DEEP LEARNING PROJECT

COVID-19 SEVERITY

GROUP MEMBERS

ASAD ASLAM – 1000053142

DARIO SAMUELE PISHVAI - 1000003236

HARSH MEHTA - 1000055155

SAMEER AFZAL – 1000053143

SYED MUHAMMAD KHIZAR ALAM - 1000055853

PROFESSOR SEBASTIANO BATTIATO

Contents

Introduction

In the ever-evolving landscape of artificial intelligence, deep learning techniques have emerged as powerful tools for solving complex problems. This report delves into the realm of image processing, with a focus on the application of Convolutional Neural Networks (CNNs) using MATLAB. The utilization of CNNs in this project represents a significant leap forward in the field, showcasing the potential for advanced image analysis and feature extraction.

The primary objective of this endeavor is to harness the capabilities of deep learning to enhance image processing tasks. Convolutional Neural Networks, renowned for their ability to automatically learn hierarchical representations of visual data, are employed to decipher intricate patterns within images. MATLAB, a versatile and widely-used platform, serves as the arena for implementing and fine-tuning these CNNs, allowing for a seamless integration of theory and practice.

In this report, we've employed various types of CNNs (Convolutional Neural Networks) to enhance our accuracy in image processing. Our goals go beyond just setting up the CNNs – we're aiming to make them work even better by adjusting and refining their structure. This way, we hope to improve how well they perform in different situations where we use them for image processing.

Dataset

We've put together a big collection of medical images for our project. This collection has everything, and we've also made four smaller groups from it. Each of these smaller groups focuses on a different picture size: 224x224 pixels, 227x227 pixels, 299x299 pixels, and 224x224 cropped images.

Now, to make things fair, we've organized each of these smaller groups to have exactly 40 patients or folders for each severity type. Severity types are like categories based on how bad a condition is. We have "Mild" for less than 25% severity, "Moderate" for 25-50%, "Severe" for 51-75%, and "Critical" for more than 76% severity. This careful organization helps us make sure every severity type is represented equally and addresses any imbalances we had in the original dataset.

Data samples in each class:

Severity Class	Training	Validation
Mild	133	31
Moderate	124	20
Severe	166	45
Critical	39	5

Pre-processing

Before to try the different settings of the Training Phase, is a good idea working on the images that we want to pass through the network.

A first attempt is represented by the different resizes of the images obtained by not a simple resize but with an interpolation operation. (As you can see in the section "Code".)

What we do during the pre-processing is apply to the image some operation useful to emphasize specific patterns for our problem of multi-classification. Operations such as:

- High Pass filter, applied after a transformation in the Fourier domain (in our case we choose the High Pass Gaussian Filter);
- Morphological Operator (opening, closing);
- Cropping the images.

At the end of this process, in our opinion, the most useful operation is the cropping of the images.

The code of those operators will be provide in the "Code" section.

Training Phase

Processor used
Nvidia RTX 3060 6gb

Hyperparameters optimization

We have used different types of hyperparameters with different networks. Following are the best hyperparameters according to the networks.

| Network	Optimizer	Learning Rate	Batch Size	Epochs	Validation Frequency
AlexNet	Adam	0.001	60	10	50
DenseNet201	Sgdm	0.0001	128	10	50
Efficient-b0	Sgdm	0.0001	60	10	50
GoogleNet	Adam	0.001	30	10	50
Resnet-18	Adam	1e-06	128	10	50
Resnet-50	Adam	0.0001	60	10	50
SquezeNet	Adam	0.001	60	10	50
VGG-19	Adam	0.001	30	7	50
Xception	Sgdm	1e-05	30	5	50

Training Networks

We use 9 networks for training our data.

AlexNet
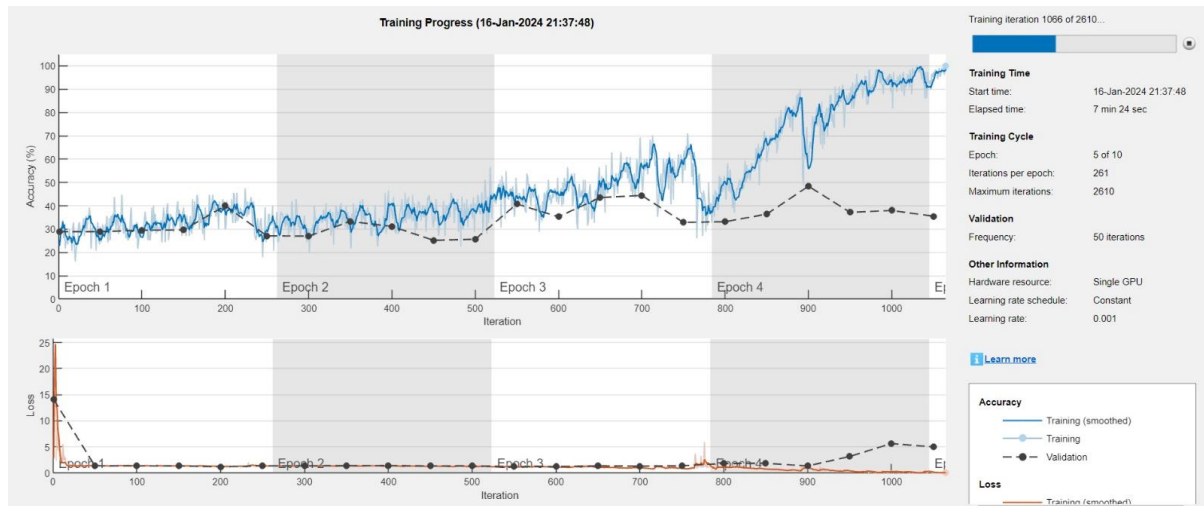
This is the first network we used for training our data and we observed that data is overfitting and accuracy is moving toward the worst.

DenseNet201
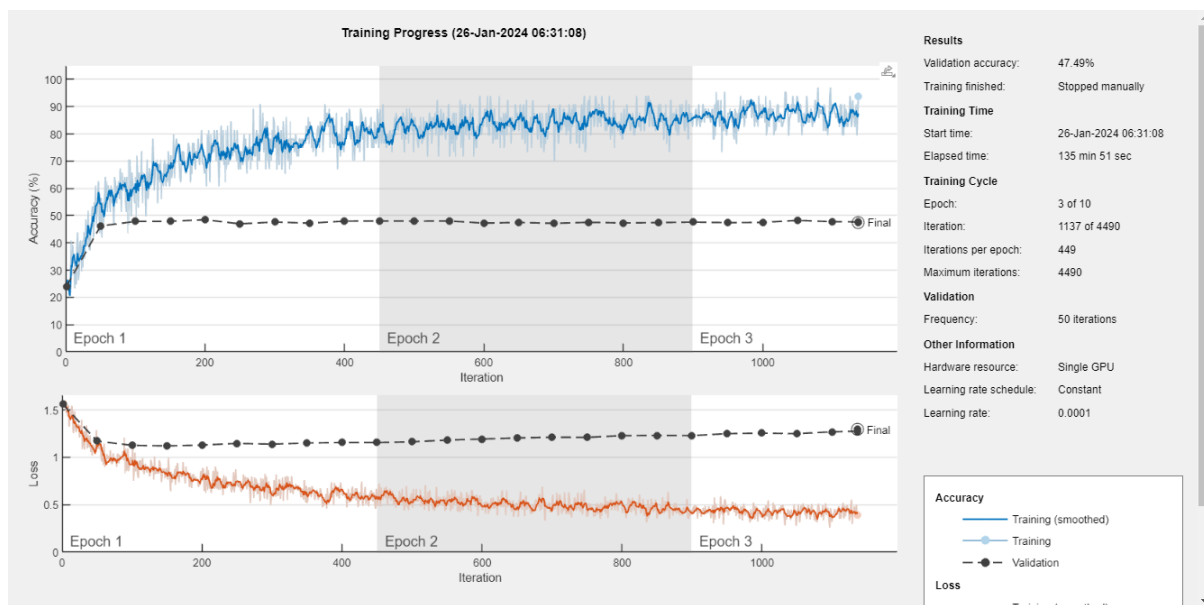
As we can see that in training data (blue line) a bit suffers from a over fitting and the loss curve in the training set is also indicative of over fitting of data. Furthermore, we tried different hyperparameters but still got the accuracy of 47.49% which is very bad.
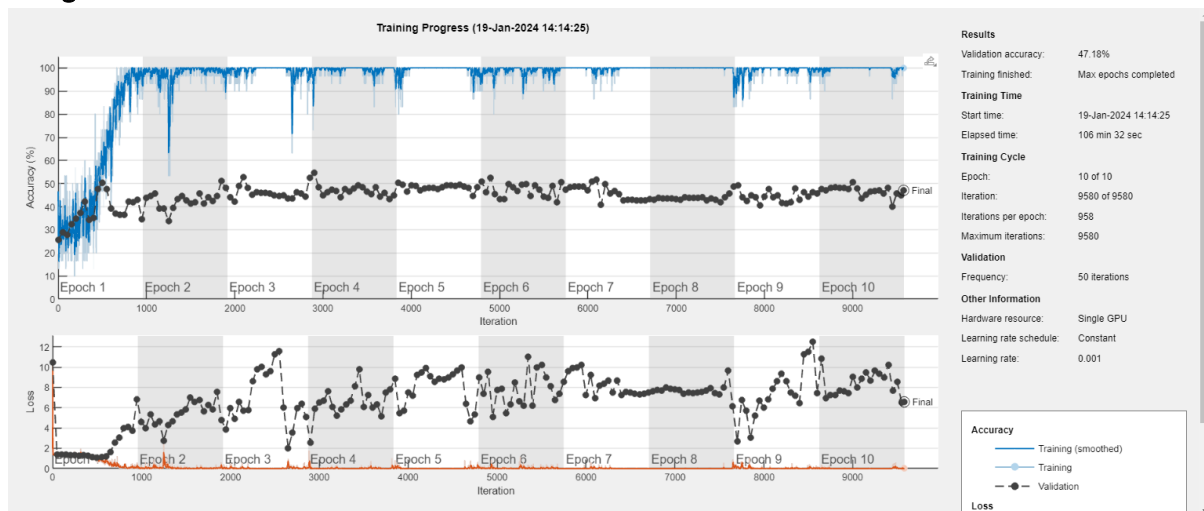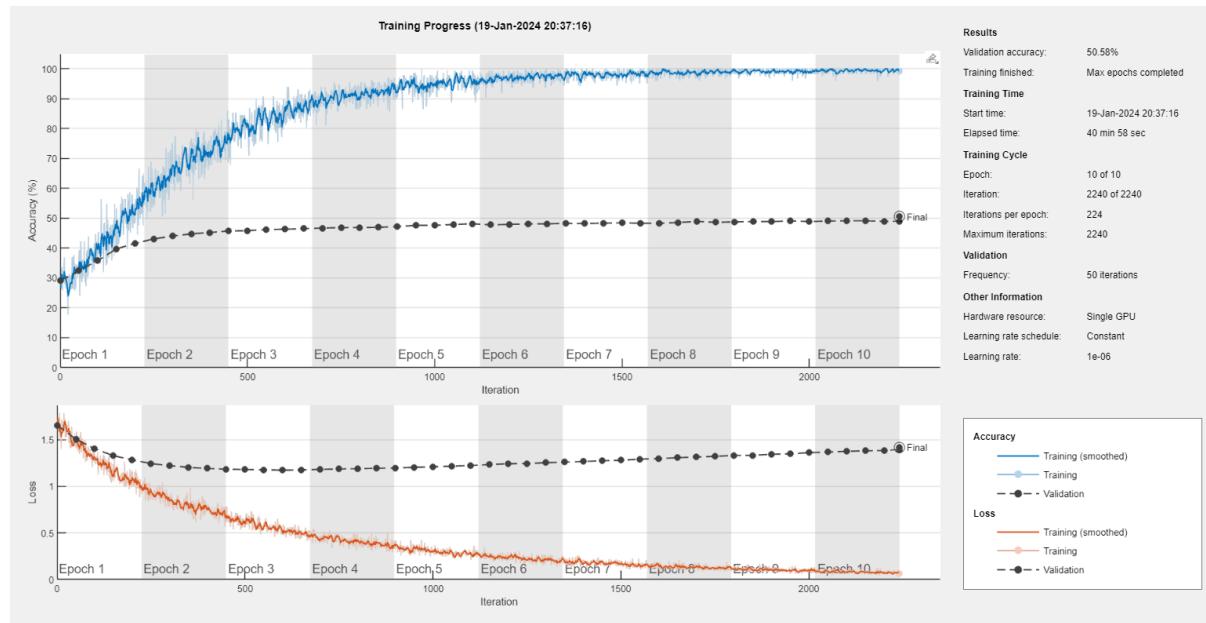
Efficient-b0
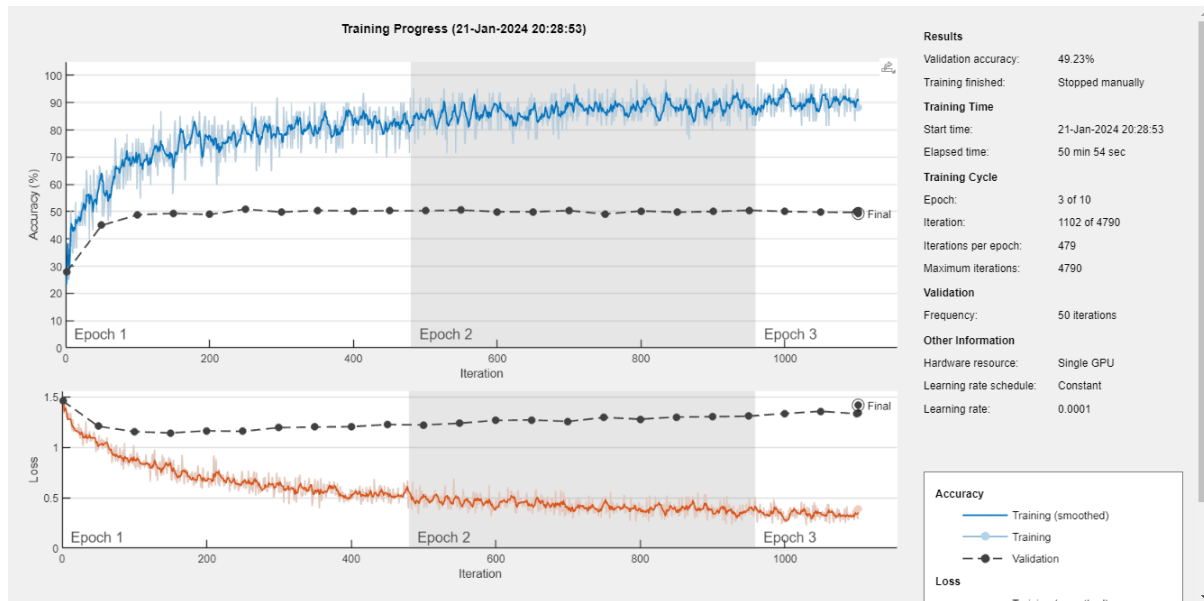
As we can see that in training data (blue line) a bit suffers from a over fitting and the loss curve in the training set is also indicative of over fitting of data. This network is almost providing the same results as we got in above Dense Network 201. However, the accuracy of this model in 49.23% which is also very bad.
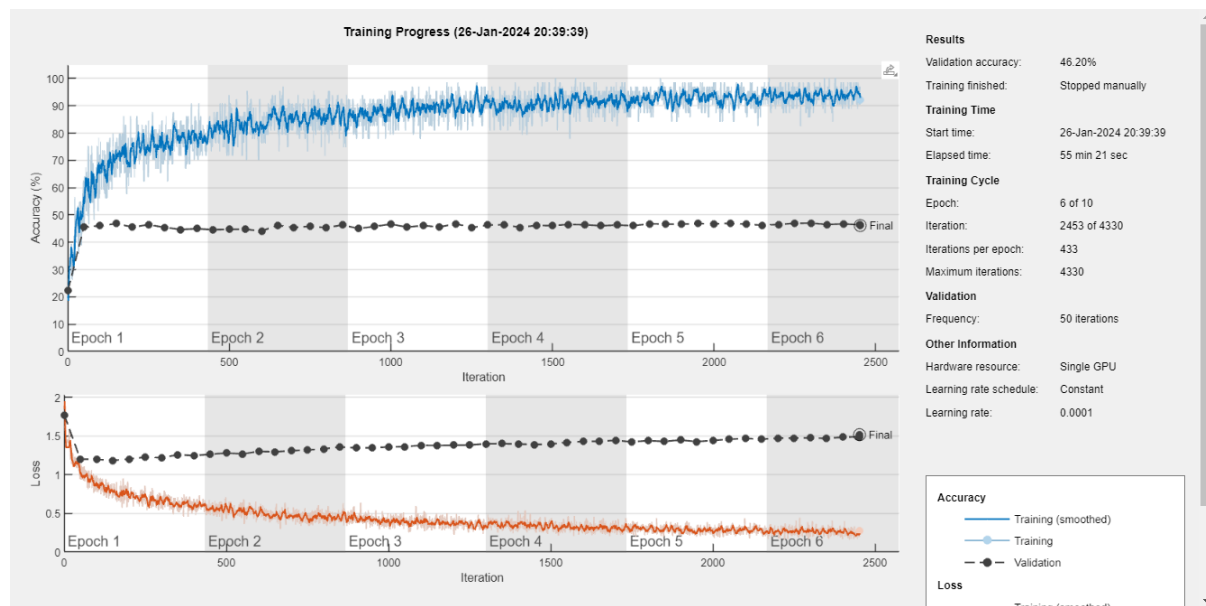
GoogleNet

The training line (blue line) is indicating that the data is 100% overfitting in this case with the worst accuracy of 47.18%

Resnet-18

Resnet-50

As we can see that we used two different resnet networks which are resnet-18 and resnet-50. In the above image, training data (blue line) shows the data is 100% overfitted and on the other hand, training data bit suffers from a over fitting and the loss curve in the training set is also indicative of over fitting of data. Furthermore, the accuracy of resnet-18 is 50.58%. In contrast, the accuracy of resnet-50 is less than resnet-18 which is 46.20%. Overall both of these networks are having worse performance.

SqueezeNet
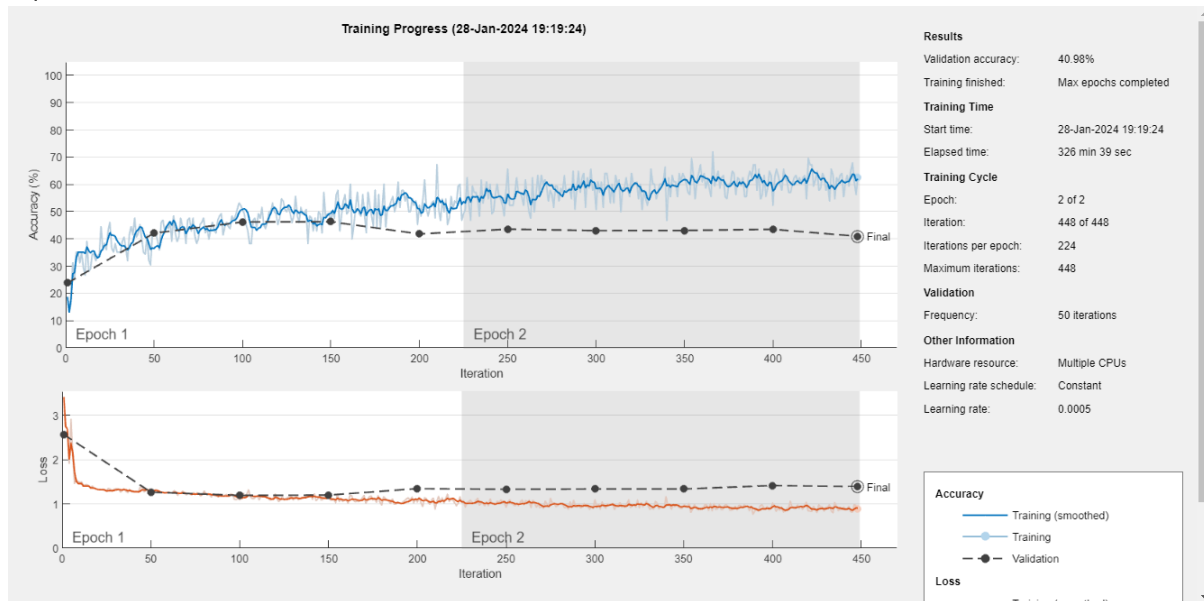
VGG-19
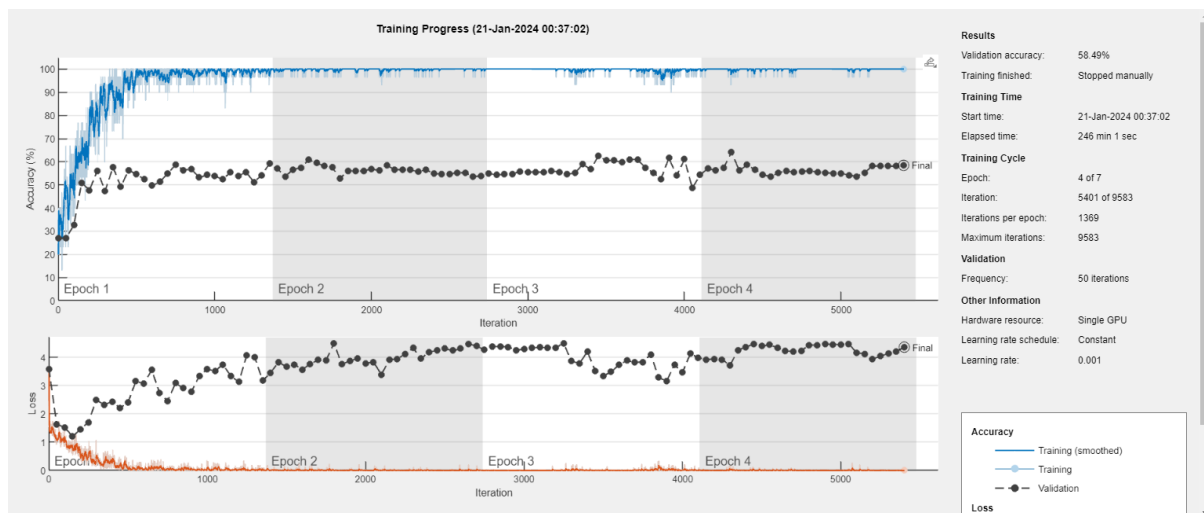
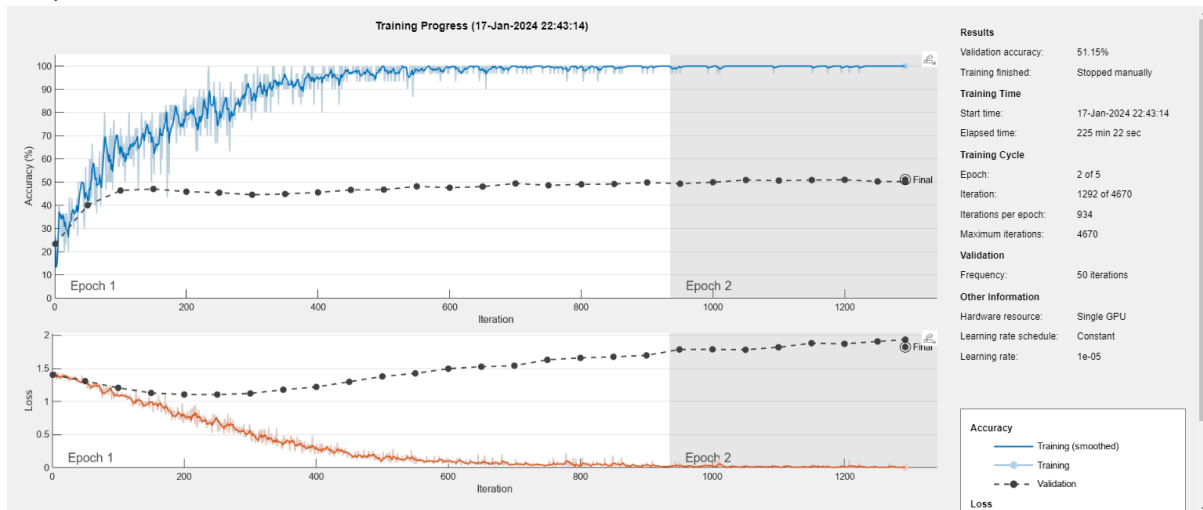We can see that the training data start overfitting from the very first epoch just like we see in googleNet. However this networks performs very well as compared to other networks with accuracy of approximately 60%.

Xception

As we see that this network is also following the same curve of training data which we see above. It is also overfitting the data in its first epoch. This network achieving the accuracy of 51.15% which is the second best from all above seven networks.

Evaluation Networks

Validation	AlexNet	DenseNet201	Efficient-b0	GoogleNet	Resnet-18	Resnet-50	Squezze Net	VGG-19	Xception
Accuracy	30%	47.49%	49.23%	47.18%	50.58%	46.20%	40.98	58.49%	51.15%
Loss	5	1.2	1.4	6.1	1.45	1.5	1.1	4.1	1.9

Testing Phase

After completing training phase, the next step involves testing them on unknown data. To do that, MATLAB Script were created. The following table shows the most important metrics recorded by each network on the test set.

Network	Class	Accuracy	Precision	Recall	F1-Score
AlexNet	Mild	0.47	0.42	0.44	0.43
	Moderate		0.4	0.39	0.4
	Severe		0.48	0.39	0.43
	Critical		0.63	0.8	0.7
DenseNet201	Mild	0.44	0.44	0.39	0.41
	Moderate		0.32	0.19	0.24
	Severe		0.35	0.48	0.40
	Critical		0.69	0.80	0.74
Efficient-b0	Mild	0.49	0.44	0.31	0.36
	Moderate		0.41	0.33	0.37
	Severe		0.42	0.55	0.48
	Critical		0.73	0.88	0.8
GoogleNet	Mild	0.44	0.4	0.1	0.17
	Moderate		0.35	0.32	0.34
	Severe		0.37	0.69	0.48
	Critical		0.9	0.7	0.79
Resnet-18	Mild	0.44	0.42	0.39	0.4
	Moderate		0.33	0.24	0.28
	Severe		0.36	0.38	0.37
	Critical		0.65	0.94	0.77
Resnet-50	Mild	0.49	0.46	0.47	0.46
	Moderate		0.38	0.29	0.33
	Severe		0.46	0.43	0.44
	Critical		0.67	0.95	0.78
SquezeNet	Mild	0.40	0.46	0.30	0.36
	Moderate		0.37	0.29	0.32
	Severe		0.36	0.27	0.31
	Critical		0.43	0.95	0.59
VGG-19	Mild	0.58	0.42	0.24	0.30
	Moderate		0.37	0.38	0.38
	Severe		0.39	0.50	0.44
	Critical		0.79	0.89	0.84
Xception	Mild	0.44	0.44	0.42	0.43
	Moderate		0.40	0.22	0.28
	Severe		0.40	0.64	0.49
	Critical		0.63	0.49	0.55

Code

Freezing Weights & Pre-Processing

https://github.com/DarioOz/PreProcessing_medical-image