UNIVERSITÀ degli STUDI di CATANIA

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 SEBASTANIO 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

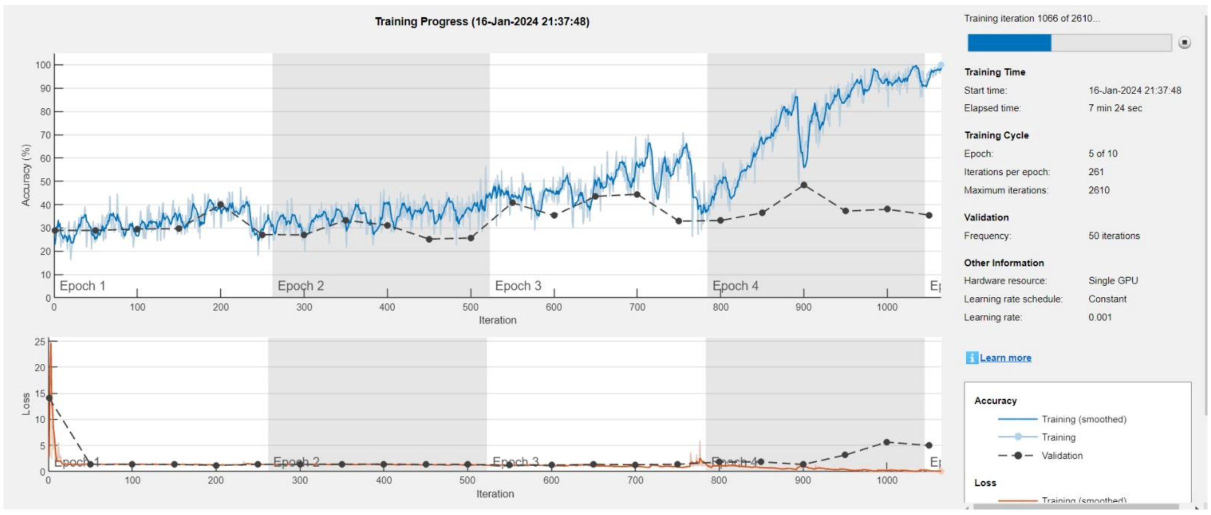
We have used different types of hyperparameters with different networks. Following are the best hyperparameters according to the networks.

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



Training Progress (16-Jan-2024 21:37:48)

Training iteration 1066 of 2610...

Training Time

Start time:	16-Jan-2024 21:37:48
Elapsed time:	7 min 24 sec

Training Cycle

Epoch:	5 of 10
Iterations per epoch:	261
Maximum iterations:	2610

Validation

Frequency:	50 iterations

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.001

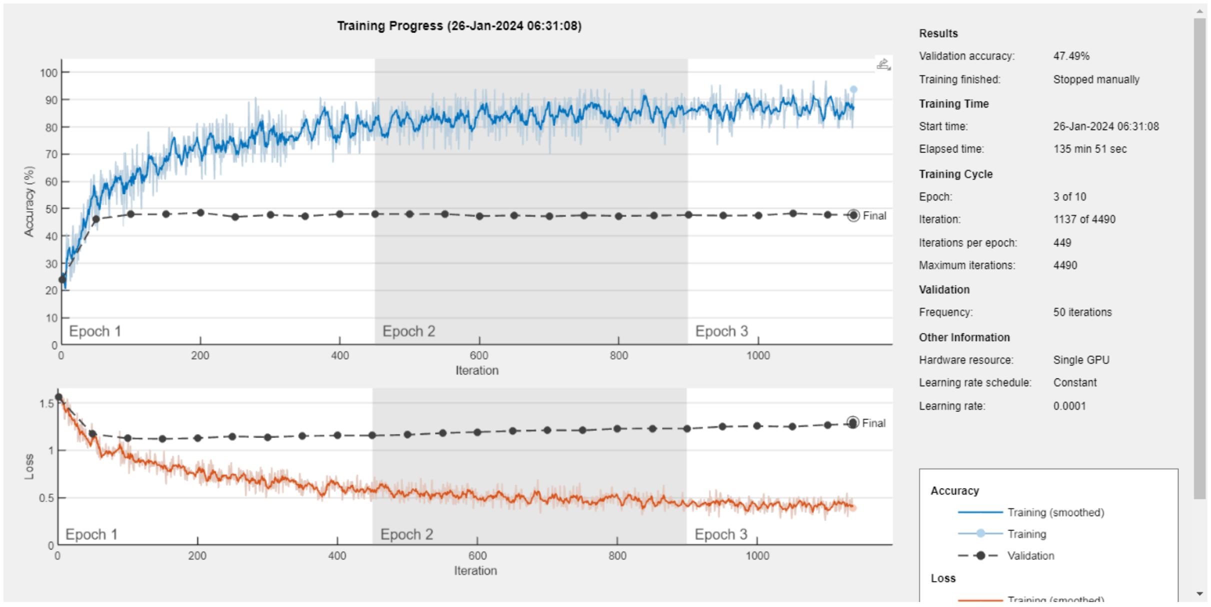
Learn more

Accuracy
- Training (smoothed)
- Training
- Validation

Loss

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



Training Progress (26-Jan-2024 06:31:08)

Results

Validation accuracy:	47.49%
Training finished:	Stopped manually

Training Time

Start time:	26-Jan-2024 06:31:08
Elapsed time:	135 min 51 sec

Training Cycle

Epoch:	3 of 10
Iteration:	1137 of 4490
Iterations per epoch:	449
Maximum iterations:	4490

Validation

Frequency:	50 iterations

Other Information

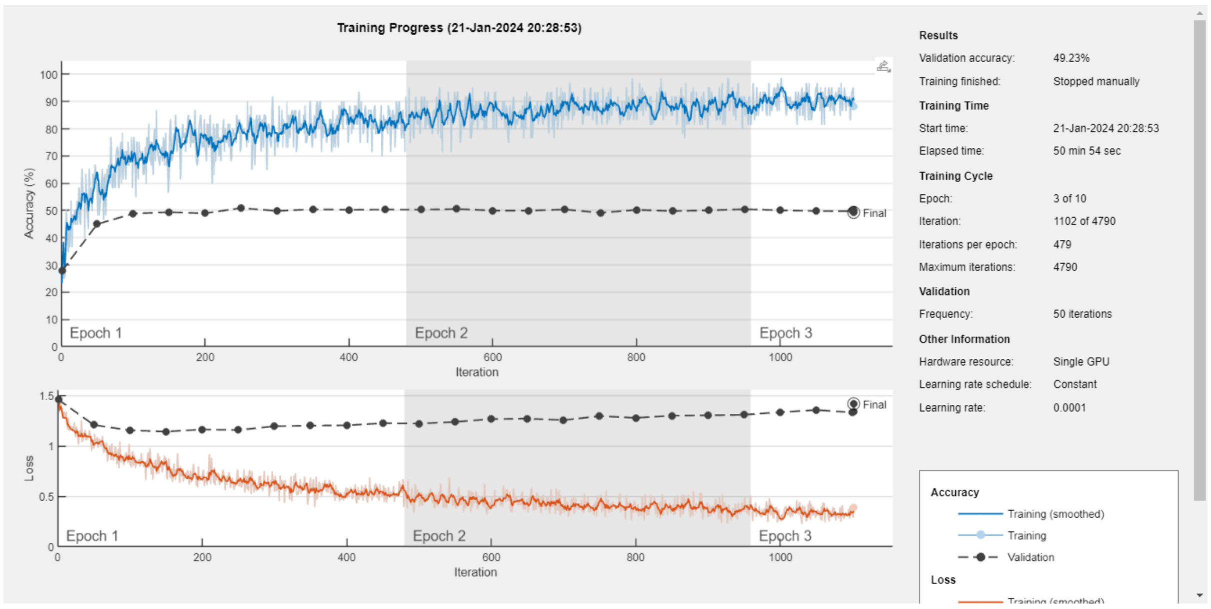
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Accuracy
- Training (smoothed)
- Training
- Validation

Loss

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



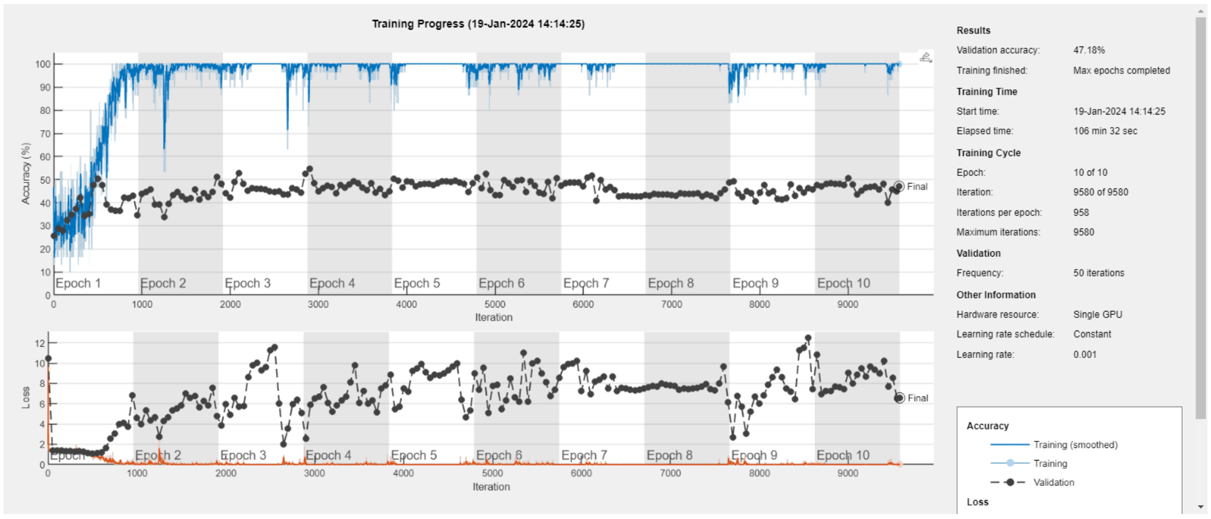
Results	
Validation accuracy:	49.23%
Training finished:	Stopped manually
Training Time	
Start time:	21-Jan-2024 20:28:53
Elapsed time:	50 min 54 sec
Training Cycle	
Epoch:	3 of 10
Iteration:	1102 of 4790
Iterations per epoch:	479
Maximum iterations:	4790
Validation	
Frequency:	50 iterations
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Accuracy
- Training (smoothed)
- Training
- Validation

Loss
- Training (smoothed)

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



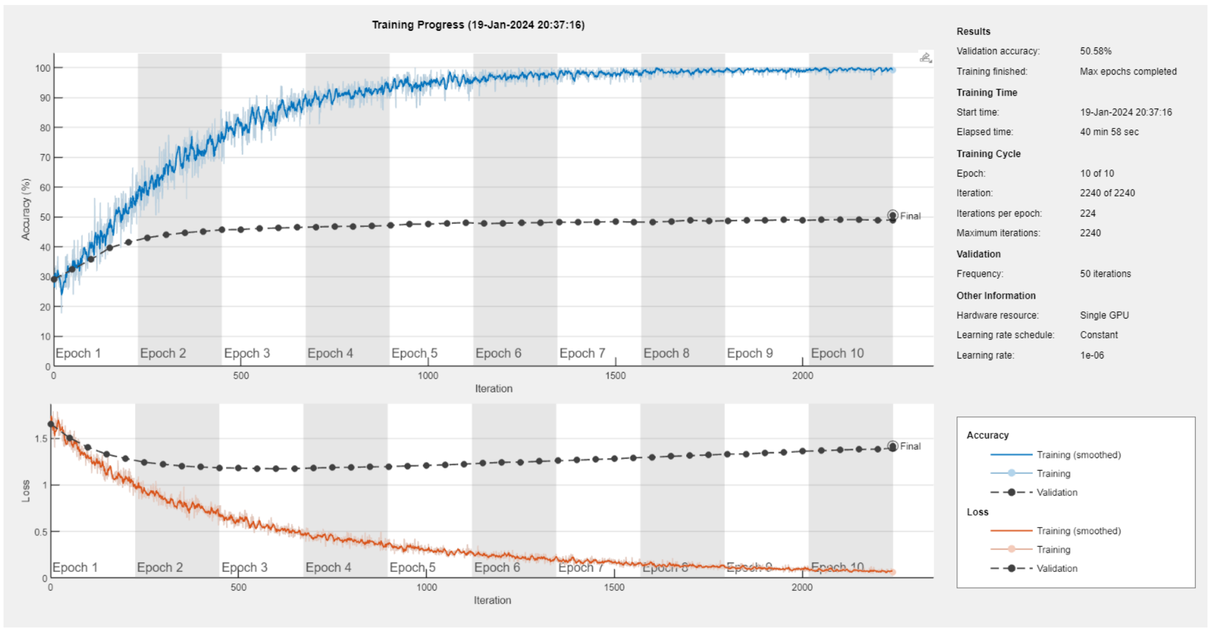
Results	
Validation accuracy:	47.18%
Training finished:	Max epochs completed
Training Time	
Start time:	19-Jan-2024 14:14:25
Elapsed time:	106 min 32 sec
Training Cycle	
Epoch:	10 of 10
Iteration:	9580 of 9580
Iterations per epoch:	958
Maximum iterations:	9580
Validation	
Frequency:	50 iterations
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.001

Accuracy
- Training (smoothed)
- Training
- Validation

Loss

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



Results

Validation accuracy:	50.58%
Training finished:	Max epochs completed

Training Time

Start time:	19-Jan-2024 20:37:16
Elapsed time:	40 min 58 sec

Training Cycle

Epoch:	10 of 10
Iteration:	2240 of 2240
Iterations per epoch:	224
Maximum iterations:	2240

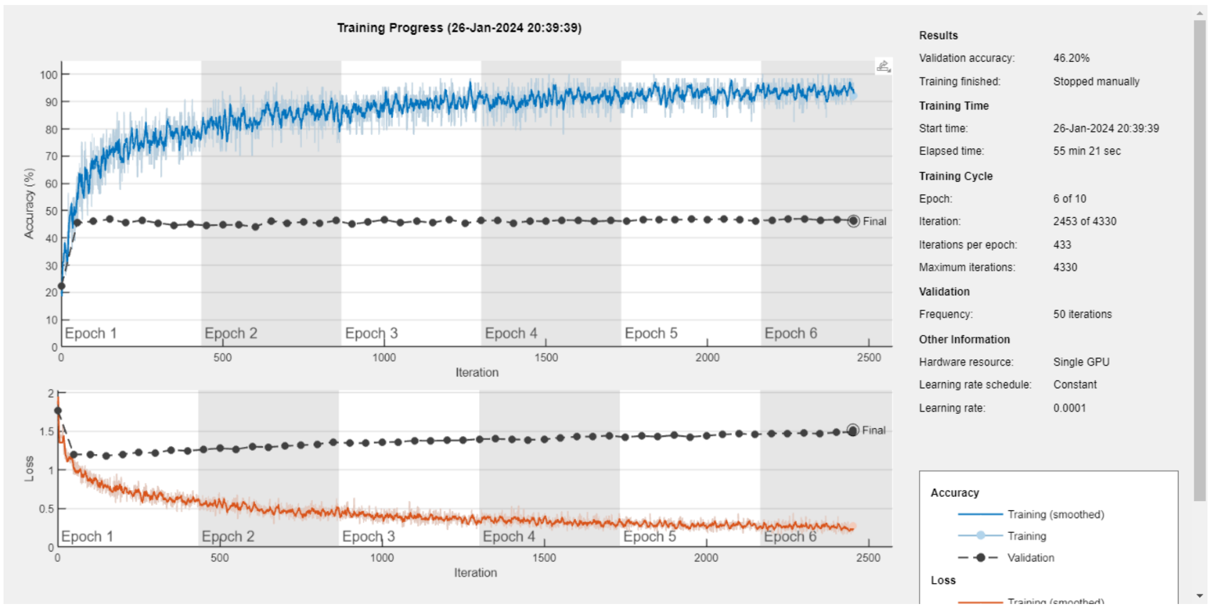
Validation

Frequency:	50 iterations

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	1e-06

Resnet-50



Results

Validation accuracy:	46.20%
Training finished:	Stopped manually

Training Time

Start time:	26-Jan-2024 20:39:39
Elapsed time:	55 min 21 sec

Training Cycle

Epoch:	6 of 10
Iteration:	2453 of 4330
Iterations per epoch:	433
Maximum iterations:	4330

Validation

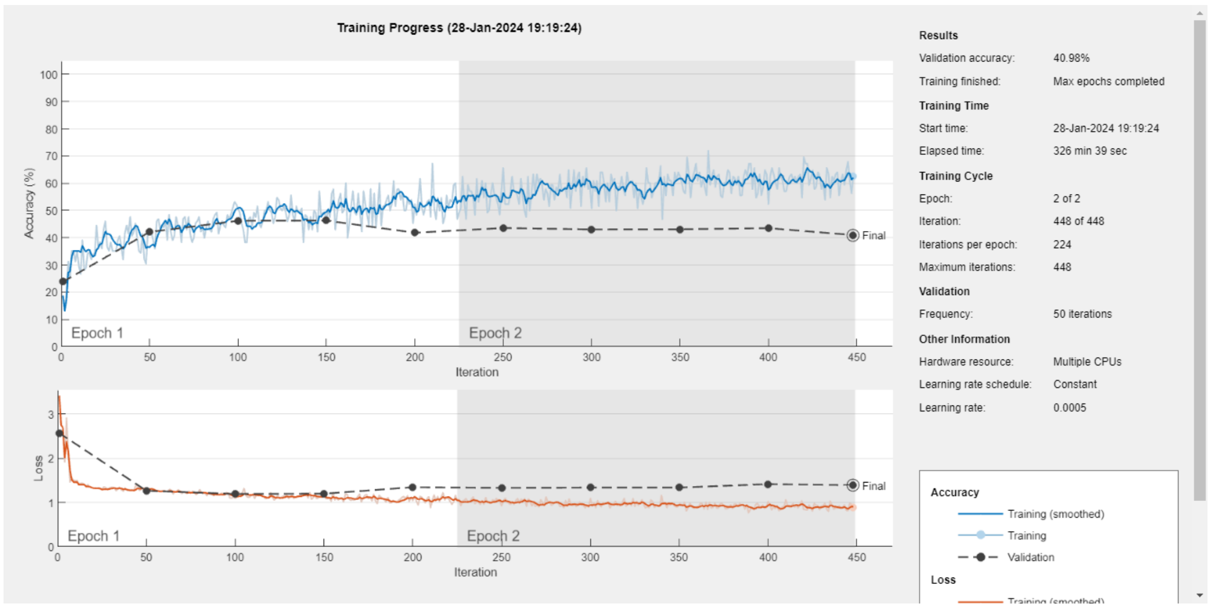
Frequency:	50 iterations

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

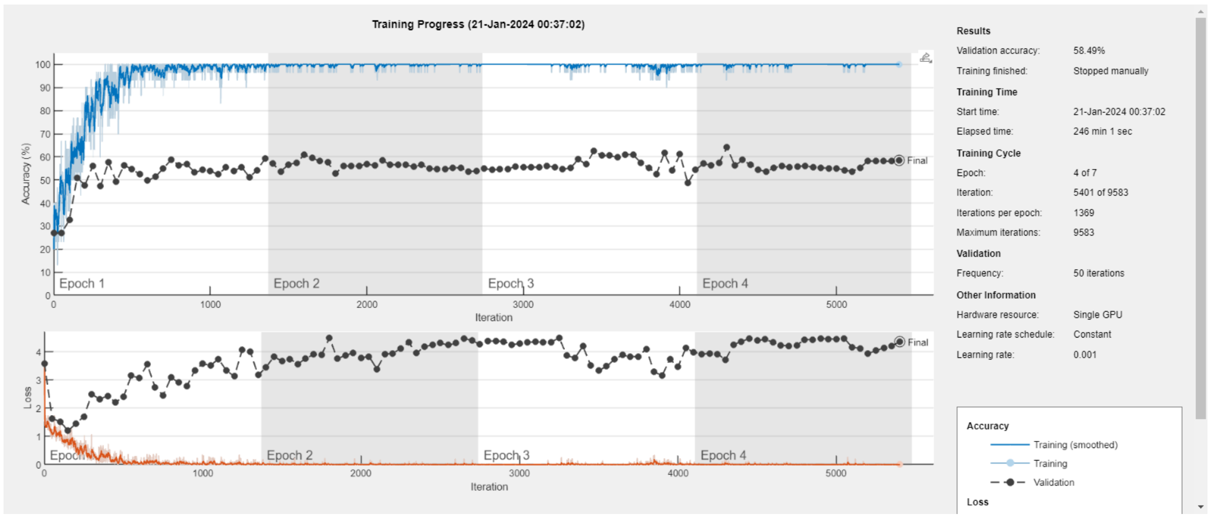
As we can see that we used two different resnet networks which are resnet-18 and resnet-50. In the above image, training data (blue line) shows the data is 100% overfitted and on the other hand, training data bit suffers from a over fitting and the loss curve in the training set is also indicative of 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



Results
Validation accuracy: 40.98%
Training finished: Max epochs completed
Training Time
Start time: 28-Jan-2024 19:19:24
Elapsed time: 326 min 39 sec
Training Cycle
Epoch: 2 of 2
Iteration: 448 of 448
Iterations per epoch: 224
Maximum iterations: 448
Validation
Frequency: 50 iterations
Other Information
Hardware resource: Multiple CPUs
Learning rate schedule: Constant
Learning rate: 0.0005

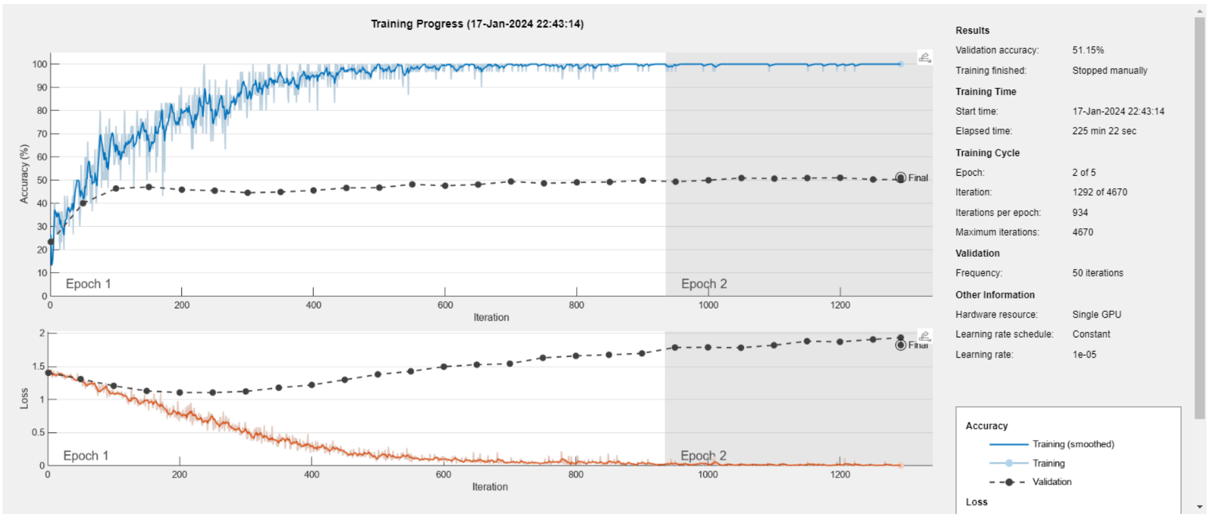
VGG-19



Results
Validation accuracy: 58.49%
Training finished: Stopped manually
Training Time
Start time: 21-Jan-2024 00:37:02
Elapsed time: 246 min 1 sec
Training Cycle
Epoch: 4 of 7
Iteration: 5401 of 9583
Iterations per epoch: 1369
Maximum iterations: 9583
Validation
Frequency: 50 iterations
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.001

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



Results

Validation accuracy: 51.15%

Training finished: Stopped manually

Training Time

Start time: 17-Jan-2024 22:43:14

Elapsed time: 225 min 22 sec

Training Cycle

Epoch: 2 of 5

Iteration: 1292 of 4670

Iterations per epoch: 934

Maximum iterations: 4670

Validation

Frequency: 50 iterations

Other Information

Hardware resource: Single GPU

Learning rate schedule: Constant

Learning rate: 1e-05

Accuracy

Training (smoothed)

Training

Validation

Loss

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