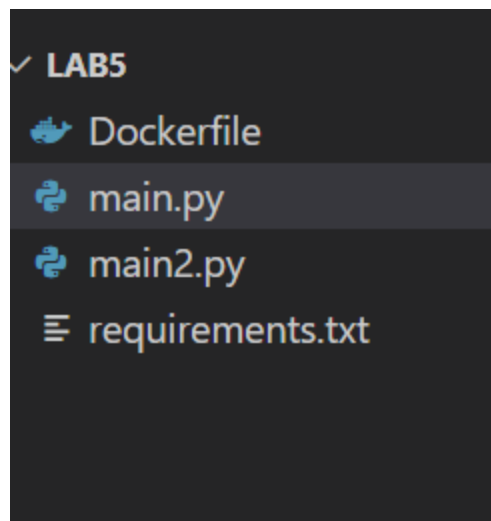


# B20BB030 LAB5

## CONTENT

1. Workspace structure
2. Creating the main python file (training + testing)
3. Early stopping
4. Docker workflow that was implemented

## Workspace structure



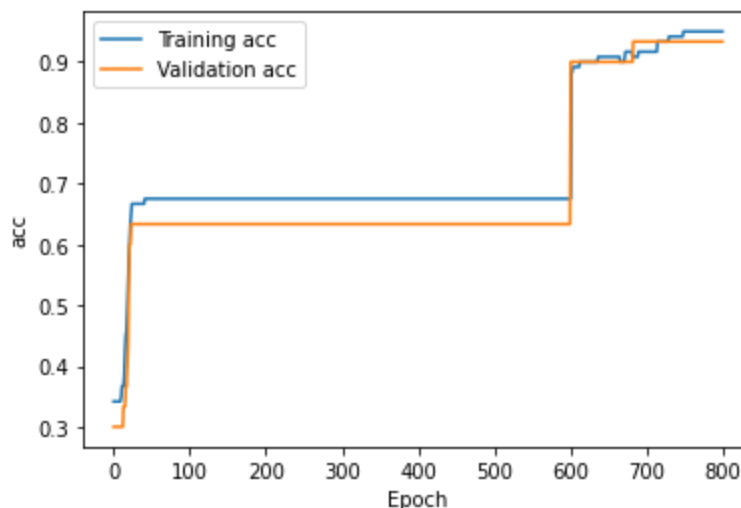
## Creating the main files for training and testing

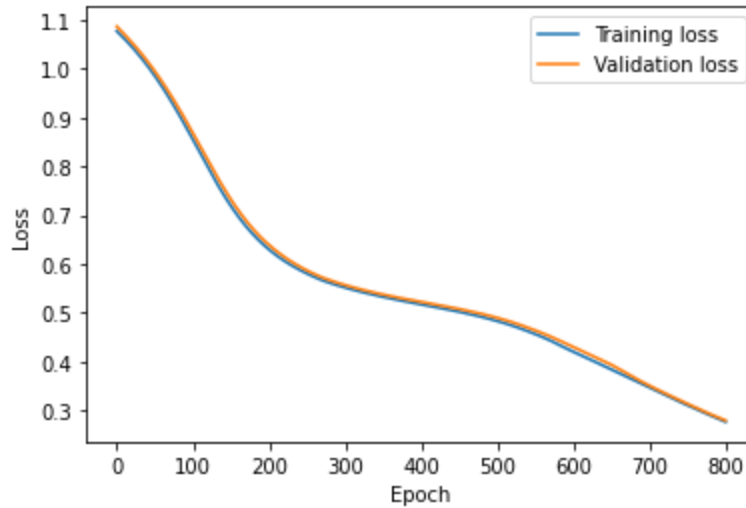
For this, I've followed these steps :

1. Imported the necessary libraries such as torch, matplotlib, and sklearn.

2. Loaded the IRIS dataset using the sklearn library.
3. Preprocessed the data by splitting it into training and testing data using `train_test_split`.
4. Normalized the data by using `StandardScaler`.
5. Defined the architecture of the Neural Network. We will use 2 hidden layers, one with 4 neurons and one with 5 neurons, an input layer with 4 neurons, and an output layer with 3 neurons.
6. Chosen the activation function and loss function for the model. We will use ReLU as the activation function for the hidden layers and Softmax as the activation function for the output layer. I've used the cross-entropy loss as the loss function.
7. Trained the model using the training data. used early stopping to avoid overfitting. stopped the training if the validation loss stops improving after 10 epochs.
8. Evaluated the model on the testing data and report the test loss, test accuracy, train loss, and train accuracy. also plotted the curves for the loss and accuracy.

## Results and plots



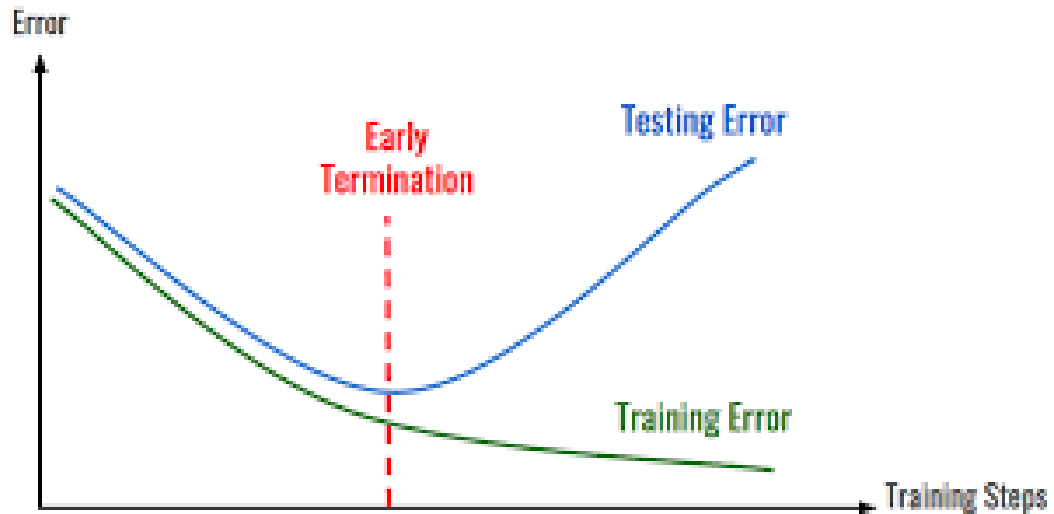


Test loss: 0.2798, Test accuracy: 0.933

## Early Stopping


Early stopping is a technique used to avoid overfitting in machine learning models. It involves monitoring the validation loss during training and stopping the training process when the validation loss stops improving after a certain number of epochs. This helps prevent the model from becoming too specialized to the training data and improves its ability to generalize to new data.

In this project, early stopping was implemented by monitoring the validation loss during training and stopping the training process if the validation loss did not improve after 10 epochs. This was done to prevent overfitting and improve the model's ability to generalize to new data.

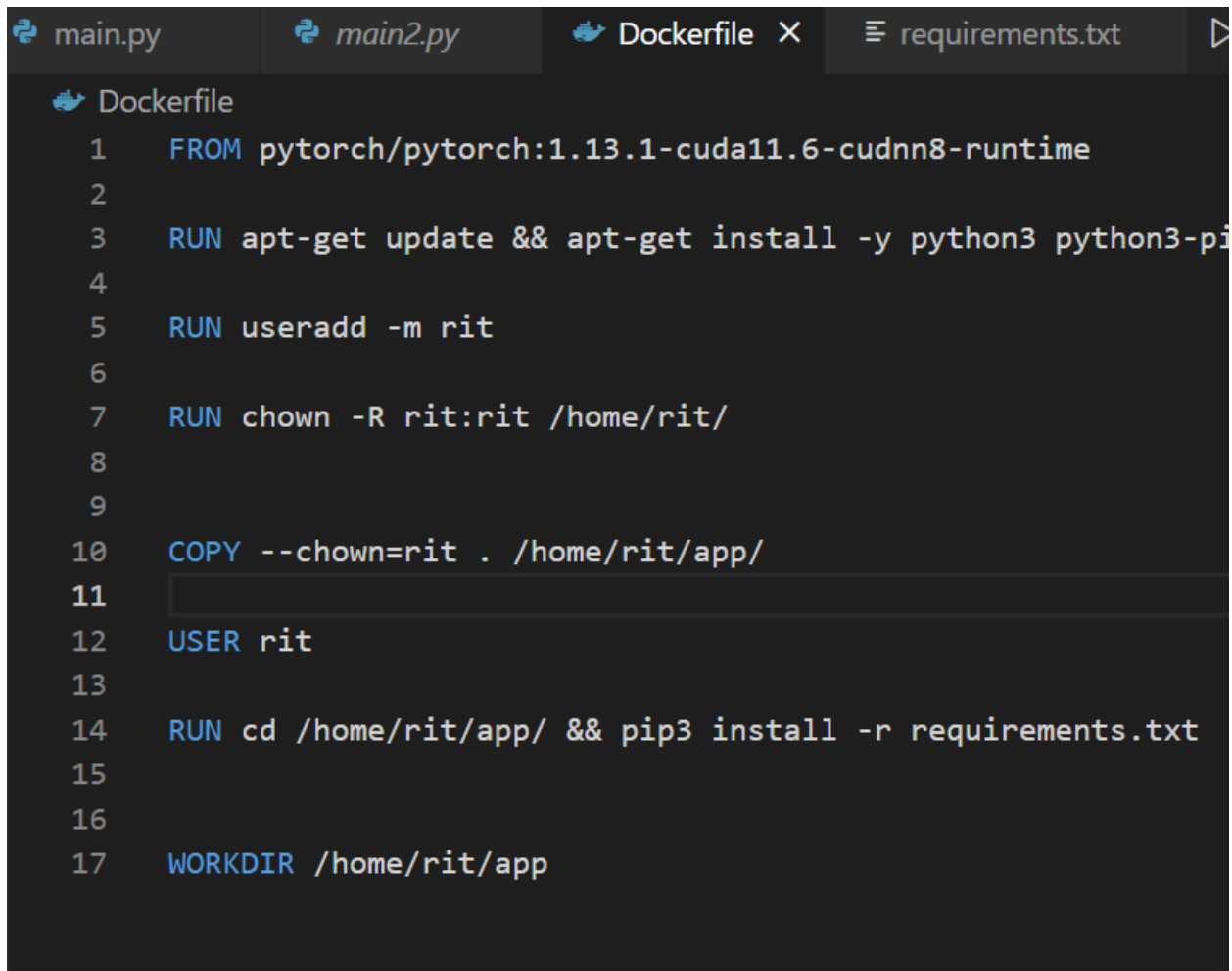


## Docker Workflow

- Found the required image on Docker Hub i.e

 `pytorch/pytorch:1.13.1-cuda11.6-cudnn8-runtime`

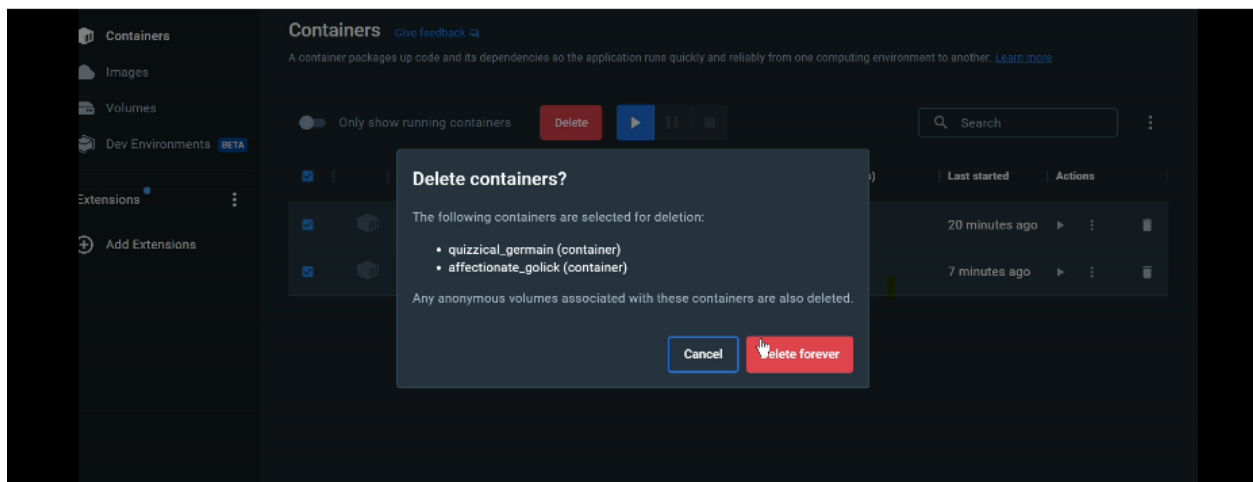
- Pulled the required image and Run the image to the host and Push your codes in the container. for this step, I had created a Dockerfile and ran the following commands on cmd



```
main.py  main2.py  Dockerfile  requirements.txt
Dockerfile
1  FROM pytorch/pytorch:1.13.1-cuda11.6-cudnn8-runtime
2
3  RUN apt-get update && apt-get install -y python3 python3-pip
4
5  RUN useradd -m rit
6
7  RUN chown -R rit:rit /home/rit/
8
9
10 COPY --chown=rit . /home/rit/app/
11
12 USER rit
13
14 RUN cd /home/rit/app/ && pip3 install -r requirements.txt
15
16
17 WORKDIR /home/rit/app
```

```
docker build -f Dockerfile -t ma .
docker run -ti ma python3 main2.py
```

- Stopped the container and Remove the container



- Deleted the image.

