CS6600: Project 02- Report

Shubham Swami – A02315672

Introduction:

I reviewed the data I selected during the project proposal and I realized that it has very few chest Xray images for COVID positive people. I moved to the source GitHub repository and found that the GitHub repository was updated frequently but the Kaggle dataset was not, so I found another dataset with more data on Kaggle at this link: https://www.kaggle.com/praveengovi/coronahack-chest-xraydataset.

I also tried uploading the downloaded dataset to Box but for some reason it failed a couple of times. So, the pck files were smaller than original dataset and hence I uploaded it to BOX:

Box link for pck files: https://usu.box.com/s/fsk9glwbjpbftfw36hzh43ozpy08649i.

Box link for all my trained models: https://usu.box.com/s/ml8rfm0q3d3jfpreba9c2bbi9412ol5r.

Deliverables:

random_forests_and_decision_trees.py	Code for applying random forests to my dataset
covid_rf_uts	Test file for random forests.
project2_image_anns	Code for implementing the ANN model on dataset
project2_image_anns	Code for implementing the CNN models on the dataset
script.py	Code to train models or both CNN and ANN.
covid_ensemble.py	Code for implementing ensembles from the best models.
Data_conversion.py	Manipulates the Kaggle data and returns pck files.
Readme.txt	Well described how to run the code in this project.
Load_nets.py	To load the saved tfl models.
Test_load_nets.py	Obtain accuracies from saved models.

Some other helper code is also in the project. In the proposal I committed to Image classification using CNN and Random Forests. However, I am submitting ANN's, Ensembles as well just to showcase my efforts and my own work.

In the proposal comments all I see is not mentioning the real risks, I mentioned GTA duties and assignments along with my research project. I completed this project on time.

Another point I noticed was, you wanted to see my own work and no collaborations, I can guarantee that this was all my work, and I don't know who else was working on this project. I can assure that I implemented the entire project based on the Class notes, slides and Hw code along with project 01 Code. My models accuracies and other submitted deliverables are unique.

Schedule Followed and the Description of what I achieved:

Dates	Description
11/09 – 11/16	1. Downloaded the dataset and wrote the code to obtain the data in pck format. 2. Revised the slides from Hw07 on Random Forests and modified the code to get random forests working on my data. 3. This took me less time than expected so I decided that I will implement some ANN models too which I did not mention in my proposal.
11/17 - 11/23	1. Starting with implementing the ANN models on the data set and training it on my lab PC so that I can work on CNN models at my laptop. 2. Training ANN models to obtain the best model with the highest accuracy. 3. Started implementing the CNN models in parallel to ANN's. 4. Trained CNN models to obtain highest accuracy.
11/24 – 11/30	1. Using my best model and experimenting by changing learning rates, epochs, batch size etc. to find out if I can find any better accuracies.

	2. I also wrote the code to plot some data while training so that I can present a better report in submission.
12/01 -12/08	1. Implementing ensembles and checking if they are any better than the models I have. 2. Implemented a readme file and completed almost all the deliverables as the next week would be busy because of final exams, assignment submissions and grading wrap up for my GTA duties.
11/09 -12/14	1. Reviewing all the work I did to validate my submission and draft a report for this project. 2. Completed the report and I think I am done with this project.

1. Data Manipulation:

The data was very small when we compare it to the bee dataset.
I used the metadata.csv file and formatted the data in pck form so that I can work in a similar way as I did in Project 01. The data basically contains Xray images that are *grayscale*, so I reshaped the data to **[64,64]** and **[64,64,1] accordingly**.

I obtained test data by the following split function:

```
train_images_X, test_images_X, train_images_Y, test_images_Y = train_test_split(train_images_X, train_images_Y,
                                                    random_state=0,                    test_size=0.25,
                                                    shuffle=True)
```

2. Random Forests

How to run the code:

The code for random forests is in "random_forests_and_decision_trees.py" and "covid_data_loader_rf.py" contains the code for loading the dataset as we did in mnist_loader.py in hw07 and this code can be run through the "covid_rf_uts.py".

Run: "python3.7 covid_rf_uts.py" You can also run **covid_rf_uts.py directly through PyCharm.**

Results:

Decision Tree Performance (5 random DT's)
Accuracies
0.73
0.73
0.74
0.73
0.71

Performance of random forest	
Number of trees	**Accuracies**
10	0.75
20	0.75
30	0.74
40	0.73
50	0.72
60	0.75
70	0.72
80	0.7
90	0.74
100	0.74

3. ANN Models:

How to run the code:

Let's move to getting the required ANN models:

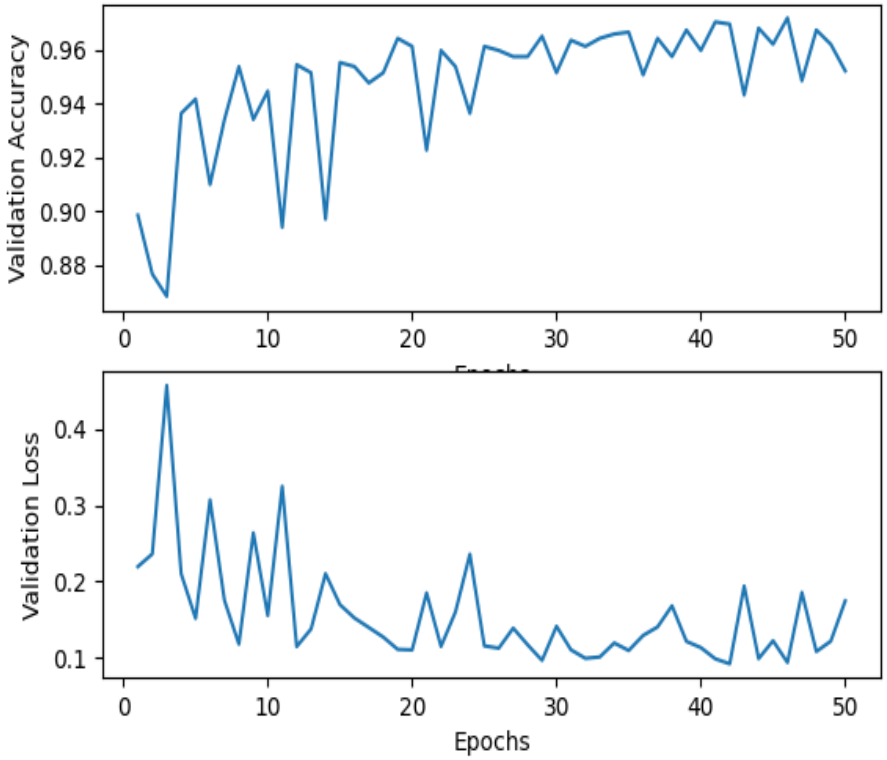
a) (Optional) Go to script.py and edit the epochs and batch size in the get_ann_models().

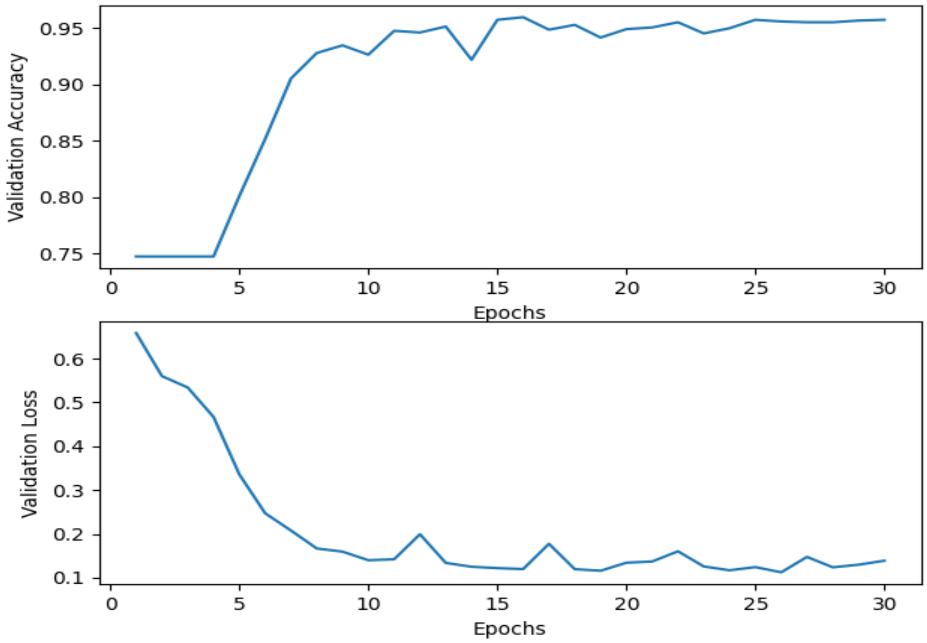
b) (Optional) Change the layers and other parameter like learning rate in project2_image_anns.py.
c) Uncomment line 23 (#get_ann_models()).
d) run python3.7 script.py to train the model and obtain validation accuracy. or run scrpt.py directly through pycharm.

Results:
I am submitting top-3 ANN models, however for effort If you need to see the amount of training I did, I will upload the 9-10 best models of ANN on Box. The excel for training is at the last page.

image_ann_model_001.tfl	**Validation Accuracy: 0.8205128205128205**
image_ann_model_002.tfl	**Validation Accuracy: 0.8125**
image_ann_model_003.tfl	**Validation Accuracy: 0.7996794871794872**

I explored tflearn and looks like it doesn't store history as Keras does and hence the model.fit() function gave me nothing to plot graphs. I used matplotlib to create graphs for validation accuracy vs epochs and loss vs epochs for the top 2 ANN's.





4. CNN Models:

How to run the code:

Let's move to getting the required CNN models:

a) (Optional) Go to script.py and edit the epochs and batch size in the get_cnn_models().
b) (Optional) Change the layers and other parameter like learning rate in project2_image_cnns.py.
c) Uncomment line 23 (#get_cnn_models()).
d) run python3.7 script.py to train the model and obtain validation accuracy. or run scrpt.py directly through pycharm.

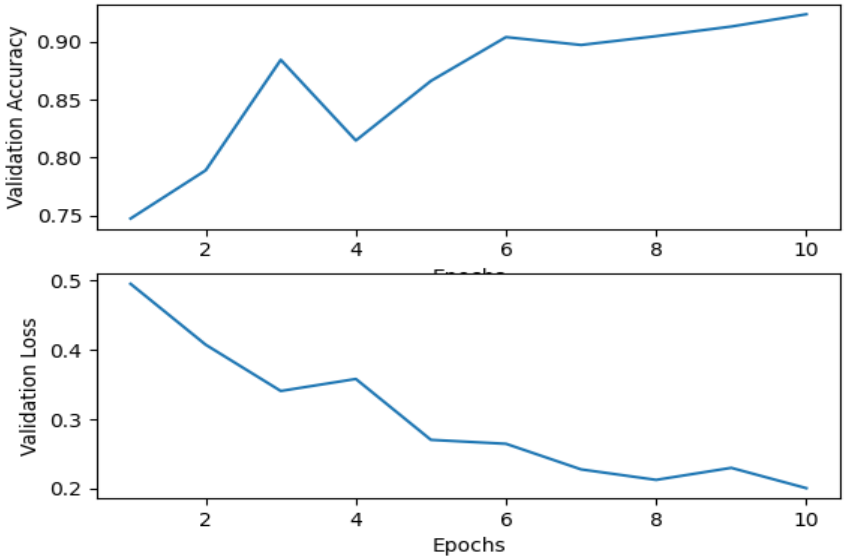
Results:

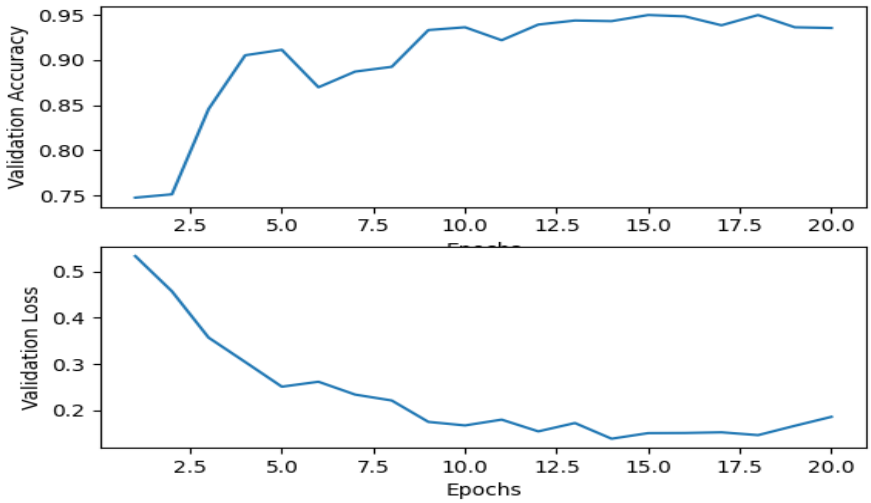
I am submitting top-3 CNN models, however for effort If you need to see the amount of training I did, I will upload the 9-10 best models of CNN on Box. The excel for training is at the last page.

image_cnn_model_001.tfl	**Validation Accuracy: 0.8766025641025641**
image_cnn_model_002.tfl	**Validation Accuracy: 0.7932692307692307**
image_cnn_model_003.tfl	**Validation Accuracy: 0.8189102564102564**

I explored tflearn and looks like it doesn't store history as Keras does and hence the model.fit() function gave me nothing to plot graphs. I used matplotlib to create graphs for validation accuracy vs epochs and loss vs epochs for the top 2 ANN's.

Graphs for top 2 CNN's:





5. Ensembles:

I implemented Ensembles too, but they did come out better than the individual models. I tried using combinations of my best models and decision trees. But I think the best accuracies were obtained by individual models.

How to run the code:

"covid_ensemble.py" has ensembles for ANN's, CNN's and ANN's + CNN's + Decision Trees respectively. Just run using "python3.7 covid_ensemble.py" or covid_ensemble.py run directly through pycharm.

CNN ensemble Accuracy: 0.844551282051282
ANN ensemble Accuracy: 0.8173076923076923
ANN + CNN + Decision Tree Ensemble Accuracy: 0.8173076923076923

Conclusions:

I think the best on this dataset was performed by CNN with a validation accuracy 0.87. I tried my best to achieve the better results and I hope I did show my efforts for this project. This project more or less includes almost everything that was taught in the class. My models can be seen through the box link I gave in this report and some training accuracies are included below. This class offered me a lot of learning. Thank you Professor!

Below are the results from my training for both ANN and CNN.

6. Training Documentation and Experiments:

I did a lot of training by varying the number of layers, batch size, epochs and learning rate and I am documenting some of my top results. I have the entire excel file and It will make this report too lengthy if I include all the results. I will submit a few other models as well along with the best models.

1. ANN's:

The best results were obtained by varying the learning rate (0.001 and 0.01). Accuracies for both these learning rates are mentioned along with different epochs and batch size.

Model Description	Epochs	Batch Size	Learning Rate	Accuracy	Learning Rate	Accuracy
2 layers	10	8	0.001	0.753205	0.01	0.625
	10	16	0.001	0.72435	0.01	0.625
	10	32	0.001	0.727	0.01	0.625
	10	64	0.001	0.681	0.01	0.625
	20	8	0.001	0.74679	0.01	0.625
	20	16	0.001	0.770833	0.01	0.375
	20	32	0.001	0.719551	0.01	0.625
	20	64	0.001	0.67948	0.01	0.625
	30	8	0.001	0.74679	0.01	0.625
	30	16	0.001	0.7051	0.01	0.625
	30	32	0.001	0.69871	0.01	0.625
	30	64	0.001	0.7051	0.01	0.375
	40	8	0.001	0.7067	0.01	0.625
	40	16	0.001	0.7019	0.01	0.625
	40	32	0.001	0.79807	0.01	0.625
	40	64	0.001	**0.79967**	0.01	0.625
	50	8	0.001	0.70673	0.01	0.625
	50	16	0.001	0.743	0.01	0.625
	50	32	0.001	0.7131	0.01	0.625
	50	64	0.001	0.79807	0.01	0.625

Model Description	Epochs	Batch Size	Learning Rate	Accuracy	Learning Rate	Accuracy
3 layers	10	8	0.001	0.791666	0.01	0.7676
	10	16	0.001	0.708333	0.01	0.658
	10	32	0.001	**0.79647**	0.01	0.679
	10	64	0.001	0.7788	0.01	0.8189

	20	8	0.001	0.71955	0.01	0.7387
	20	16	0.001	0.75	0.01	0.79006
	20	32	0.001	0.722	0.01	0.74358
	20	64	0.001	0.7532	0.01	0.79487
	30	8	0.001	0.7435	0.01	0.78365
	30	16	0.001	0.77724	0.01	0.69711
	30	32	0.001	0.7676	0.01	0.711
	30	64	0.001	0.759	0.01	0.7548
	40	8	0.001	0.74038	0.01	0.7067
	40	16	0.001	0.75961	0.01	0.78846
	40	32	0.001	0.77403	0.01	0.80128
	40	64	0.001	0.75641	0.01	0.79807
	50	8	0.001	0.769	0.01	0.7868
	50	16	0.001	0.774	0.01	**0.82051**
	50	32	0.001	0.7291	0.01	0.7051
	50	64	0.001	0.7451	0.01	0.78846

Model Description	Epochs	Batch Size	Learning Rate	Accuracy	Learning Rate	Accuracy
6 layers	10	8	0.001	0.7532	0.01	**0.7804**
	10	16	0.001	0.7371	0.01	0.7467
	10	32	0.001	0.7371	0.01	0.70032
	10	64	0.001	0.66025	0.01	0.7291
	20	8	0.001	0.73076	0.01	0.7115
	20	16	0.001	0.732371	0.01	0.7387
	20	32	0.001	0.807	0.01	0.7179
	20	64	0.001	0.7724	0.01	0.6381
	30	8	0.001	0.7051	0.01	0.7211
	30	16	0.001	0.708	0.01	0.7227
	30	32	0.001	**0.8125**	0.01	0.73
	30	64	0.001	0.7996	0.01	0.738
	40	8	0.001	0.78846	0.01	0.777
	40	16	0.001	0.7788	0.01	0.759
	40	32	0.001	0.769	0.01	0.743
	40	64	0.001	0.769	0.01	0.748
	50	8	0.001	0.7788	0.01	0.766
	50	16	0.001	0.754	0.01	0.75
	50	32	0.001	0.78044	0.01	0.7355
	50	64	0.001	0.714	0.01	0.7419

2. CNN's

The best results were obtained by varying the learning rate (0.001 and 0.01). Accuracies for both these learning rates are mentioned along with different epochs and batch size.

No dropout used						
Model Description	**Epochs**	**Batch Size**	**Learning Rate**	**Accuracy**	**Learning Rate**	**Accuracy**
1 conv layer	10	8	0.001	0.7541	0.01	0.67
1 pool layer	10	16	0.001	0.67	0.01	0.679
2 fc layers	10	32	0.001	0.625	0.01	0.625
	10	64	0.001	0.75	0.01	0.7467
	20	8	0.001	0.625	0.01	0.75
	20	16	0.001	0.8125	0.01	0.625
	20	32	0.001	0.762	0.01	0.375
	20	64	0.001	0.719	0.01	**0.761**
	30	8	0.001	0.753	0.01	0.625
	30	16	0.001	0.732	0.01	0.375
	30	32	0.001	0.682	0.01	0.743
	30	64	0.001	**0.8189**	0.01	0.625
	40	8	0.001	0.78365	0.01	0.375
	40	16	0.001	0.79807	0.01	0.625
	40	32	0.001	0.72916	0.01	0.625
	40	64	0.001	0.751	0.01	0.375
	50	8	0.001	0.782	0.01	0.723
	50	16	0.001	0.7131	0.01	0.687
	50	32	0.001	0.7788	0.01	0.679
	50	64	0.001	0.625	0.01	0.69

Model Description	**Epochs**	**Batch Size**	**Learning Rate**	**Accuracy**
1 conv layer	10	8	0.001	0.748
1 pool layer	10	16	0.001	0.625
2 fc layers	10	32	0.001	0.7131
	10	64	0.001	0.625
	20	8	0.001	0.727
	20	16	0.001	0.7435
	20	32	0.001	0.7532
	20	64	0.001	0.738

	30	8	0.001	**0.79166**
	30	16	0.001	0.7419
	30	32	0.001	0.625
	30	64	0.001	0.7532
	40	8	0.001	0.7756
	40	16	0.001	0.625
	40	32	0.001	0.76121
	40	64	0.001	0.7724
	50	8	0.001	0.7339
	50	16	0.001	0.7644
	50	32	0.001	0.790064
	50	64	0.001	0.7644

Dropout used						
Model Description	**Epochs**	**Batch Size**	**Learning Rate**	**Accuracy**	**Learning Rate**	**Accuracy**
2 conv layers	10	8	0.001	0.72275	0.01	**0.79**
2 pool layers	10	16	0.001	0.7147	0.01	0.7
3 fc layers	10	32	0.001	0.7772	0.01	0.78
dropout=0.5	10	64	0.001	0.78044	0.01	0.74
	20	8	0.001	0.7548	0.01	0.769
	20	16	0.001	0.7724	0.01	0.746
	20	32	0.001	0.7724	0.01	0.782051
	20	64	0.001	0.75801	0.01	0.7198
	30	8	0.001	0.748397	0.01	0.748397
	30	16	0.001	0.75641	0.01	0.7692
	30	32	0.001	**0.783653**	0.01	0.74038
	30	64	0.001	0.775641	0.01	0.75
	40	8	0.001	0.770833	0.01	0.743589
	40	16	0.001	0.7435	0.01	0.7772
	40	32	0.001	0.73397	0.01	0.770833
	40	64	0.001	0.76121	0.01	0.73079
	50	8	0.001	0.7307	0.01	0.7435
	50	16	0.001	0.7692	0.01	0.75
	50	32	0.001	0.77724	0.01	0.753
	50	64	0.001	0.769	0.01	0.7451

No Dropout used

Model Description	Epochs	Batch Size	Learning Rate	Accuracy	Learning Rate	Accuracy
2 conv layers	10	8	0.001	0.66	0.01	**0.80447**
2 pool layers	10	16	0.001	0.75	0.01	0.7291
3 fc layers	10	32	0.001	0.828	0.01	0.7307
No Dropout used	10	64	0.001	**0.8766**	0.01	0.7561
	20	8	0.001	0.77	0.01	0.7516
	20	16	0.001	0.8157	0.01	0.7628
	20	32	0.001	0.79647	0.01	0.74038
	20	64	0.001	0.84935	0.01	0.7099
	30	8	0.001	0.716	0.01	0.75641
	30	16	0.001	0.79006	0.01	0.7147
	30	32	0.001	0.7323	0.01	0.7644
	30	64	0.001	0.78044	0.01	0.7147
	40	8	0.001	0.8301	0.01	0.75801
	40	16	0.001	0.78	0.01	0.7596
	40	32	0.001	0.7099	0.01	0.746
	40	64	0.001	0.6987	0.01	0.754