in them, and over time, the learning rate is reduced to get closer to the optimal point without passing it.

The ‘initial$_{lr}$‘$parameteriscommonamongmanylearningratesandindicatestheinitialvalueofthelear$

First, we examine the Cosine decay learning rate. Its general relation is as follows:

$$\text{initial\_lr} \times \frac{1}{2}\left((1-\alpha)(1+\cos(\frac{\pi \cdot \text{Step}}{\text{DecaySteps}}))+\alpha\right)$$

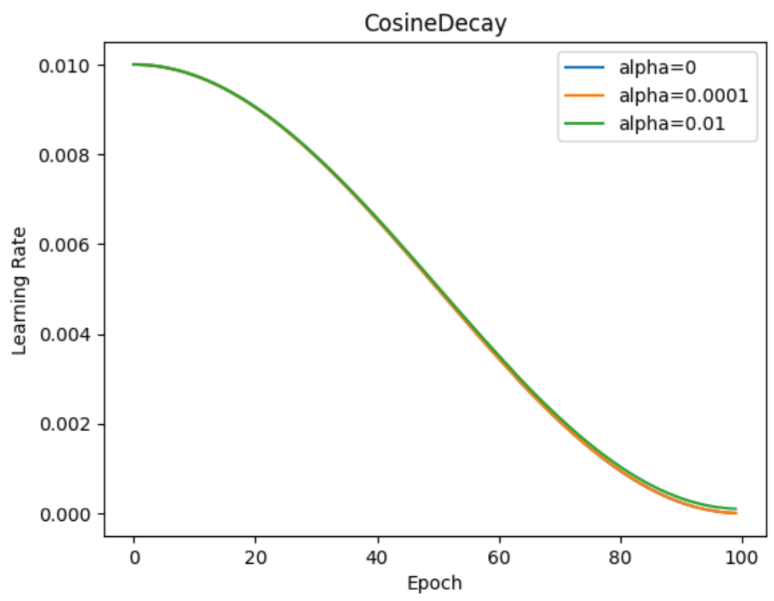
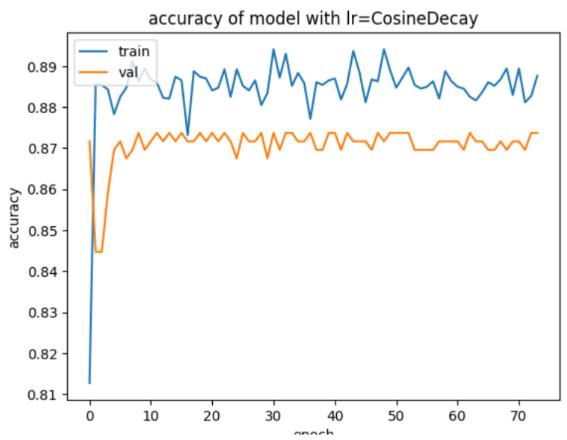
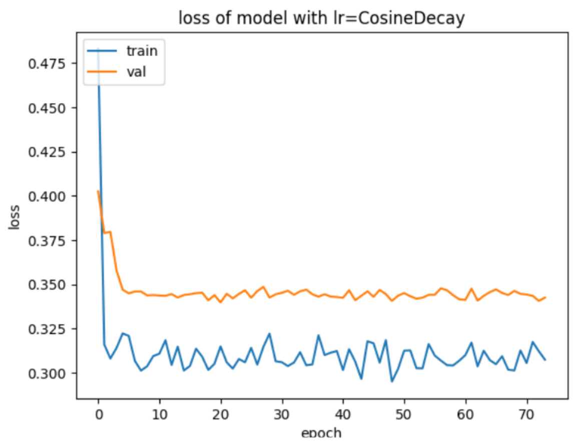


Figure 13: Learning rate changes according to CosineDecay.

It can be seen that by keeping the initial learning rate and the number of change epochs constant, the alpha parameter does not create much change, and we set it to zero and train a model.



(a) Loss with CosineDecay.



(b) Accuracy with CosineDecay.

Figure 14: Training history with CosineDecay learning rate.

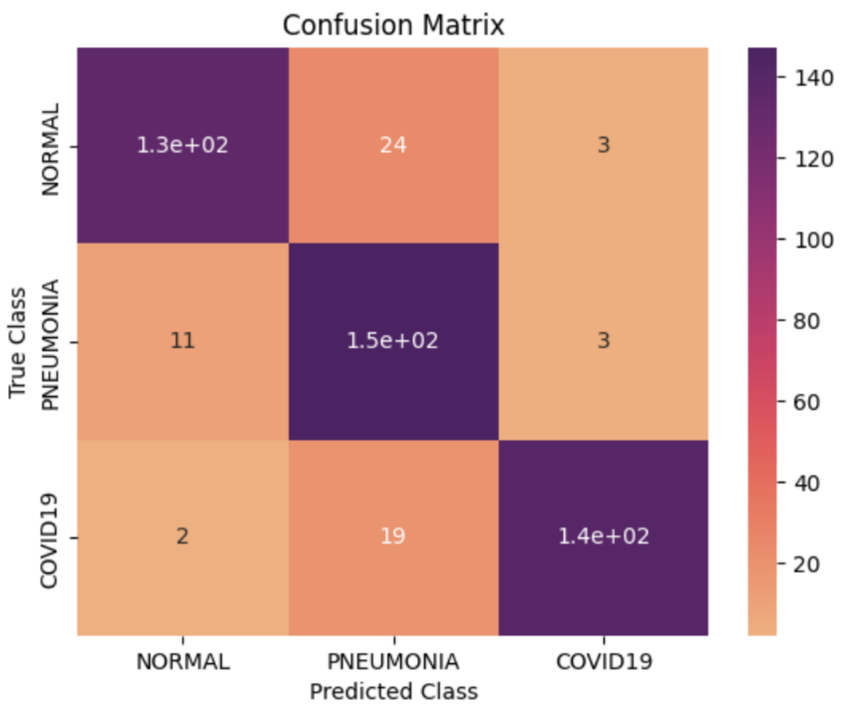


Figure 15: Confusion matrix with CosineDecay.

Table 7: Evaluation metrics with CosineDecay.

	NORMAL	**PNEUMONIA**	**COVID19**	**Micro**	**Macro**
Accuracy	83.23	91.30	86.96	87.16	87.16
Precision	91.16	77.37	95.89	87.16	88.14
Recall	83.23	91.30	86.96	87.16	87.16
F1-score	87.01	83.76	91.21	87.16	87.33

It can be seen that the model has a similar performance to the previous models. Now we move on to the next method for determining the learning rate. The next method is Exponential decay, whose relation is as follows:

$$\text{initial\_lr} \times \text{decay\_rate}^{\frac{\text{Step}}{\text{DecaySteps}}}$$

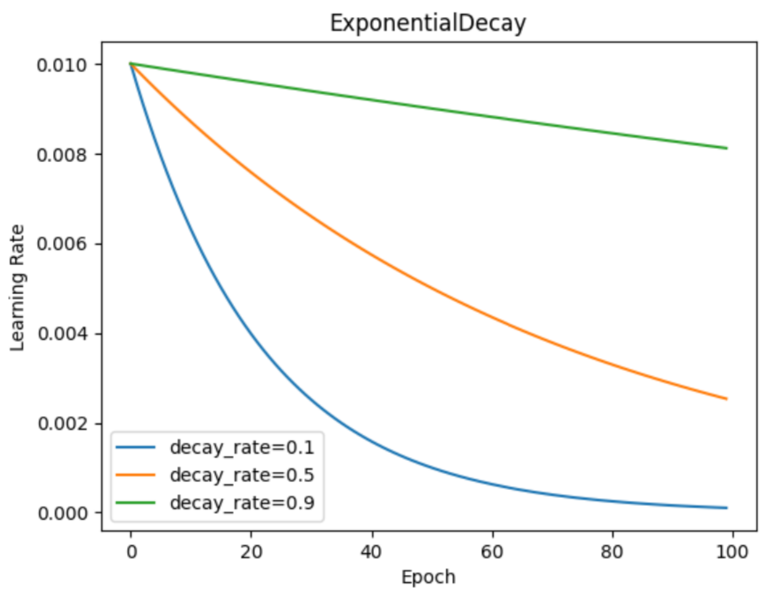
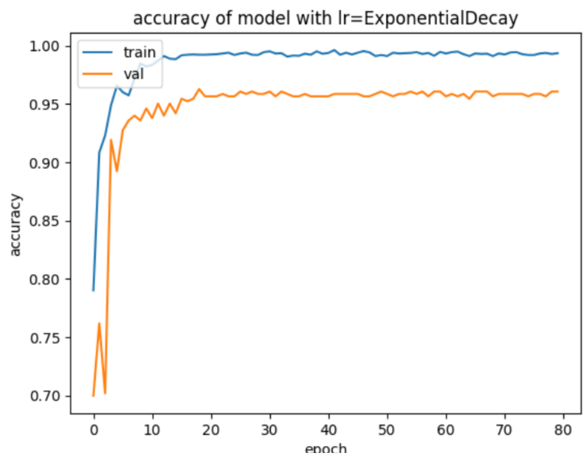
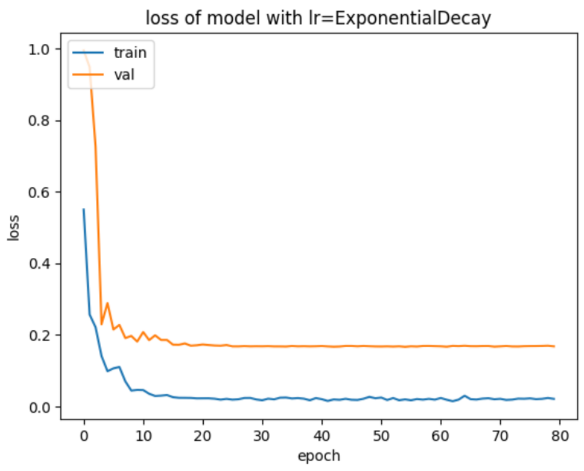


Figure 16: Learning rate changes according to ExponentialDecay.

It can be seen that with the increase of ‘decay$_{r}ate‘, thespeedofthelearningratedecreasedecreases.Si$



(a) Loss with ExponentialDecay.



(b) Accuracy with ExponentialDecay.

Figure 17: Training history with ExponentialDecay learning rate.

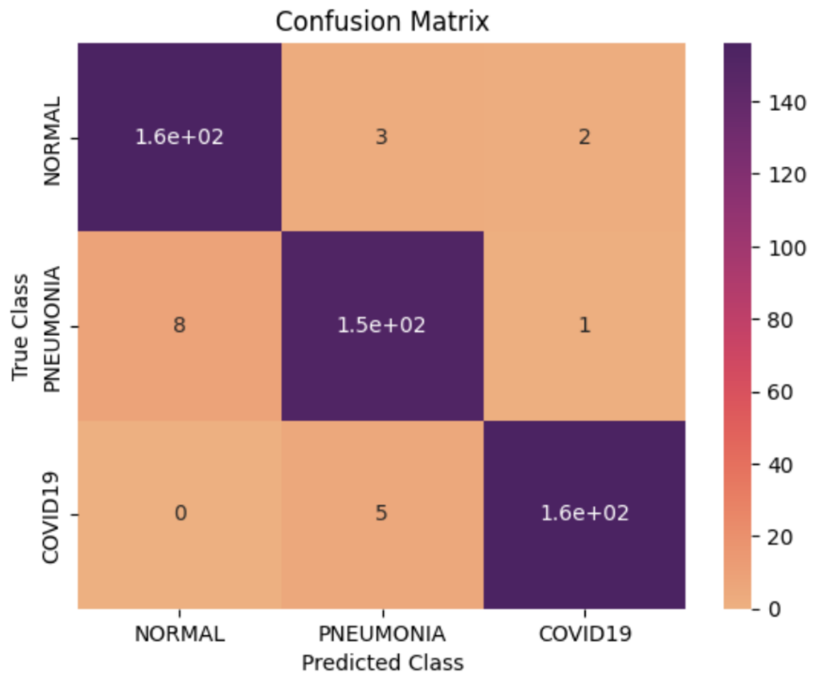


Figure 18: Confusion matrix with ExponentialDecay.

Table 8: Evaluation metrics with ExponentialDecay.

	NORMAL	**PNEUMONIA**	**COVID19**	**Micro**	**Macro**
Accuracy	96.89	94.41	96.89	96.07	96.07
Precision	95.12	95.00	98.11	96.07	96.08
Recall	96.89	94.41	96.89	96.07	96.07
F1-score	96.00	94.70	97.50	96.07	96.07

Again, we arrive at a model with a similar performance. In some methods, periodic functions are used to determine the learning rate, which helps the model to periodically escape from local optima. The next method is Cosine decay restarts, whose relation is similar to Cosine decay but is repeated periodically.

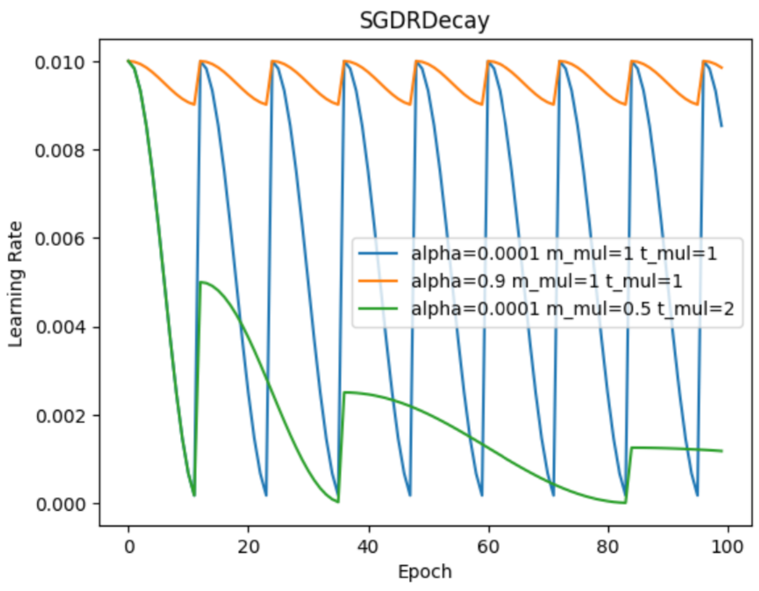
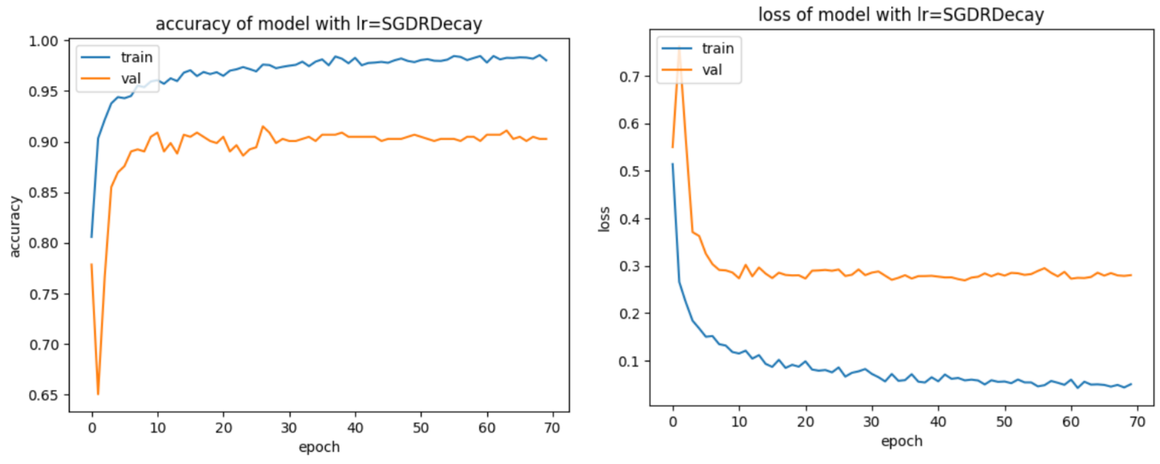


Figure 19: Learning rate changes according to CosineDecayRestarts.

We use the green function in the image above, where although its value increases again after reaching its lowest point, it reaches a value lower than the previous maximum, and thus, over time, the learning rate decreases.



(a) Loss with SGDR. **(b)** Accuracy with SGDR.

Figure 20: Training history with CosineDecayRestarts learning rate.

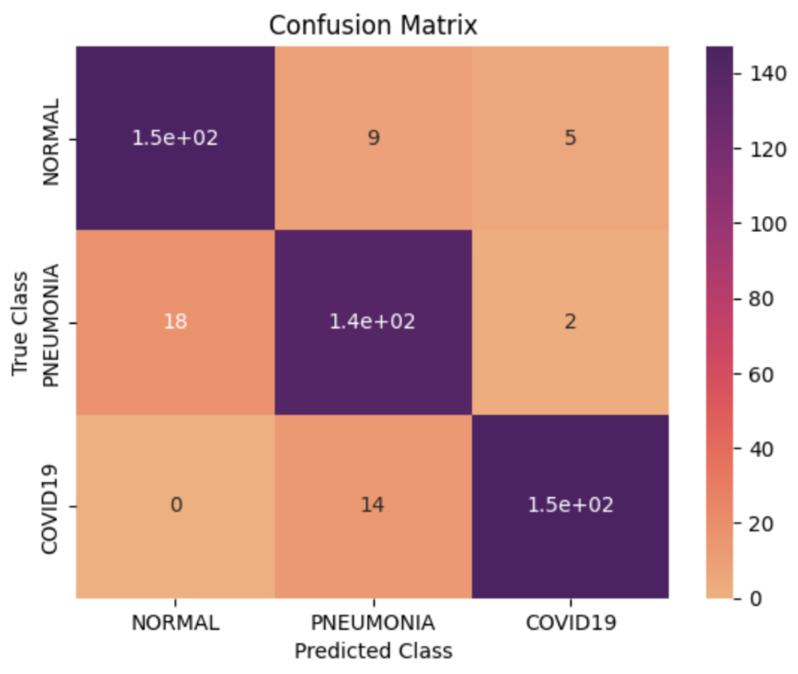


Figure 21: Confusion matrix with CosineDecayRestarts.

Table 9: Evaluation metrics with CosineDecayRestarts.

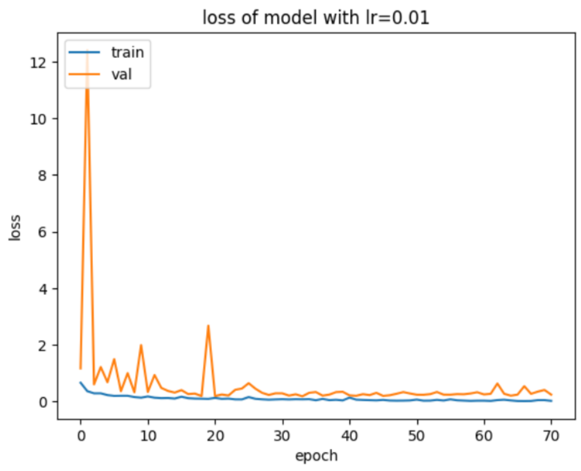
	NORMAL	**PNEUMONIA**	**COVID19**	**Micro**	**Macro**
Accuracy	91.30	87.58	91.30	90.06	90.06
Precision	89.09	85.98	95.45	90.06	90.17
Recall	91.30	87.58	91.30	90.06	90.06
F1-score	90.18	86.77	93.33	90.06	90.10

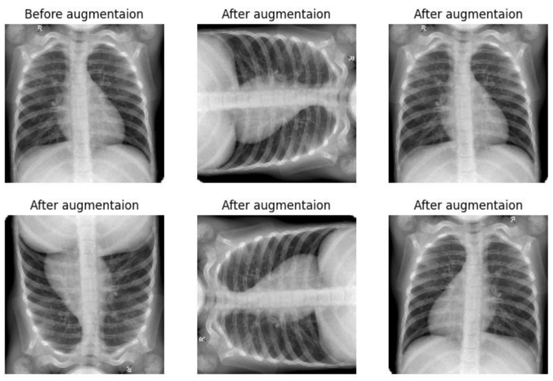
Again, we reached a similar performance. It seems that for this model and images, and even many other models and images, there is not much difference between the different methods of choosing the learning rate. However, it is possible that the methods of changing the learning rate can lead to both an increase in speed and an increase in accuracy, but for our images, all methods reach a high accuracy in a relatively short time, and we did not observe much difference. Of course, it seems that the graph of changes in loss and accuracy with a variable learning rate is smoother and stabilizes sooner, and the reason for this can be that after passing a number of epochs with a large learning rate, the model is located near a local optimum, and then by reducing the learning rate, the changes in loss and accuracy decrease.

1.5 Final Evaluation

Since the performance of the models did not improve, now we train the model again with a fixed learning rate of 0.01 on both the training and validation data and on the test data as the final performance evaluation. By doing this, the number of training data becomes 6900, and we face a GPU memory problem. For this reason, for the final evaluation, we also use data with one channel, i.e., black and white images.

The training is stopped after 84 epochs.





(a) Final model loss on test data. **(b)** Final model accuracy on test data.

Figure 22: Final evaluation on the test set.

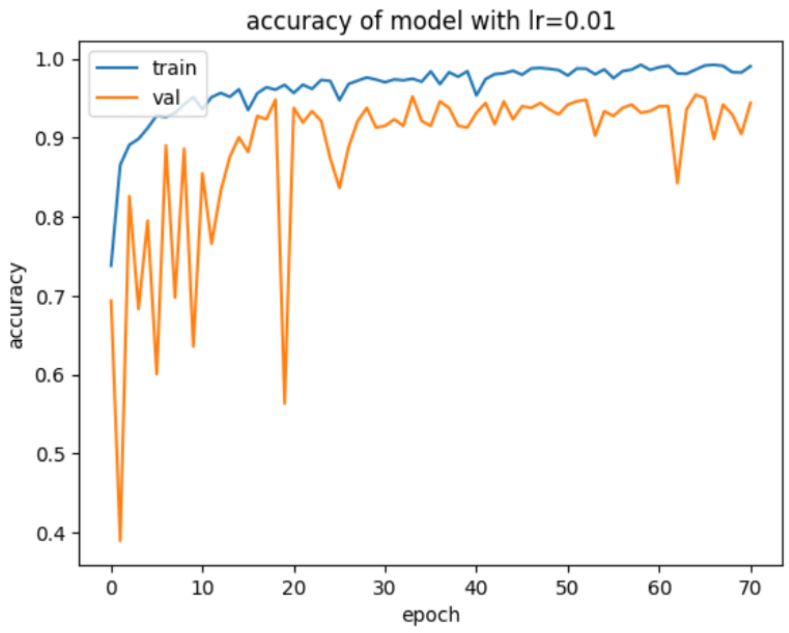


Figure 23: Final confusion matrix for evaluation on the test set.

Table 10: Final evaluation metrics of the model on the test set.

	NORMAL	PNEUMONIA	COVID19	Micro	Macro
Accuracy	90.52	99.14	100.00	96.55	96.55
Precision	99.06	91.27	100.00	96.55	96.78
Recall	90.52	99.14	100.00	96.55	96.55
F1-score	94.59	95.04	100.00	96.55	96.55

Finally, we arrive at a model with high performance. It can be seen that the model has no errors in diagnosing the class of patients with corona. Also, among the images that have been diagnosed as healthy, only one case, which is related to patients with pneumonia, has been mistakenly diagnosed as healthy. But the most error is in the diagnosis of healthy individuals, where 11 cases have been mistakenly diagnosed with pneumonia. We know that in reality, for the diagnosis of diseases, the recall metric is very important, and as is clear, for both diseases of pneumonia and corona, this metric is above 99 percent, which can save many lives and less than one percent of the time, it is possible for a sick person to be diagnosed as healthy. Thus, the training was successfully completed, and we reached the desired performance on the test data. Now we check whether with transfer learning, we can reach this high performance or not.

1.6 Transfer Learning

Transfer learning is a method in machine learning and deep learning that has many applications in various fields, especially in computer vision and natural language processing. In this method, we modify a model that has been trained for a specific task so that it can also perform well on our desired task.

One of the advantages of transfer learning is the lower training cost. Building a new model, in addition to the difficulties of choosing a suitable architecture, can also be costly in terms of training because a model that has not seen any data needs to be trained, whereas in transfer learning, the model has been previously trained on a dataset, and thus, it can be trained in less time on the new dataset. Also, unlike training a new model that requires a lot of data, in transfer learning, since the model has seen a lot of data before and has learned the common features of all images, such as edges, it can perform very well with less data. Similarly, due to training on a lot of data before, in transfer learning, overfitting is prevented because the model has gained general information about reality through different data and its generalization ability increases.

Of course, to achieve acceptable performance in transfer learning, it is recommended that the task we want the model to perform before and after training be somewhat similar, and similarly, the distribution of the initial data and our data do not have a very large difference.

In general, there are three types of transfer learning. In Inductive transfer learning, the source and target data can be different or similar, and both are labeled. In the second method, which is called Unsupervised transfer learning, it is similar to the previous method, but in this case, both the source and target data are unlabeled. For example, a model that recognizes patterns in images can be selected and used to recognize more specific patterns. The last method is Transductive transfer learning, where the source data is labeled while the target data is unlabeled. For example, a model that is trained on sentiment analysis of products can be used for sentiment analysis of users' opinions about movies.

In computer vision, convolutional networks usually first identify basic features like edges in the initial layers. Then, in the next layers, they identify shapes, and in the final layers, they identify features specific to the training data. Since both VGG16 and MobileNetV2 models are trained on the ImageNet dataset, which includes general images from different classes, we remove the final fully connected layers and put new layers in their place and train them. We put a final layer with three neurons to diagnose the three classes of images in both models. We also put another layer before it and chose the number of its neurons in a way that the number of trainable parameters in both models is more than the initial model, which was 1.8 million parameters.

1.6.1 ImageNet Images

This dataset includes more than 14 million images that have been labeled by humans, and in at least one million of them, the bounding box of the object of interest is also present. The images consist of more than 20,000 different classes, and from common classes like strawberries and balloons, there are several hundred images. This dataset is composed of images available on the internet with various dimensions. Also, some of the labels include the features present in the image.

Since 2010, an annual competition for the diagnosis of these images is held, and in 2012, the famous AlexNet model, by using convolutional networks, was able to reduce the error by about 10 percent, which led to an increase in attention to these models and their use in computer vision.

1.6.2 VGG16

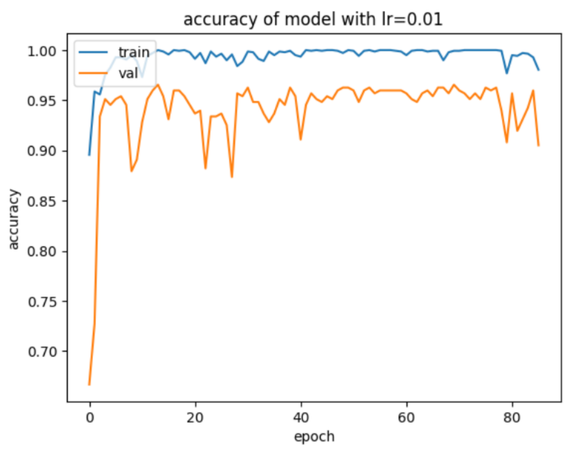
In the 2014 ImageNet competition, the VGG16 model was introduced, which was first in object detection and second in classification. Since then, this model has been known as one of the best computer vision models and is widely used in transfer learning. The creators of the model, by reducing the filter size to 3x3, were able to increase the number of layers to 16 and achieve good performance. The number 16 in the model's name also refers to these 16 layers. The model consists of 13 convolutional layers, 5 max pooling layers, and 3 deep layers, which in total becomes 21 layers, but only 16 layers are trainable. This model takes images with dimensions of 224x224 with 3 channels as RGB input.

Converting images to 224x224 with three channels and augmenting them causes the GPU memory to run out, so we refrain from the image augmentation stage. Thus, we have 1380 training data and 348 test data with dimensions of 224x224 pixels. Now we load the VGG16 model without the final fully connected part and freeze its convolutional part and form a model with the following architecture.

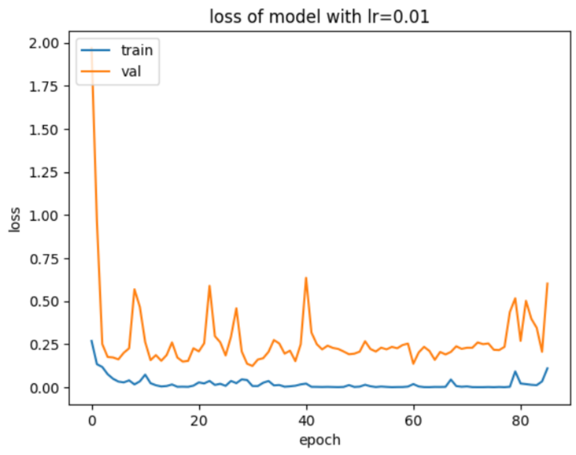
Table 11: Summary of the model architecture formed from VGG16 for transfer learning.

Layer (type)	**Output Shape**	**Param #**
input_layer_13 (InputLayer)	(None, 224, 224, 3)	0
VGG16	(None, 7, 7, 512)	14,714,816
flatten_10 (Flatten)	(None, 25088)	0
dense_29 (Dense)	(None, 64)	1,605,696
batch_normalization_19	(None, 64)	256
re_lu_19 (ReLU)	(None, 64)	0
dense_30 (Dense)	(None, 3)	195

It can be seen that the number of trainable parameters is less than the number of trainable parameters in the previous models. Now we train this model as well.



(a) VGG16 loss on test data.



(b) VGG16 accuracy on test data.

Figure 24: Training history of VGG16 on the test set.

The model training is stopped after 86 epochs, and it can be seen that it reaches a model with acceptable performance. A notable point is that the model's accuracy on the test data after the third epoch is more than 93 percent, which shows the high speed of training due to pre-training and having layers for feature extraction. Thus, transfer learning from ImageNet images to our three-class images was successful, and we reached a model with suitable performance.

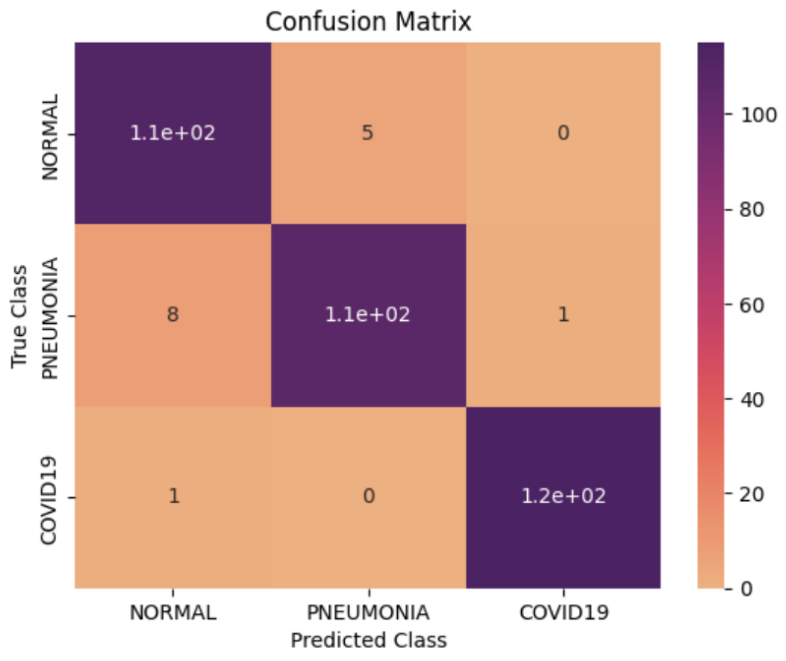


Figure 25: Confusion matrix of VGG16 for evaluation on the test set.

Table 12: Evaluation metrics of the VGG16 model for evaluation on the test set.

	NORMAL	PNEUMONIA	COVID19	Micro	Macro
Accuracy	95.69	92.24	99.14	95.69	95.69
Precision	92.50	95.54	99.14	95.69	95.72
Recall	95.69	92.24	99.14	95.69	95.69
F1-score	94.07	93.86	99.14	95.69	95.69

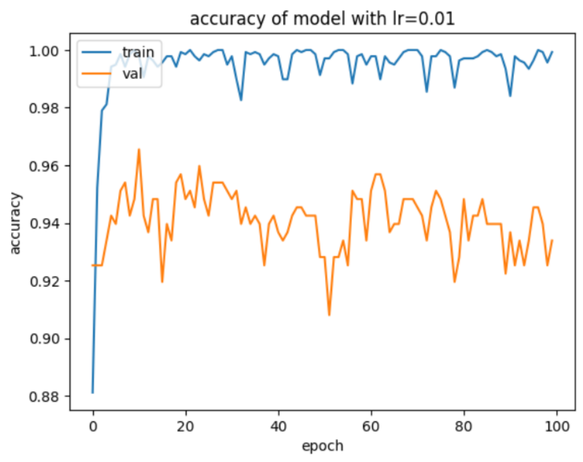
1.6.3 MobileNetV2

This model was created by Google researchers with the aim of working on mobiles and embedded systems. For this reason, various methods have been used to reduce the size and increase the speed and performance of the model. Thus, we remove the final fully connected part of the model and freeze the initial part and add a new fully connected part and arrive at the following model.

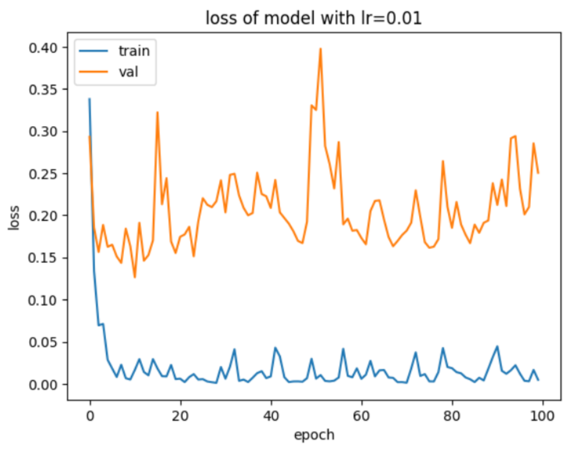
Table 13: Summary of the model architecture formed from MobileNetV2 for transfer learning.

Layer (type)	Output Shape	Param #	Connected to
input_layer_14 (InputLayer)	(None, 224, 224, 3)	0	-
MobileNetV2	(None, 7, 7, 1280)	2,258,016	input_layer_14[0][0]
flatten_17 (Flatten)	(None, 62720)	0	out_relu[0][0]
dense_43 (Dense)	(None, 16)	1,003,536	flatten_17[0][0]
batch_normalization_26	(None, 16)	64	dense_43[0][0]
re_lu_26 (ReLU)	(None, 16)	0	batch_normalization
dense_44 (Dense)	(None, 3)	51	re_lu_26[0][0]

The MobileNetV2 model has many more layers than the VGG16 model, but due to the techniques used and the type of layers, its number of parameters is less than one-sixth of the VGG16 model. We add a number of fully connected layers to the end of the model so that the number of training parameters is less than the first model we used. We also use the images used for training VGG16 for training this model.



(a) MobileNetV2 loss on test data.



(b) MobileNetV2 accuracy on test data.

Figure 26: Training history of MobileNetV2 on the test set.

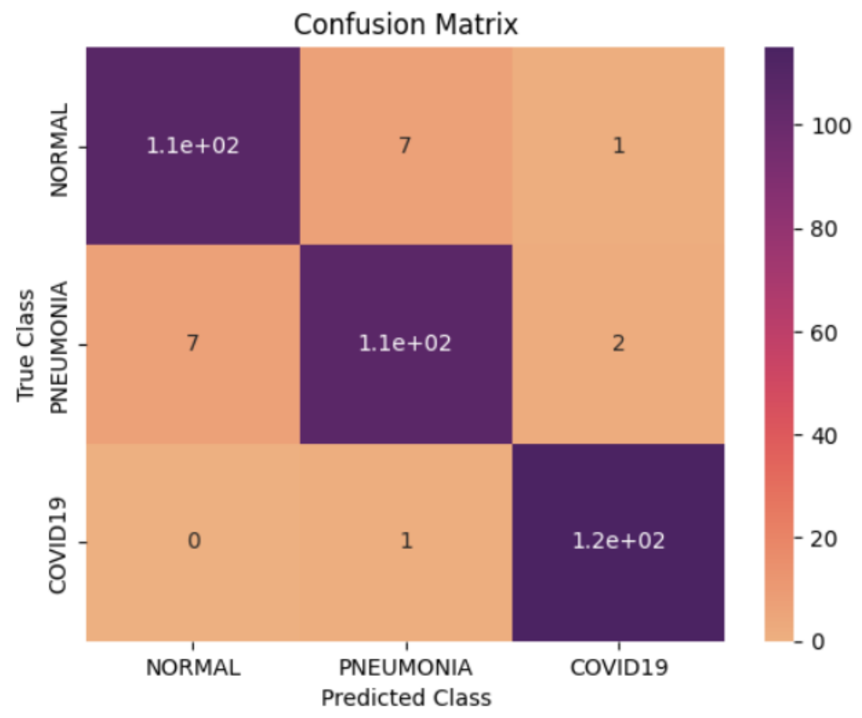


Figure 27: Confusion matrix of MobileNetV2 for evaluation on the test set.

Table 14: Evaluation metrics of the MobileNetV2 model for evaluation on the test set.

	NORMAL	**PNEUMONIA**	**COVID19**	**Micro**	**Macro**
Accuracy	93.10	92.24	99.14	94.83	94.83
Precision	93.91	93.04	97.46	94.83	94.80
Recall	93.10	92.24	99.14	94.83	94.83
F1-score	93.51	92.64	98.29	94.83	94.81

Again, it can be seen that we reach a suitable performance, and in the very first epoch, the accuracy of the test data is above 92 percent, which can be due to the pre-training of the model on other images.

1.7 Summary

In this exercise, we examined the effect of different architectures and different augmentations for training medical images with three classes of healthy individuals, patients with pneumonia, and patients with COVID-19. We also examined a large number of different learning rates. All models achieved an acceptable performance of over 90 percent, and in the best case, their performance reached over 96 percent. Finally, with the help of VGG16 and MobileNetV2 models, we showed that even with fewer training parameters, with the help of transfer learning, we can achieve high-quality models.

Now we examine the three models that were reviewed on the test data, namely the model built in the paper, the VGG16 model, and the MobileNetV2 model with the transfer learning method. All three models were trained with a fixed learning rate of 0.01. Due to the GPU limitations for the models that we trained with transfer learning, we did not use the augmented data. Given the balancing of the classes, all the evaluation metrics have similar values, so we consider accuracy as the main metric.

Table 15: Comparison of the three models with evaluation on the test data.

	Accuracy	**First epoch to reach 90%**	**Total params**	**Trainable params**
Model used in paper	96.55%	2	1,808,963	1,805,635
VGG16	95.69%	3	16,320,019	1,606,019
MobileNetV2	94.83%	1	3,261,635	1,003,619

It can be seen that all three models were able to quickly reach an accuracy of over 90 percent, and ultimately, they have a similar performance, but the model used in the paper has a slightly better performance. The better performance of the model can be due to the larger number of training parameters and the training of the convolutional layers, the use of augmented data, and consequently seeing 5 times more data than the other two models, or due to the difference in the architecture used.

It can be seen that the total number of parameters of the VGG16 model is much larger than the other two models, but the number of trainable parameters of the two models VGG16 and MobileNetV2, as we wanted, is less. Thus, the best performance can be considered for the model used in the paper with a fixed learning rate of 0.01.