Classification using GCN

Submitted By:

Divya Saxena (1001773376)

Table of Contents

1. Introduction

Neuro-imaging is a branch of medical imaging that focuses on the brain. In addition to diagnosing disease and assessing brain health, neuro-imaging also studies:

- How the brain works
- How various activities impact the brain

Currently, most graph neural network models have a somewhat universal architecture in common. I will refer to these models as *Graph Convolutional Networks* (GCNs); convolutional, because filter parameters are typically shared over all locations in the graph

For these models, the goal is then to learn a function of signals/features on a graph G=(V,E)

which takes as input:

- A feature description *xi*

for every node *i*; summarized in a $N \times D$ feature matrix X (N: number of nodes, D

- : number of input features)
- A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix A
- (or some function thereof)

and produces a node-level output Z

(an $N \times F$ feature matrix, where F is the number of output features per node). Graph-level outputs can be modeled by introducing some form of pooling operation

2D Images have been given as Input to GCN for binary classification of images as Healthy and Patient.

Independent test dataset has been used for evaluating the model performance.

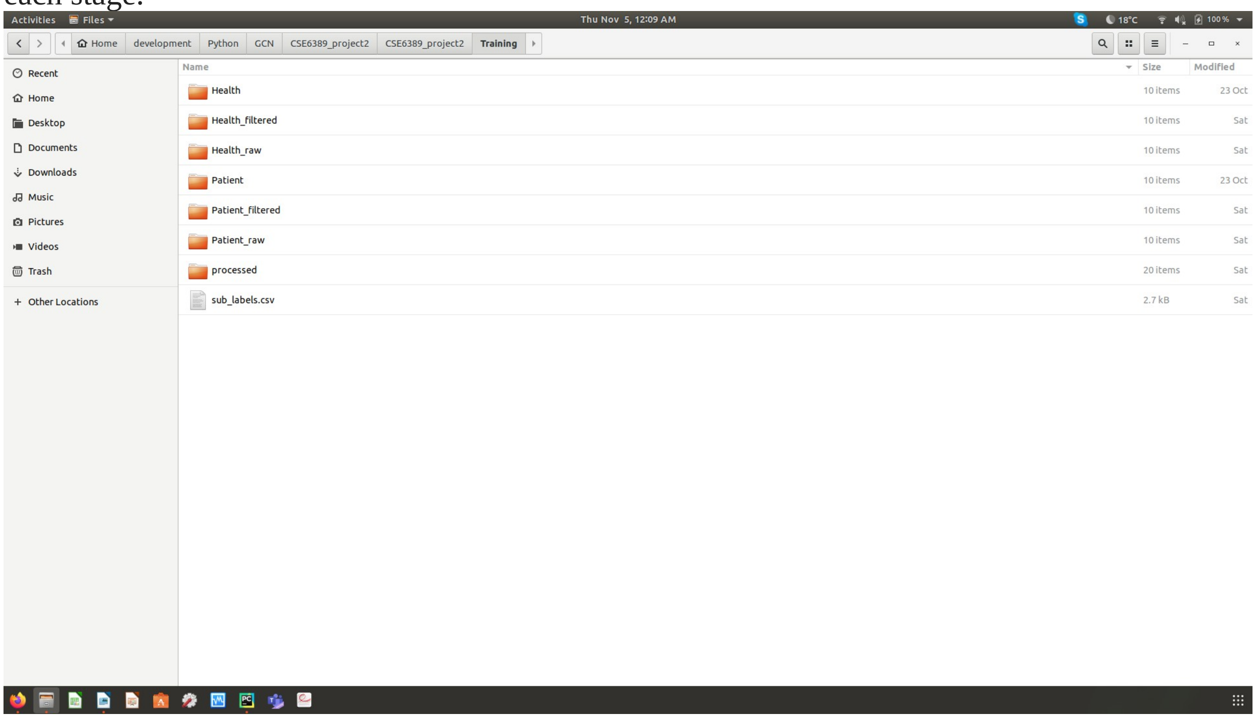
2. Code Description

Python Files
/GCN/preprocess_data.py
/GCN/preprocessing_testing.py
/GCN/show_image.py
/GCN/dataset_split.py
/GCN/graphplt.py
/GCN/load_data_sets.py
/GCN/train.py

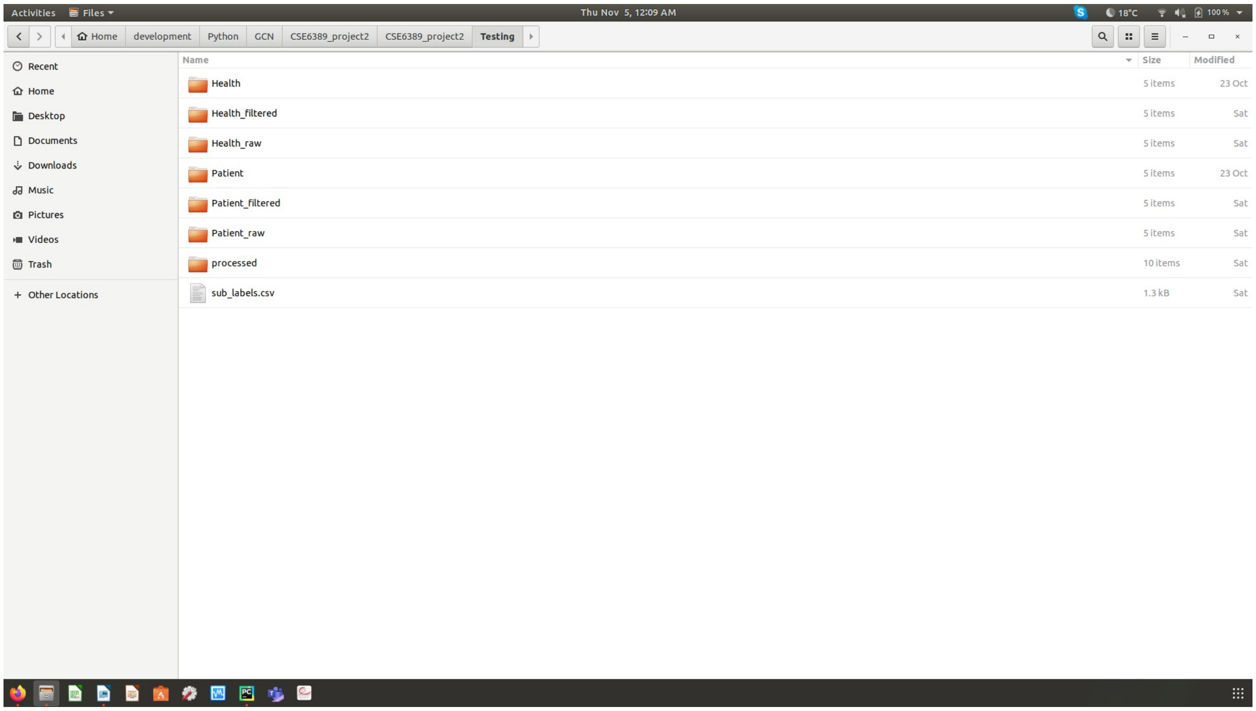
Packages Used for Coding

numpy
scipy
os
matplotlib.pyplot
pandas
sklearn
skimage
pathlib
tensorflow
spektral

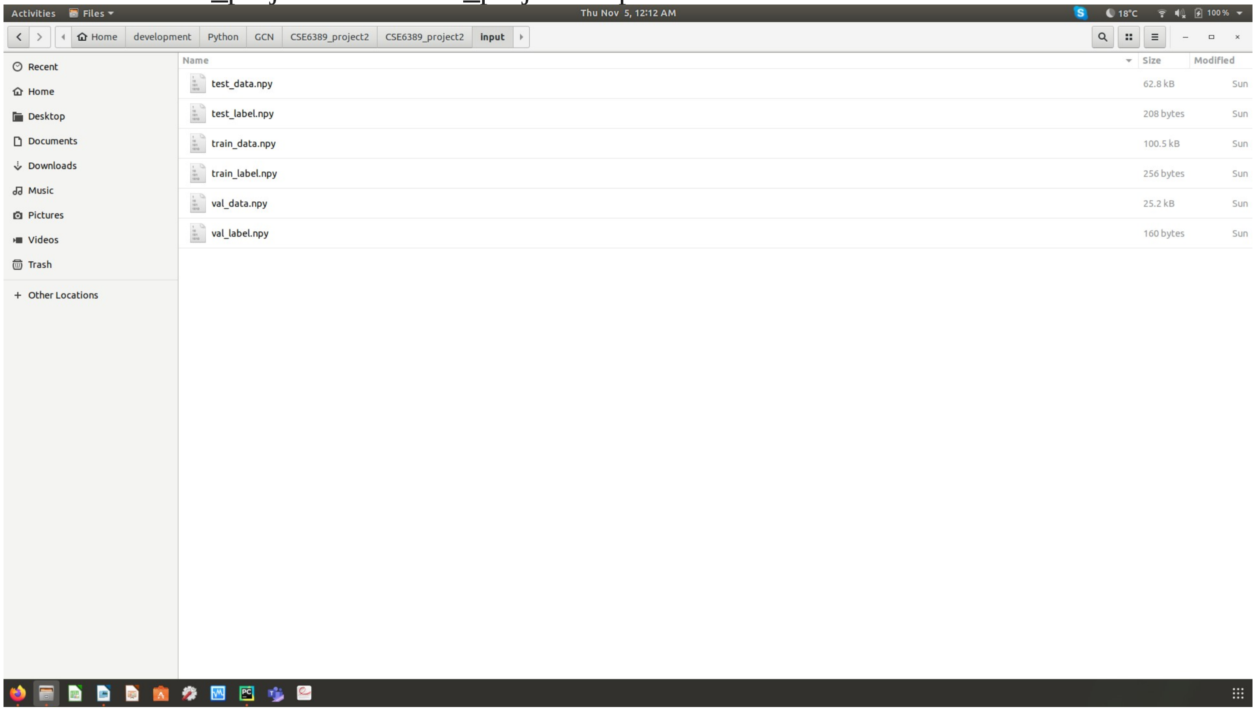
Folder Structure for dataset:
/GCN/CSE6389_project2/CSE6389_project2/Training
I have created some folders after each stage during pre-processing as attached below, images during each stage will get saved in new folder, so that we can easily identify the changes during each stage.



/GCN/CSE6389_project2/CSE6389_project2/Testing
We have created some folders after each stage during pre-processing as attached below, images during each stage will get saved in new folder, so that we can easily identify the changes during each stage.



After pre-processing, some files have been generated for the GCN model as attached below. These files will be used by GCN model for training/testing/validation /GCN/CSE6389_project2/CSE6389_project2/input



3. How to run code

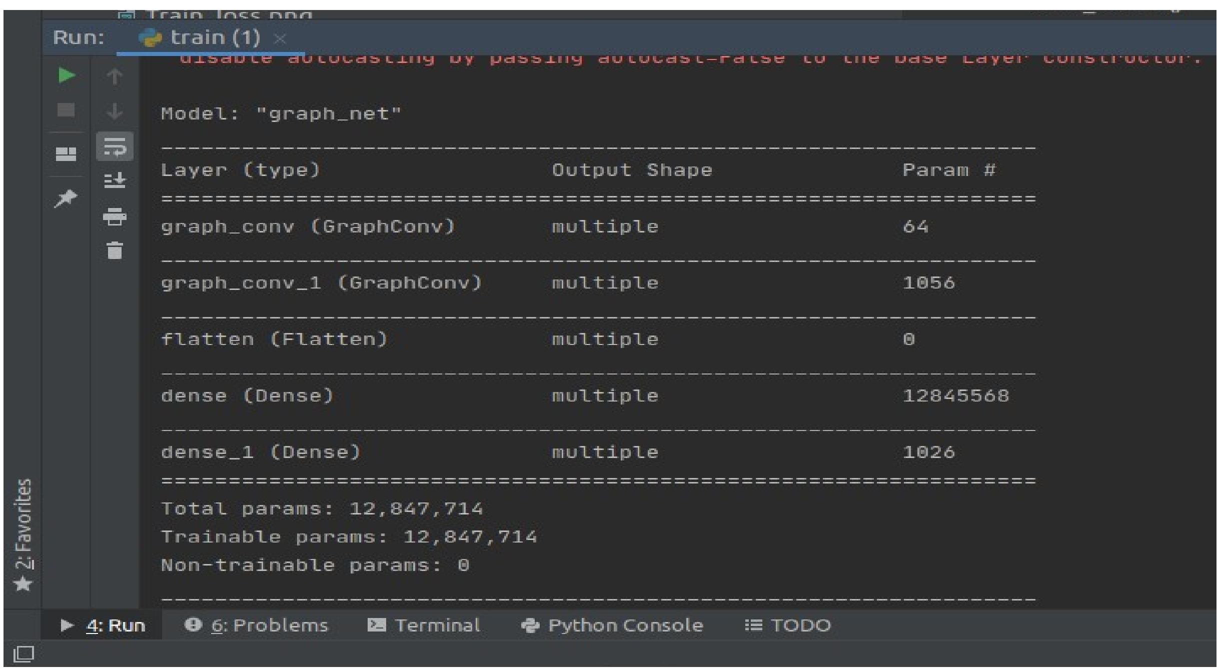
To Show Image
python /GCN/show_image.py

For Pre-processing Data
python /GCN/preprocessing_testing.py

To Train Model
python /GCN/train.py

4. Graph Convolutional Neural Network Model Description

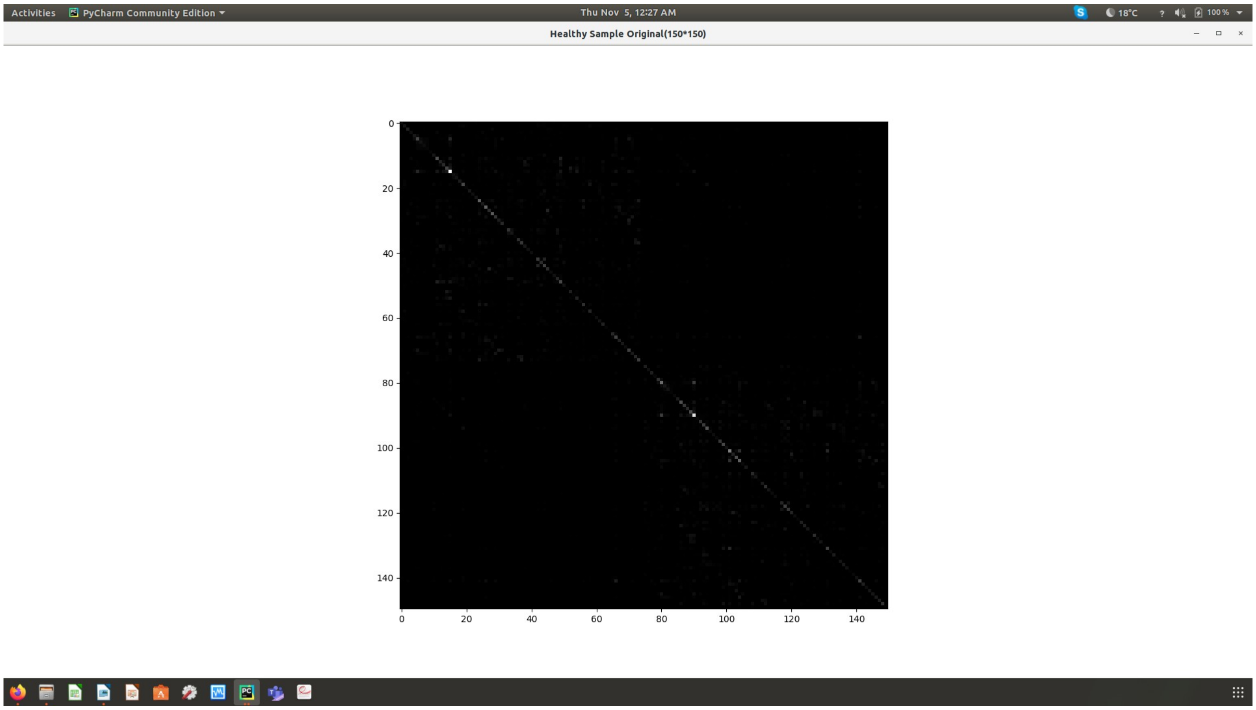
"GraphConv" model has been used to create GCN. 5 layers have used to create the model. Attaching screenshot for the same. Images given are very big in size approximately 150*150 as we can see here . Image has been resized to 28*28 due to limited GPU and CPU.
Adam optimizer has been used as it is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks.
SparseCategoricalCrossentropy has been used as it computes the crossentropy loss between the labels and predictions
SparseCategoricalAccuracy has been used as it Calculates how often predictions matches integer labels



5. Image Visualization Healthy vs Patient

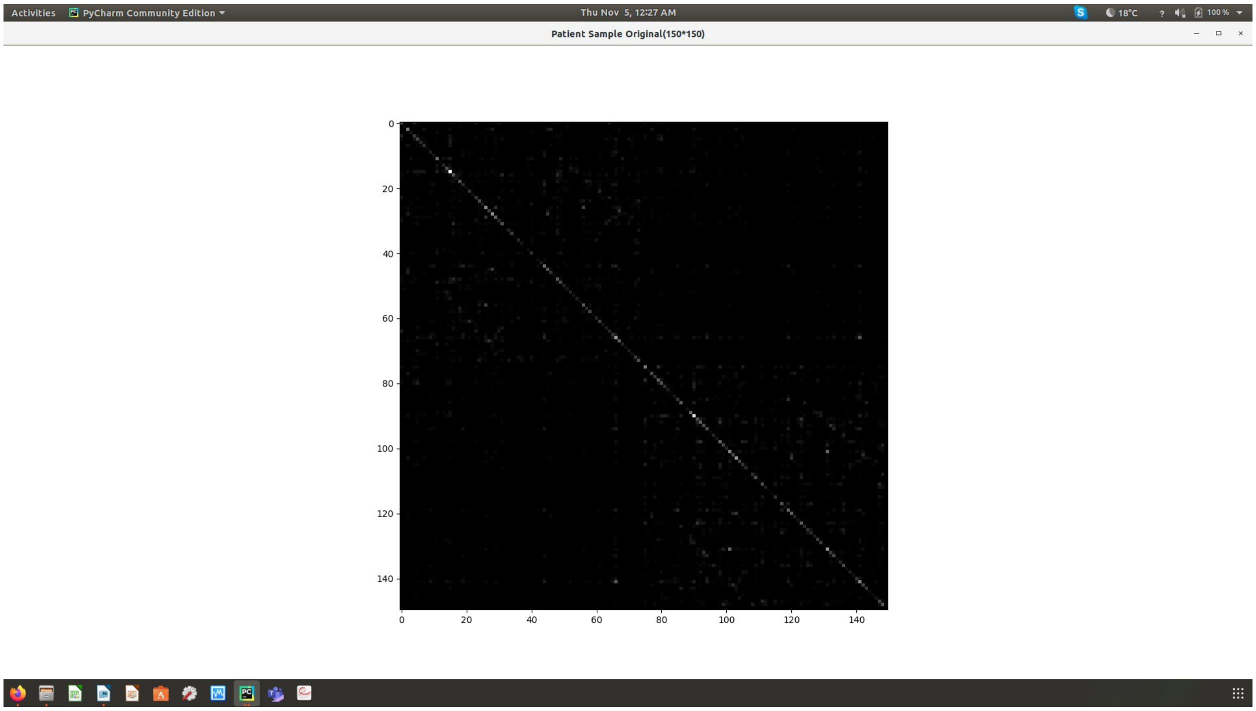
/GCN/show_image.py
In this python file, we have tried to show the 2D-image of both Healthy and Patient Dataset. We have randomly selected one image of both dataset given. The dimension used are the original ones i.e. 150*150

1. Health



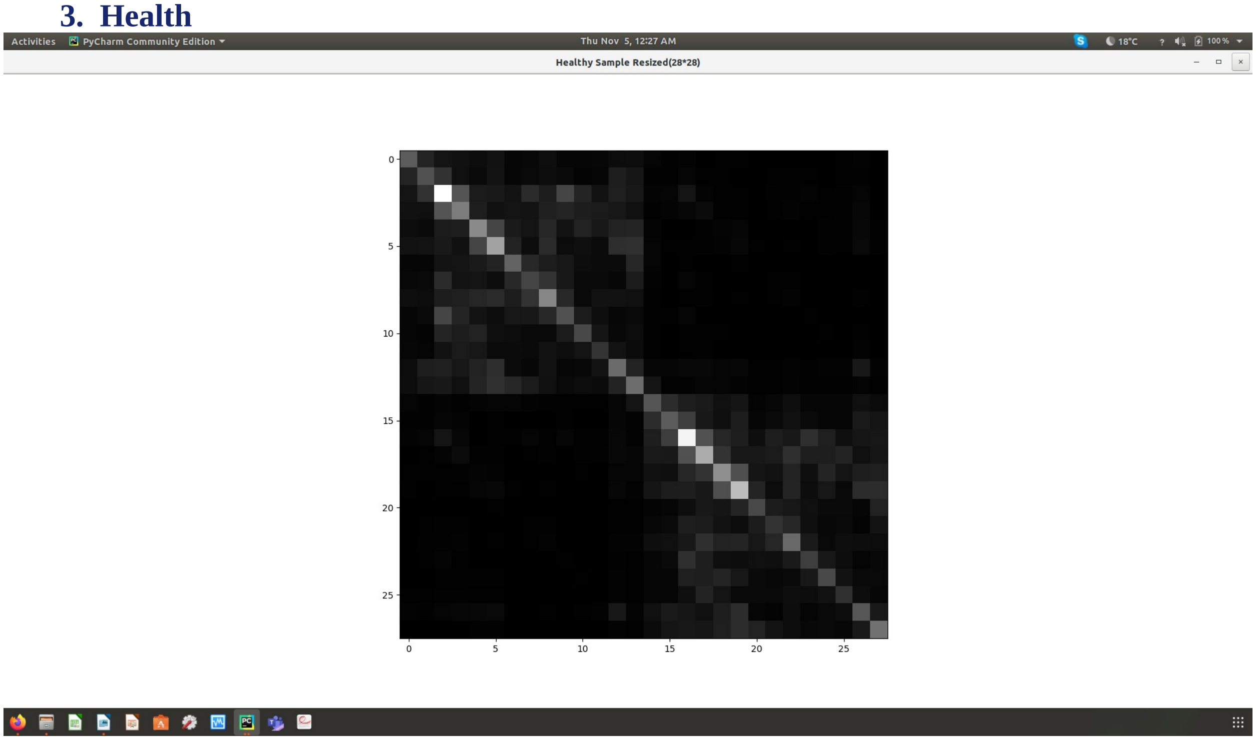
2. Patient

We can see some differences in both Health and patient images.



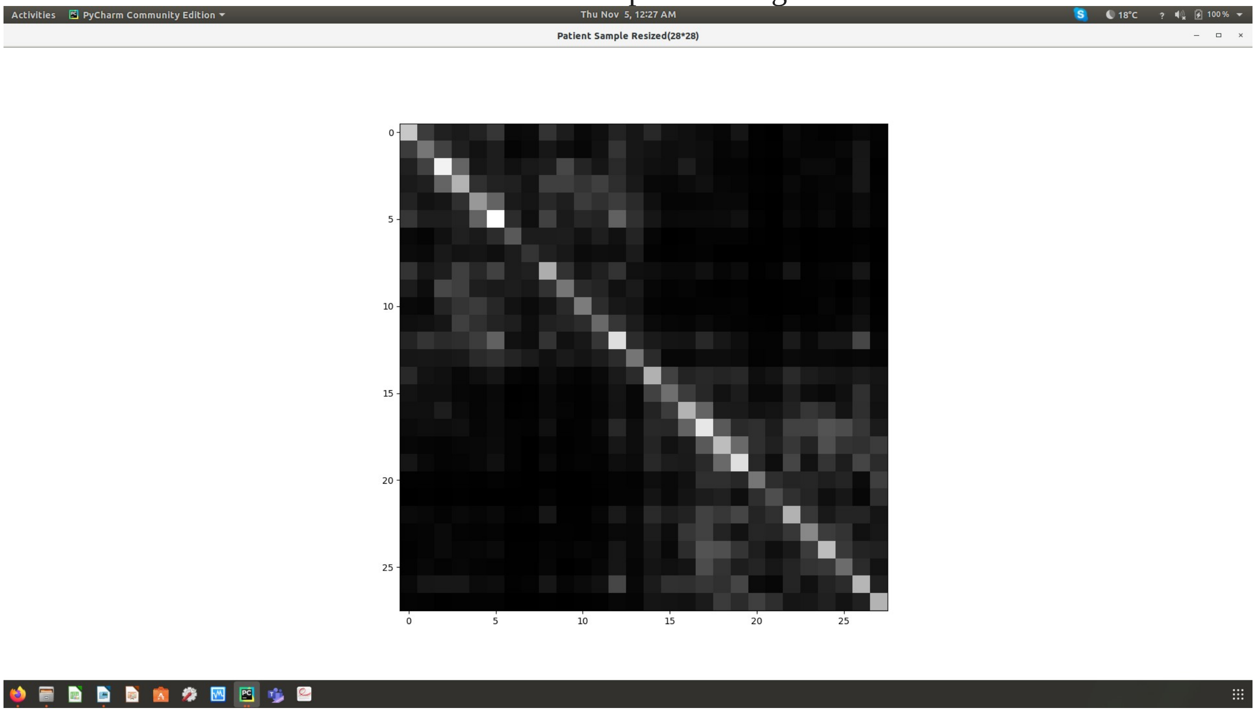
We have randomly selected one image of both dataset given. The dimension used are the resized ones i.e. 28*28. As mentioned earlier, due to limited GPU and CPU, we have resized the orginal image to 28*28 for training our model.

3. Health



4. Patient

We can see some differences in both Health and patient images.



6. Data Preprocessing

Same Pre-processing steps have been used as in previous project as stated below:
/GCN/preprocess_data.py
/GCN/preprocessing_testing.py

Below steps have been followed for both Training and Testing dataset but testing data kept separately to avoid over-fitting of the model.

Code has been implemented as below:
1.Put .txt files in patient_raw, health_raw
Matrices will be saved in folder separately for Training and testing i.e.
(/GCN/CSE6389_project2/CSE6389_project2/Training/health_raw,/GCN/CSE6389_project2/CSE6389_project2/Training/patient_raw)
and
(/GCN/CSE6389_project2/CSE6389_project2/Testing/health_raw,/GCN/CSE6389_project2/CSE6389_project2/Testing/patient_raw)

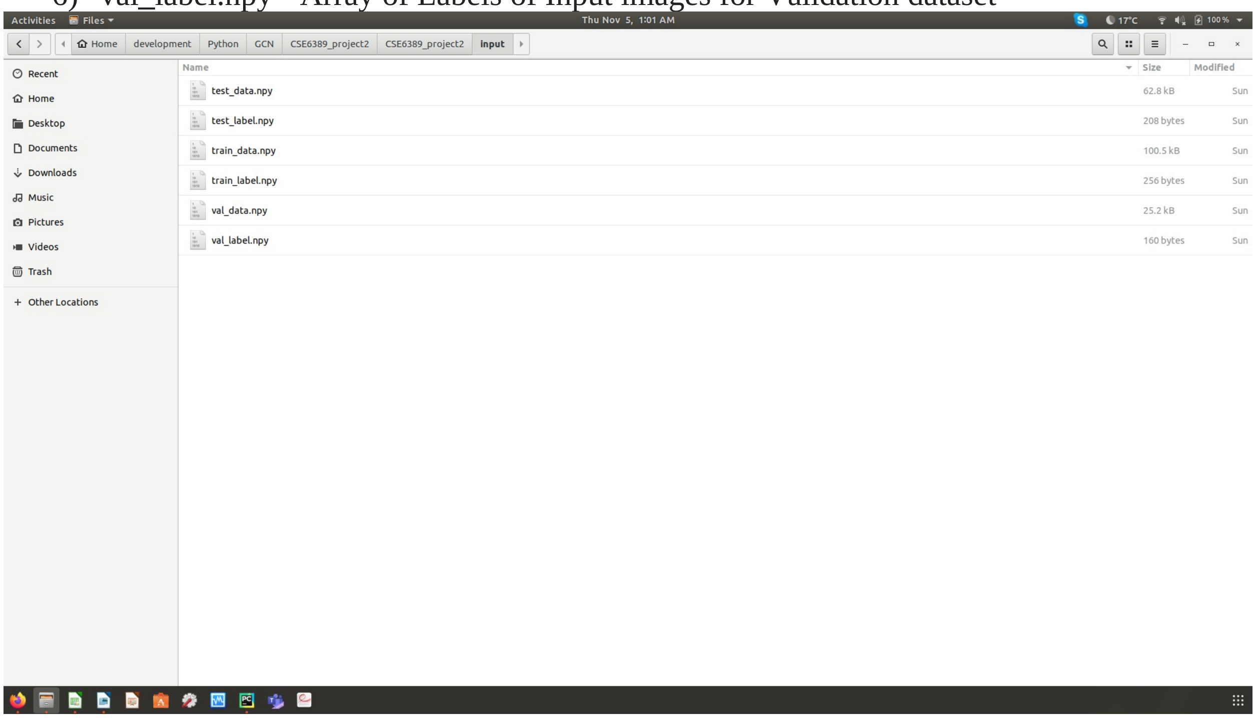
2.Filtered .txt to .npy in folder(patient_filtered, health_ filtered)
(/GCN/CSE6389_project2/CSE6389_project2/Training/health_filtered,/GCN/CSE6389_project2/CSE6389_project2/Training/patient_filtered)
and

(/GCN/CSE6389_project2/CSE6389_project2/Testing/health_filtered,/GCN/CSE6389_project2/CSE6389_project2/Testing/patient_filtered)

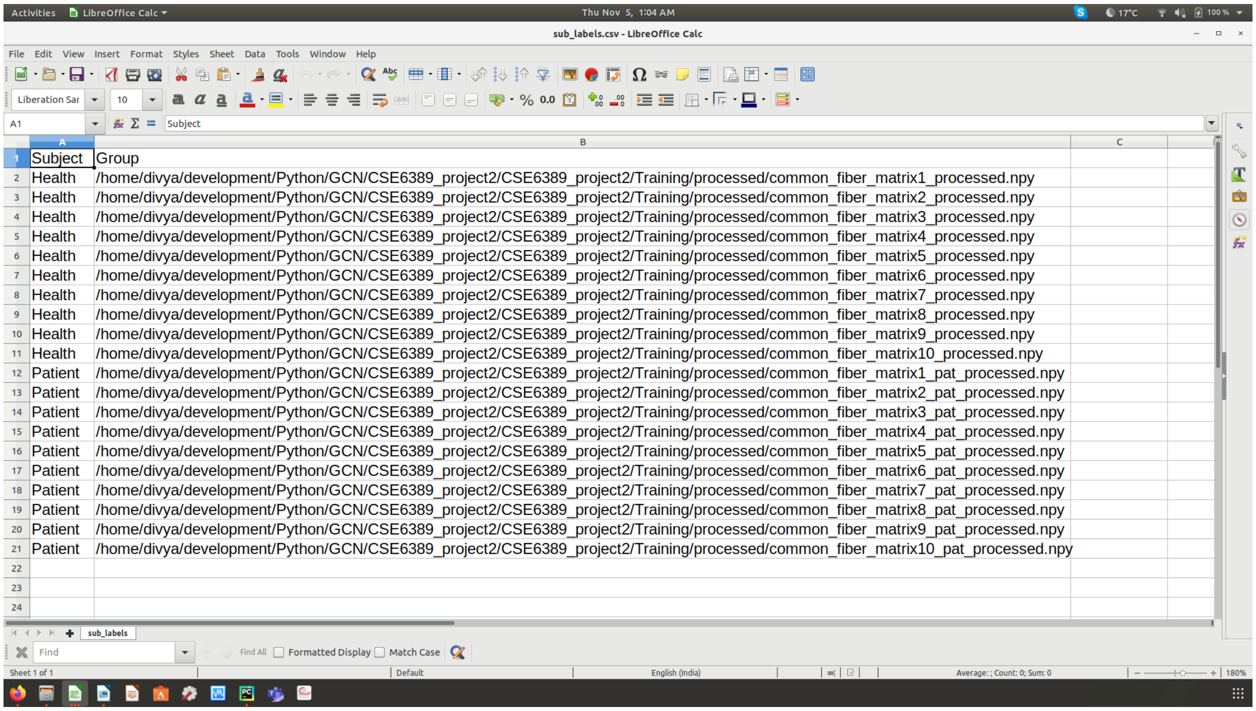
3.Processed all healthy/patient to processed folder
Pre-processing of original images. Pre-processing include: resizing
Image has been resized to 28*28 due to limited GPU and CPU. Single folder for both Health and Patient. But the images can be differentiated by their names as for patient the images have been renamed like (%_pat_%)
(/GCN/CSE6389_project2/CSE6389_project2/Training/processed)
and
(/GCN/CSE6389_project2/CSE6389_project2/Testing/processed)

4.Final step is after pre-processing, data has been split into 2 parts :
Training and Validation. All the data has been converted into single files .npy. And Testing dataset as well. All files have been created in Input folder as data and their corresponding labels

1) test_data.npy - Array of Input images for Testing dataset
2) test_label.npy - Array of Labels of Input images for Testing dataset
3) train_data.npy - Array of Input images for Training dataset
4) train_label.npy - Array of Labels of Input images for Training dataset
5) val_data.npy - Array of Input images for Vaidation dataset
6) val_label.npy - Array of Labels of Input images for Validation dataset

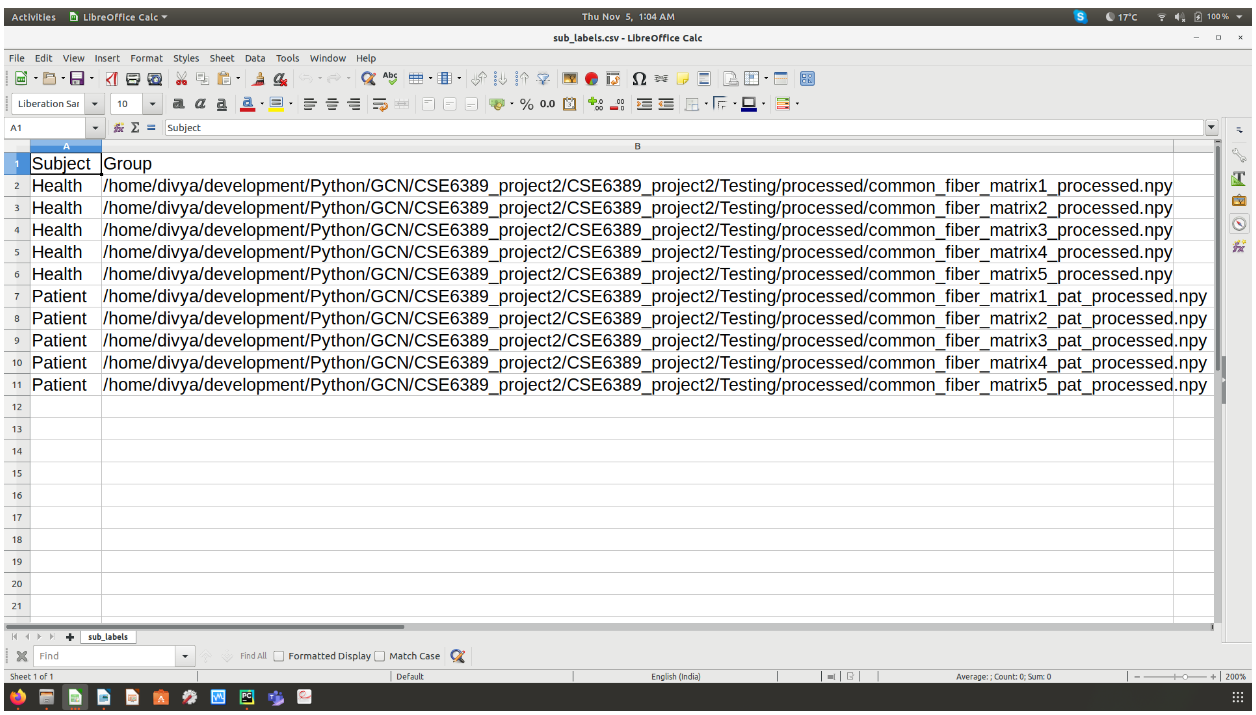


5.Some manual steps required for Training the model
Created labels for Training/Validation in sub_label.csv as Subject and Group as the heading at location /GCN/CSE6389_project2/CSE6389_project2/Training



Subject	Group
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix1_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix2_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix3_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix4_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix5_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix6_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix7_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix8_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix9_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix10_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix1_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix2_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix3_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix4_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix5_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix6_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix7_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix8_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix9_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/processed/common_fiber_matrix10_pat_processed.npy

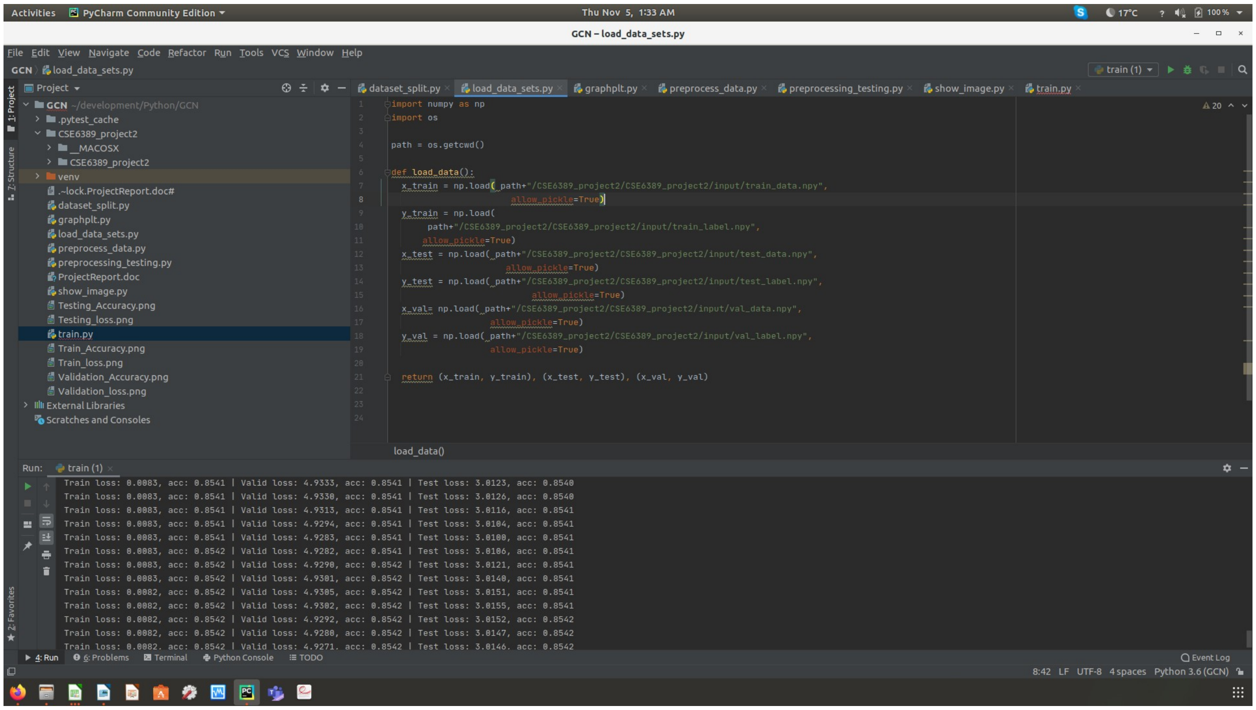
6.Created labels for Testing in sub_label.csv as Subject and Group as the heading at location /GCN/CSE6389_project2/CSE6389_project2/Testing



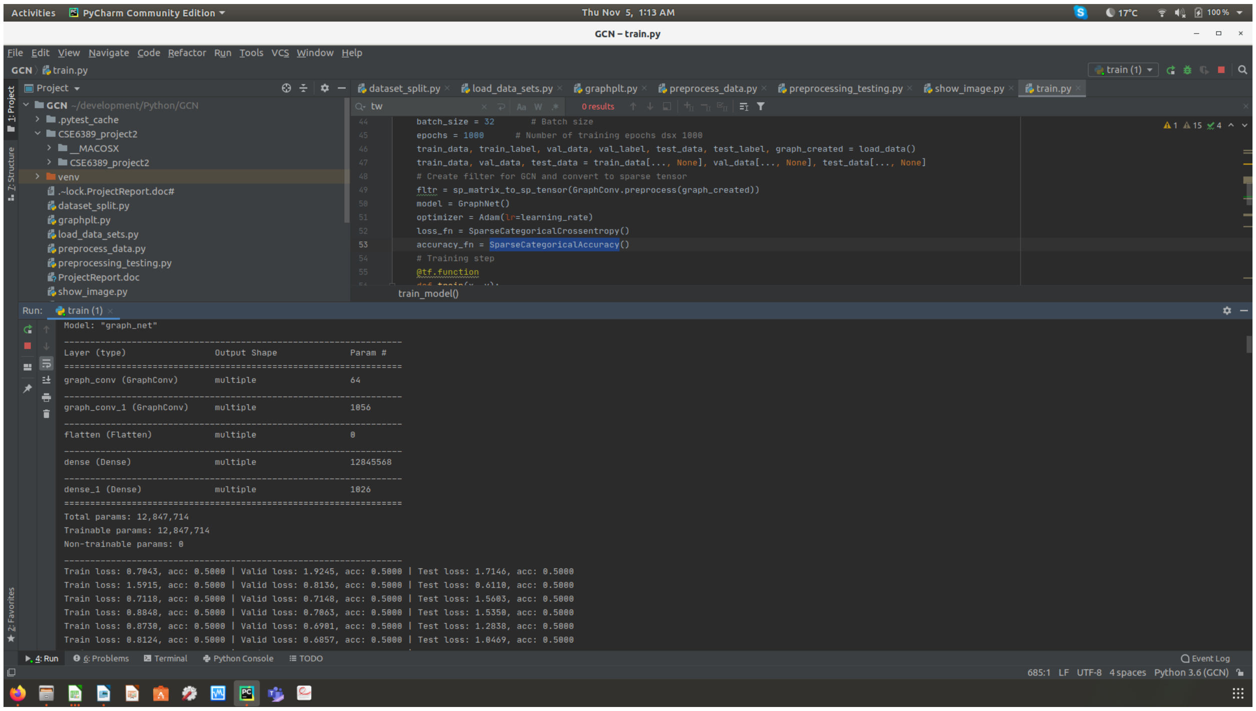
Subject	Group
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix1_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix2_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix3_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix4_processed.npy
Health	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix5_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix1_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix2_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix3_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix4_pat_processed.npy
Patient	/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Testing/processed/common_fiber_matrix5_pat_processed.npy

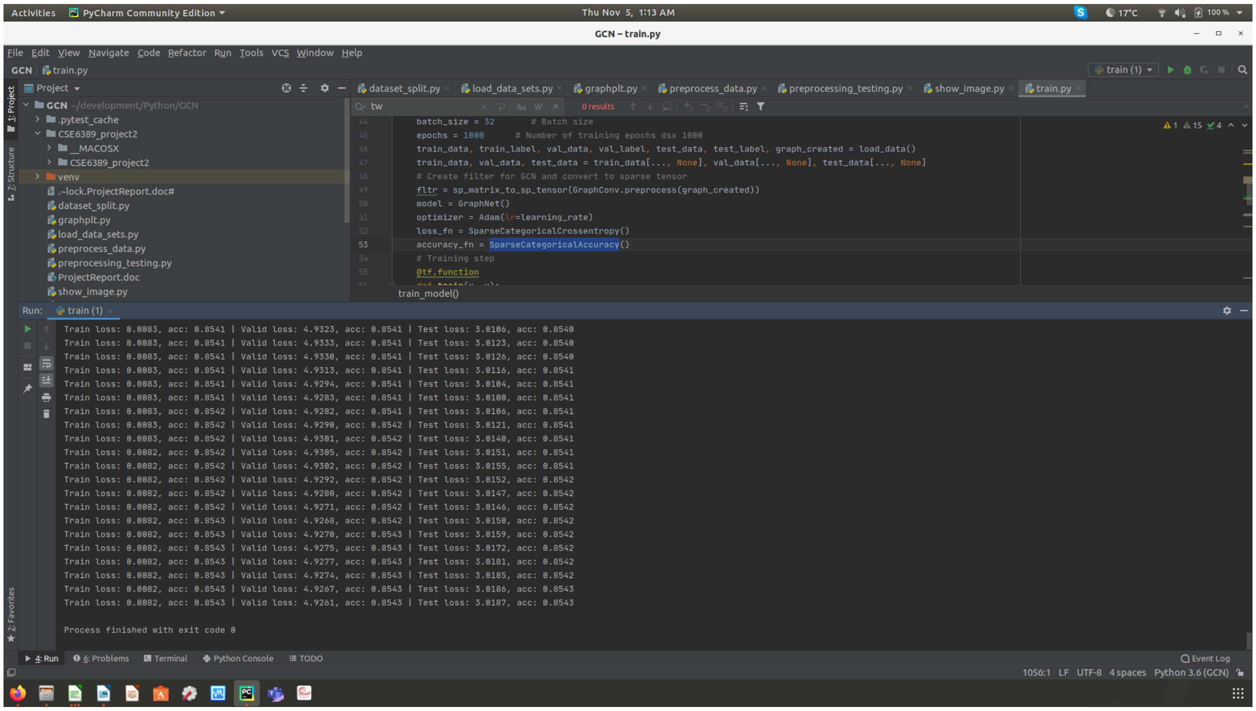
7. Model Training

/GCN/train.py
Model has been trained for 1000 number of epochs as attached below using Training and validation dataset. Testing dataset has been kept aside for evaluating the model. It has been used as Independent dataset to avoid over-fitting.
Firstly, pre-processed data has been loaded for the model as screenshot below:



For training, we have used tf.GradientTape. TensorFlow provides the API for automatic differentiation; that is, computing the gradient of a computation with respect to some inputs, usually tf.Variables . TensorFlow "records" relevant operations executed inside the context of tf.GradientTape onto a "tape". TensorFlow then uses that tape to compute the gradients of a "recorded" computation using reverse mode differentiation.

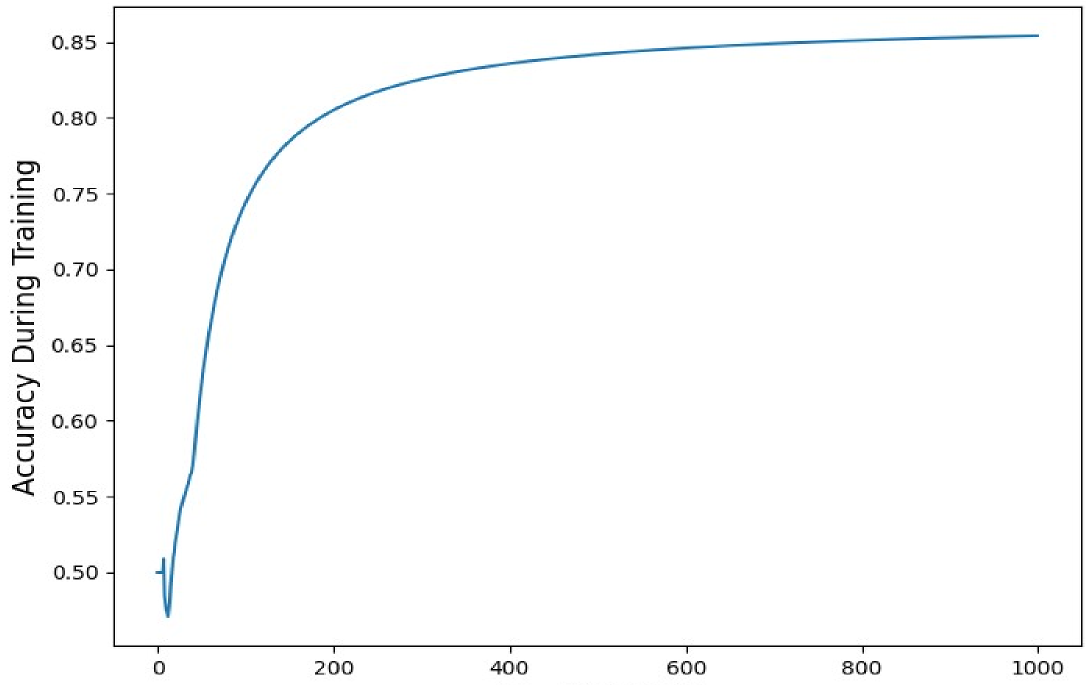


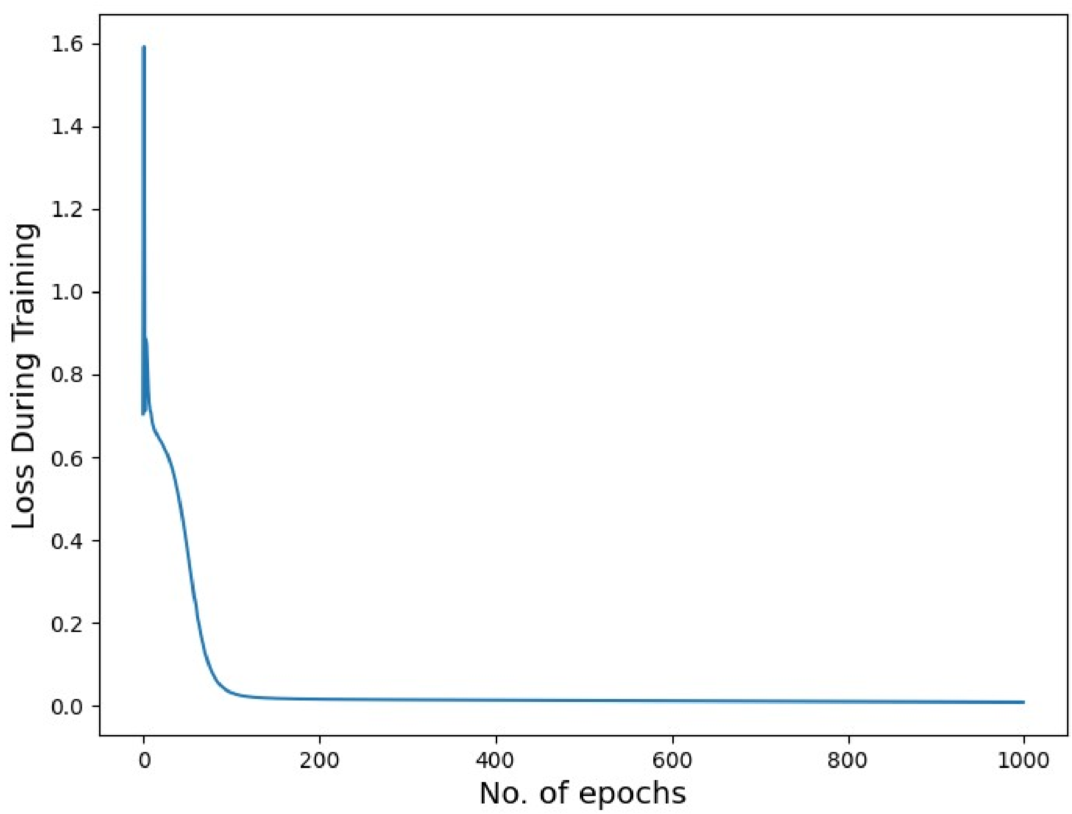


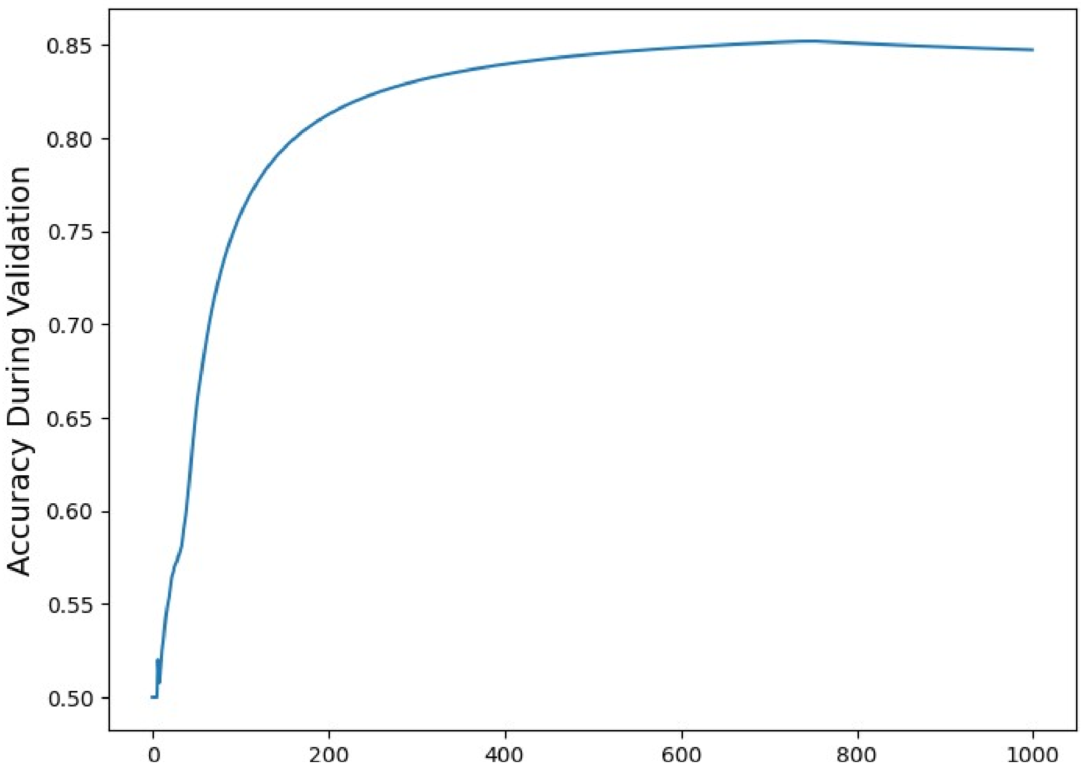
After training the model, graphs will get saved in GCN folder.
Testing dataset has been used independently for evaluating performance

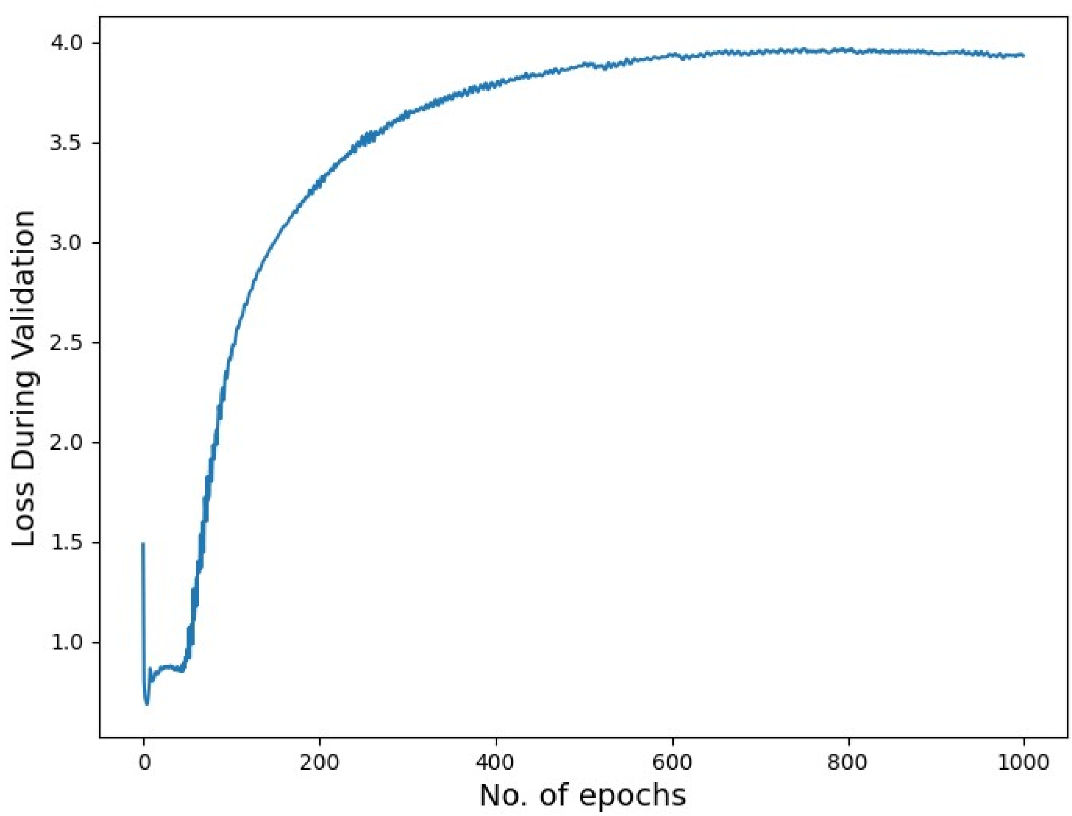
8. Graphs/Accuracy/Loss

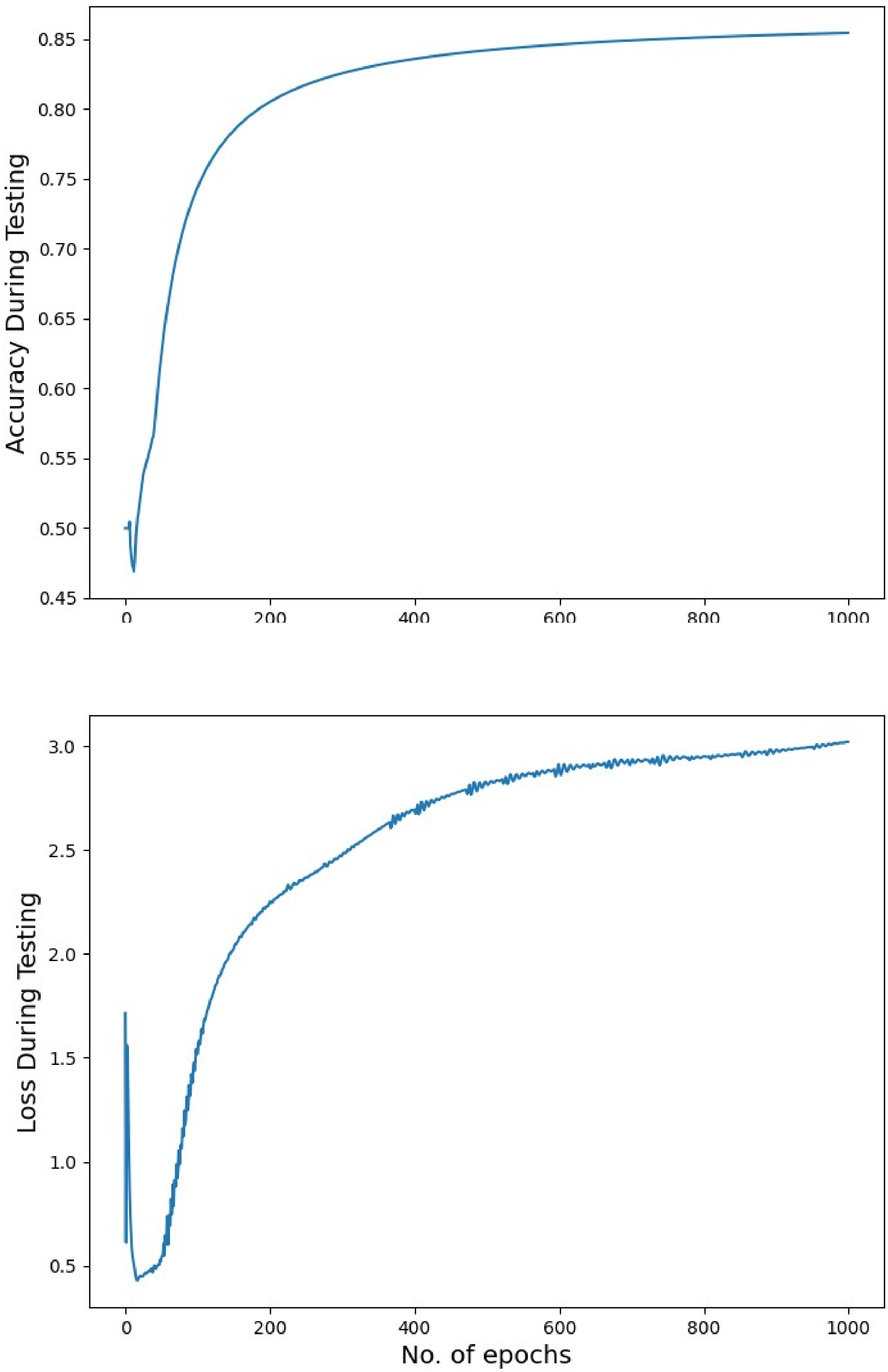
As we can see that due to limited dataset, the model has been over-fitted.
Attaching below graph for Training,validation and Testing - accuracy and loss.






Accuracy During Validation
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0
200
400
600
800
1000


Loss During Validation
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
200
400
600
800
1000
No. of epochs


Accuracy During Testing
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
200
400
600
800
1000
Loss During Testing
3.0
2.5
2.0
1.5
1.0
0.5
0
200
400
600
800
1000
No. of epochs

9. Driver script

If we want to Pre-process data again, we have to empty our folders again using Terminal

```
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Training/health_filtered;
rm *.npy;
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Training/health_raw;
rm *.nii;cd
/home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/Training/
patient_filtered;
rm *.npy;
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Training/patient_raw;
rm *.nii;
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Training/processed;
rm *.npy;
#####Testing
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Testing/health_filtered;
rm *.npy;
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Testing/health_raw;
rm *.nii;
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Testing/patient_filtered;
rm *.npy;
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Testing/patient_raw;
rm *.nii;
cd /home/divya/development/Python/GCN/CSE6389_project2/CSE6389_project2/
Testing/processed;
rm *.npy;
```

10. References

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Layer
https://www.tensorflow.org/guide/keras/transfer_learning
https://www.tensorflow.org/guide/autodiff