Deep Learning Exam, End of Semester Project Report

Name: **Tibabwetiza Muhanguzi**

Reg No: **2021/HD05/2312U**

Std No: **2100702312**

Email: **tibajoel90@gmail.com**

Abstract

This report entails the steps taken to perform the deep learning exam coursework which entailed the design of a model to perform image classification on images regarding chest opacities. The target classes to acquire were two, either normal or sick.

The tool used in this work was Pytorch which is a machine learning framework developed by Meta an AI firm. It is used on both computer vision and natural language processing tasks. In this particular work, we fine tune the convolutional neural network in order to predict the classes for a certain set of images.

1. Introduction

The images were loaded from a dropbox link into a google colab notebook. From here, the images were then sorted into training, validation and test sets using a package; split folders. This enables it to automatically allocate the images in accordance with the specified percentages for the image distribution[2].

The next step to be carried out was data augmentation for the images and normalization for the training and validation sets. Then a pretrained model Resnet18 was loaded and the final fully connected layer was reset. This is a technique known as transfer learning[1].

The training epochs were varied between 25 to 100 in order to vary the scores of the metrics for the performance of the model.

2. Data split

The figure 2.1 below shows the structure of the directories in the google colab after file processing is done to separate the different images that will be used to train the model.

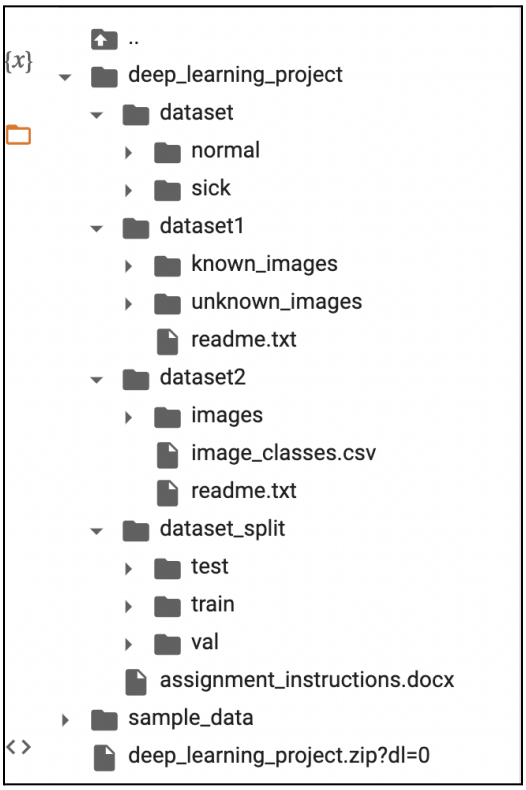

Figure 2.1: Directory Split for the data

3. Model Training and Evaluation

The model was trained on the dataset with a variation of the epochs between 25 and 100. The best value accuracy acquired from training the model was 0.98. The figure 3.1 below shows a snapshot of the epoch training frame. The established best value accuracy for the model was 98.59% which was a very good score for the trained model.

```
Epoch 23/24
----------
train Loss: 0.2763 Acc: 0.8798
val Loss: 0.0959 Acc: 0.9648

Epoch 24/24
----------
train Loss: 0.2394 Acc: 0.8958
val Loss: 0.0916 Acc: 0.9718

Training complete in 2m 6s
Best val Acc: 0.985915
```

Figure 3.1: Model training

Figure 3.2 Visualizes some of the output of the model for the images it was classifying while figure 3.3 Shows the saved model as a resnet18gputrained.pt file. This file would later be loaded in order to be used to classify the images from dataset 2 whose class was unknown.

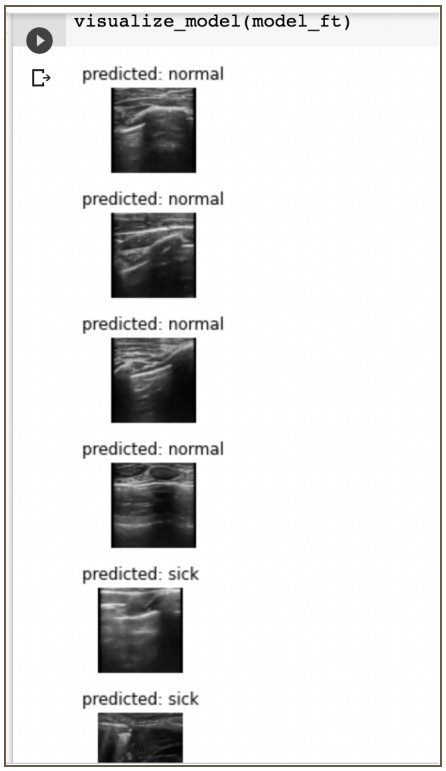

Figure 3.2: Visualize model predictions

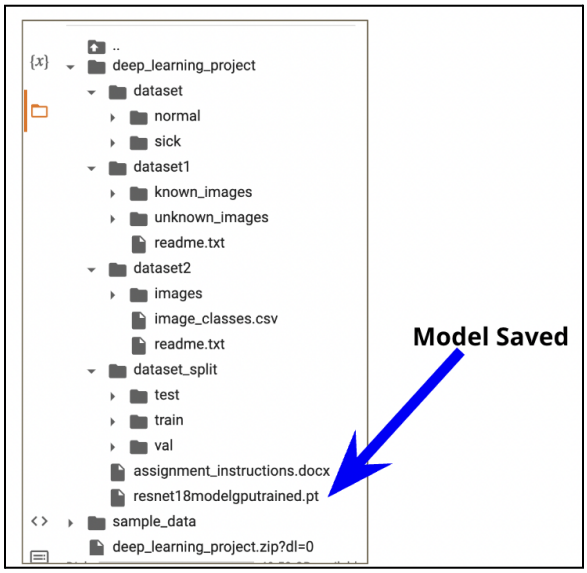

Figure 3.3: Saved Model

After training the model, the model was subjected onto the unknown images in dataset 1 in order to classify them

accordingly. The saved model was loaded from the saved path in order to perform this task. The resulting corresponding classes for the unknown images were generated and stored in a csv file which would later be uploaded onto the github repo for this particular assignment. The figure 3.4 shows the model loaded from storage, a snapshot of the classes assigned to some of the images in the unknown images directory as well as the generated csv file.

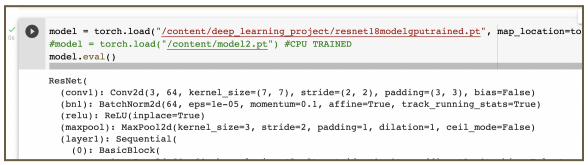

Figure 3.4: Load Saved Model

	files	predictions
0	6.png	sick
1	24.png	sick
2	16.png	sick
3	30.png	sick
4	45.png	normal
5	13.png	normal
6	22.png	sick
7	21.png	normal
8	23.png	normal
9	25.png	normal
10	9.png	normal
11	8.png	sick
12	40.png	sick
13	5.png	normal
14	39.png	normal
15	10.png	sick
16	14.png	sick
17	15.png	normal
18	19.png	normal
19	27.png	normal
20	17.png	normal
21	41.png	normal
22	11.png	normal
23	36.png	sick
24	32.png	sick
25	12.png	sick
26	49.png	normal

Figure 3.5: Classifying unknown images

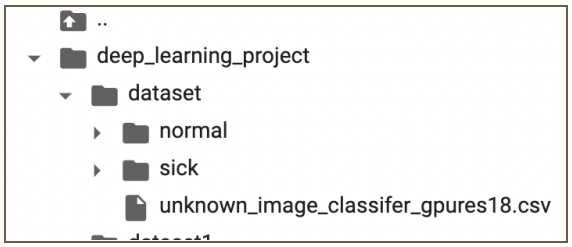

Figure 3.6 Generated CSV file with classes assigned to unknown images

4. Predict Images from Different Conditions

The next step was to test the model to score how best it would perform on the images that were captured under different conditions. The predictions were made as shown in the figure below and the corresponding scores were also acquired. The classification was done having trained the model at both 25 and 100 epochs and some key observations were made.

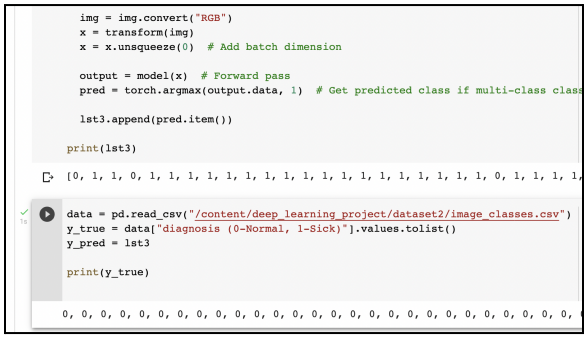

Figure 4.1: Predictions for the classes of images captured under different conditions

For 25 epochs, the following classification report was generated giving an accuracy of 0.5, a recall of 0.91 as well as a precision of

0.5 for predicting the sick class which was in our case the positive class.

Classification Report

	precision	recall	f1-score	support
Normal	0.50	0.09	0.15	112
Sick	0.50	0.91	0.65	112
accuracy			0.50	224
macro avg	0.50	0.50	0.40	224
weighted avg	0.50	0.50	0.40	224

Figure 4.2: results for 25 epochs.

```
[[102  10]
 [102  10]]
```

Figure 4.3: Confusion matrix for 25 epochs

For the model trained on 100 epochs, the accuracy was established as 0.49, the precision 0.49 and the recall as 0.93 for predicting the sick class.

Classification Report

	precision	recall	f1-score	support
Normal	0.38	0.04	0.08	112
Sick	0.49	0.93	0.64	112
accuracy			0.49	224
macro avg	0.44	0.49	0.36	224
weighted avg	0.44	0.49	0.36	224

Figure 4.4: classification report for 100 epochs

```
[[104   8]
 [107   5]]
```

Figure 4.5: Confusion matrix for 100 epochs

After having established this, the work was then uploaded onto github as shown in the figure 4.6 below.

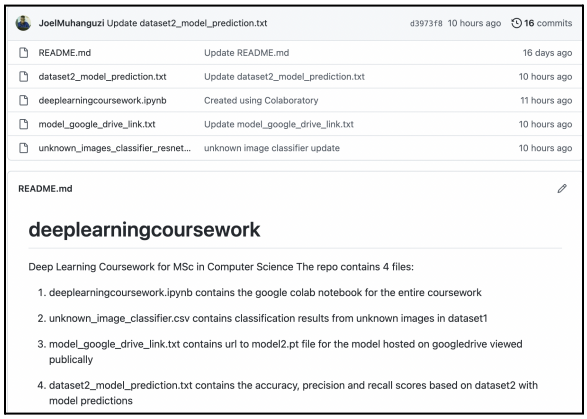

Figure 4.6: github repo containing the results from the assignment[2]

Conclusion

Training a model on aGPU gives better results compared to training using a CPU. The more the number of epochs the better the accuracy. Training a model on a GPU provides shorter training times compared to using the CPU; A GPU typically trains on 25 epochs for about 2 minutes and 15 seconds while training using a CPU takes about an hour and a half.

The trained model also struggles a bit to classify images captured under different conditions. The recommendation made from this work is to consider using resnets with deeper layers and assess their performance against different images.

References

[1]“Transfer Learning for Computer Vision Tutorial — PyTorch Tutorials 1.12.1+cu102 documentation.” https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html (accessed Sep. 17, 2022).

[2]JoelMuhanguzi, “GitHub - JoelMuhanguzi/deeplearningcoursework: Deep Learning Coursework for MSc in Computer Science,” *GitHub*. https://github.com/JoelMuhanguzi/deeplearningcoursework (accessed Sep. 17, 2022).