Assignment 2

Dipankar Sarkar (MT20307)
Shivam Dalmia   (MT20304)

Question 1

Code Explanation:

from torchvision import transforms: This is used to transform the image data into tensors. The dimensions for each image is 128*128.

validation_data_size = int((10 * len(image_dataset))/100) : The validation data percentage of all the image dataset, this is how the test data and the train data percent is divided also.

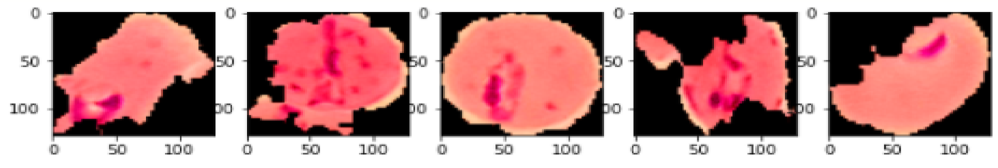
class DeviceDataLoader() && def to_device(data, device): This two is used to load the data in the device (CPU or GPU) which we are using.

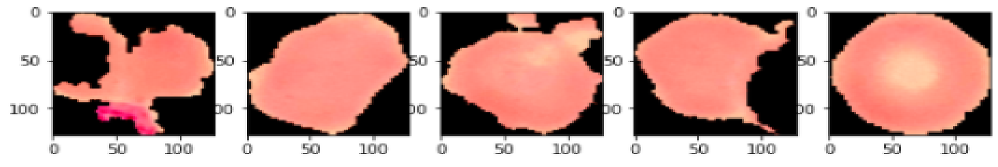
train_data, validation_data ,test_data = random_split(image_dataset, [train_data_size, validation_data_size,test_data_size]) : This is used to split the dataset into test,train and validation data according to the ratio given to the test train and validation data.

def imshow(img,k,j): This function is used to plot the image img.

for img in range(5):
   imshow(image_dataset[int(random.random()*100)][0],k,img) : This loop runs for five times and calls the imshow() function. It chooses 5 numbers randomly and plots the uninfected image.

for img in range(5):
         imshow(image_dataset[int(random.random()*100+len(image_dataset)/2+1)][0],k,img) : This is done for parasite images.





def train_validation_loss(history): This function prints all the training loss and validation loss of all the different models.

class CNN(nn.Module): This class is used to calculate the CNN structure.

def random_weights(m) && def zeros_weights(m)&& def he_weights(m): This function initializes the weights of the kernel as random values , zero values and values using the technique.

Structure of the CNN Model:

Block A:
nn.Conv2d(3, 16, kernel_size=9,stride=1,padding=3),
nn.MaxPool2d(kernel_size=2,stride=2),
Block B:
nn.Conv2d(16, 32, kernel_size=6,stride=1,padding=3),
nn.MaxPool2d(kernel_size=2,stride=2),
Block C:
nn.Conv2d(32, 64, kernel_size=3, padding=1),

Block A contains the kernel size to be 9*9 and padding used are 3(meaning 6 rows and 6 columns) and Max pooling with kernel size 2*2 and stride as 2 is used.

Block B contains the kernel size to be 6*6 and padding used are 3(meaning 6 rows and 6 columns) and Max pooling with kernel size 2*2 and stride as 2 is used.

Block C contains the kernel size to be 3*3 and padding used are 1(meaning 2 rows and 2 columns) def l1_regularisation(self,x) && def l2_regularisation(self,x): These functions use L1 and L2 regularization loss to update the weights.

def evaluate(model,val_loader): This function uses the existing model(weights) to calculate the accuracy.This function is used to calculate the accuracy of the validation dataset with the model calculated. It takes the model and the data as an input.

def training(epochs, lr, model, train_loader, val_loader,opt_func= torch.optim.SGD, regularization='l1'): This function takes the model prepared,the data,learning rates,optimizing function and regularization is default 'l1'. This function calculates the loss from the model and data. It contains two functions that calculate the loss using L1 and L2 regularization too.

model_random = to_device(CNN('random_weights'), device): This line first executes the CNN model with the random_weights and then loads the data in the model_random.So till now all the filters with random weights have been assigned.

train_device_loader=DeviceDataLoader(train_data_loader,device): Loads the train data
val_device_loader=DeviceDataLoader(val_data_loader,device):Loads the validation data

history_random=training(num_epochs,learning_rate,model_random,train_device_loader,val_device_loader,optimizer_function,"") : training function is called with number of epochs with the CNN model, learning rate,train data,validation data and the optimizer function in this case Adam optimizer.

model_he = to_device(CNN('he_weights'), device)
train_device_loader=DeviceDataLoader(train_data_loader,device)
val_device_loader=DeviceDataLoader(val_data_loader,device)
history_he=training(num_epochs,lr,model_he,train_device_loader,val_device_loader,opt_func,'')

This code does the same thing with weights initialized using he technique.

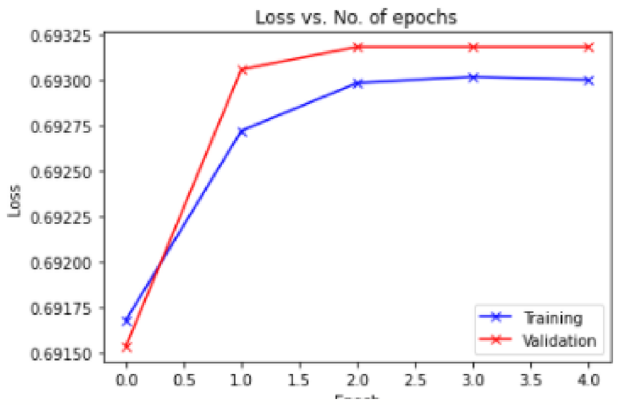
model_he = to_device(CNN('zeros_weights'), device)
train_device_loader=DeviceDataLoader(train_data_loader,device)
val_device_loader=DeviceDataLoader(val_data_loader,device)
history_he=training(num_epochs,lr,model_he,train_device_loader,val_device_loader,opt_func,'')

This code does the same thing with weights initialized as 0.

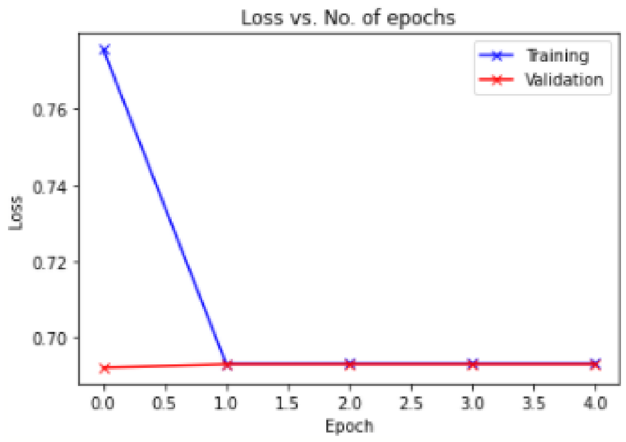
The flow of the code :

First the CNN function is called with the weights initialisation method mentioned. The CNN model which is defined in the function creates all the kernels and Neural Network.Then the train data and test data is loaded and all the loading of models and data is done with the help of to_device and DeviceDataLoader function.Then the training function is called and the model and datas are passed into that function the function calculates the weights of the model and then evaluates function is called to check the loss and score of the model then we call train_validation_loss to plot the graph.

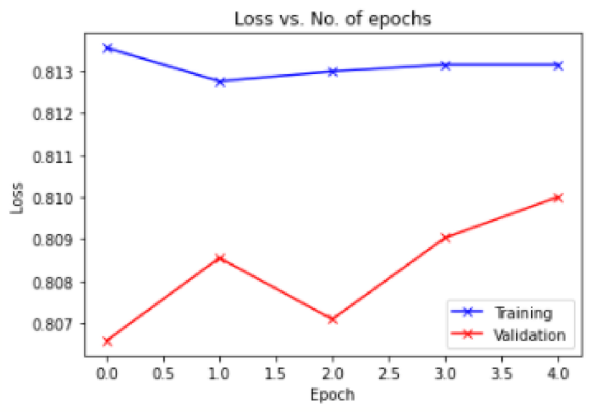
Random Weights Initializing



He Weights Initializing

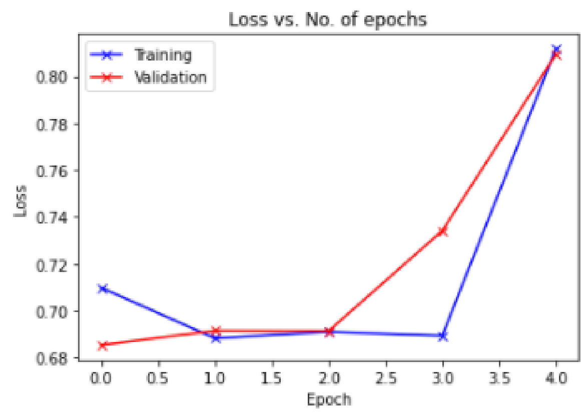


Zero Weights Initializing

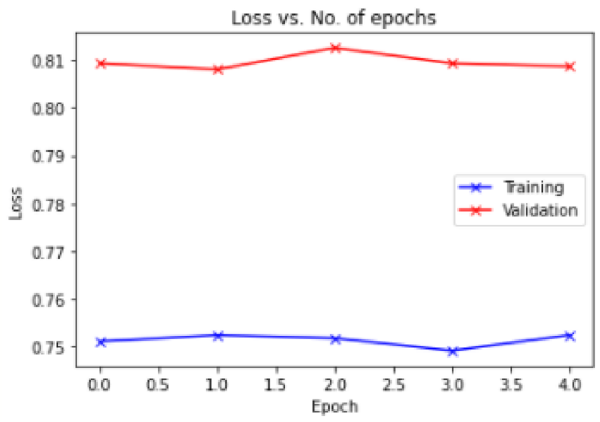


3)So out of all the weights initializing method we are inclined to choose **He initializing** method. We chose **He initializing** method because as we can see from the graph the loss of it has been constant but in other cases it seems like they have crossed their lowest loss and now their losses have an increasing trend.

Model with dropout after convolution

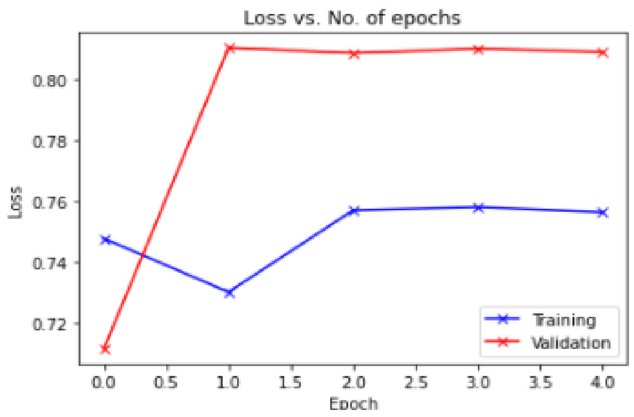


Model with dropout between fully connected layers

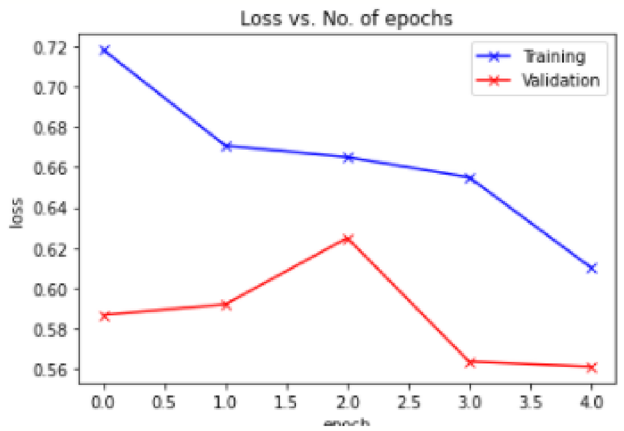


4)We choose the dropout between fully connected layers as the loss is less and dropout after convolution shows an increasing trend

Model with L1 Regularization



Model with L2 Regularization



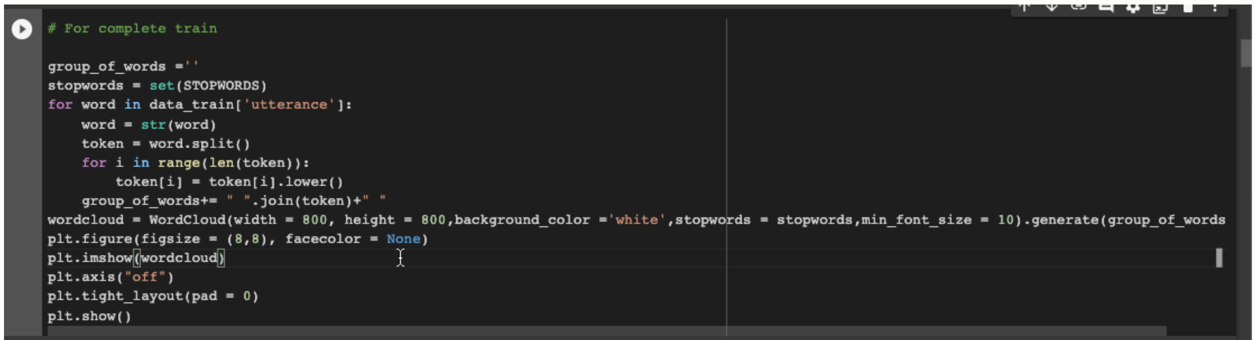
5)L2 regularization is showing a better result. This can be seen through the plot as L2 is showing a descending trend and has a lower loss.

Question 2

data_train=pd.read_csv(train_data)
data_test=pd.read_csv(test_data)
This code reads the test and the training data.

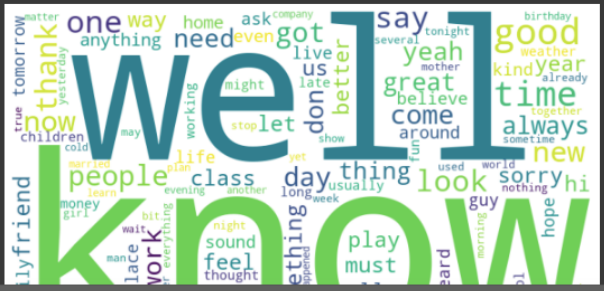
```
wordcloud = WordCloud(width = 800, height = 800,background_color
='white',stopwords = stopwords,min_font_size =
10).generate(group_of_words)
```
:
This code takes the utterance feature from the train data and plots the word. The size of the words in the plot depends upon the frequency in the dataset. So we created 5 plots like this.One plot takes all the dataset and other plots takes the dataset based on the act. The acts are 1,2,3,4.

For Entire Training Dataset

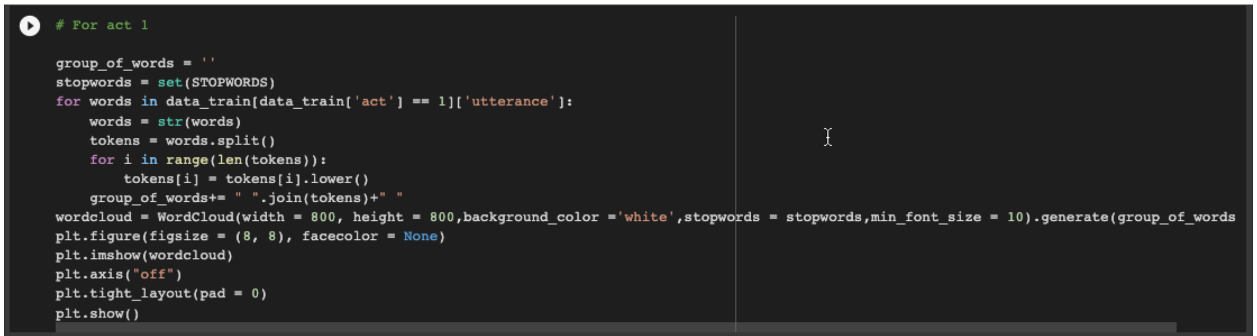


```
# For complete train

group_of_words =''
stopwords = set(STOPWORDS)
for word in data_train['utterance']:
    word = str(word)
    token = word.split()
    for i in range(len(token)):
        token[i] = token[i].lower()
    group_of_words+= " ".join(token)+" "
wordcloud = WordCloud(width = 800, height = 800,background_color ='white',stopwords = stopwords,min_font_size = 10).generate(group_of_words
plt.figure(figsize = (8,8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

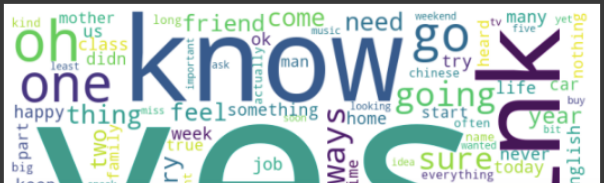


For Act 1

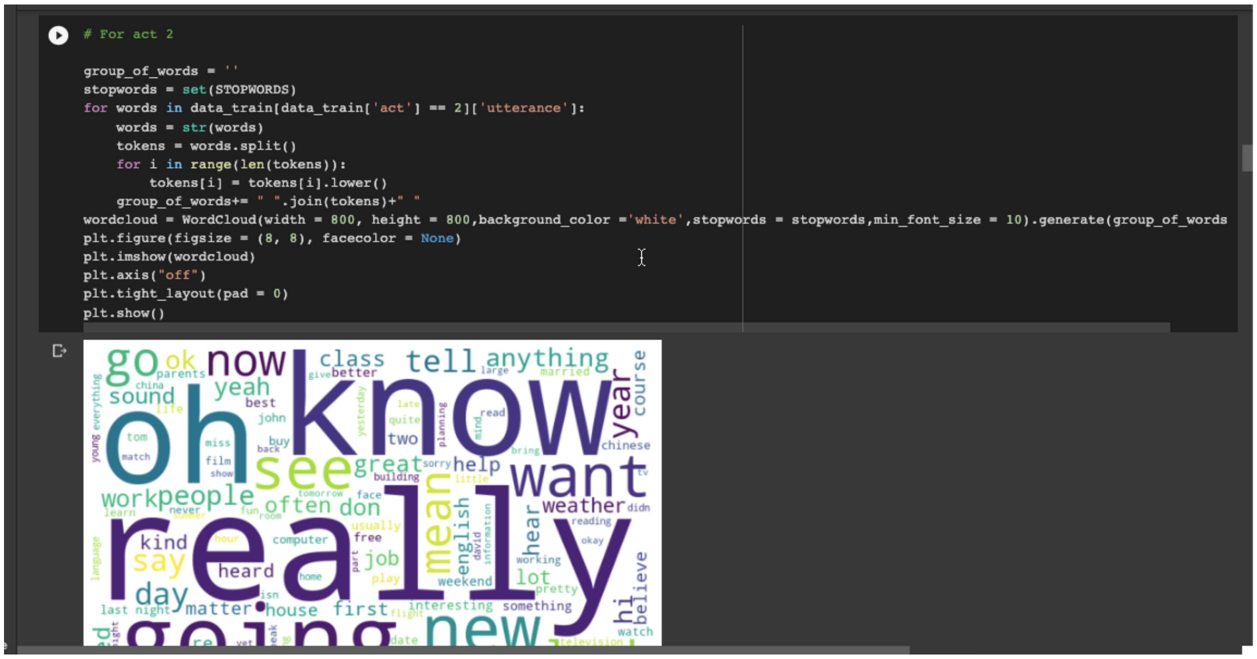


```
# For act 1

group_of_words = ''
stopwords = set(STOPWORDS)
for words in data_train[data_train['act'] == 1]['utterance']:
    words = str(words)
    tokens = words.split()
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()
    group_of_words+= " ".join(tokens)+" "
wordcloud = WordCloud(width = 800, height = 800,background_color ='white',stopwords = stopwords,min_font_size = 10).generate(group_of_words
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



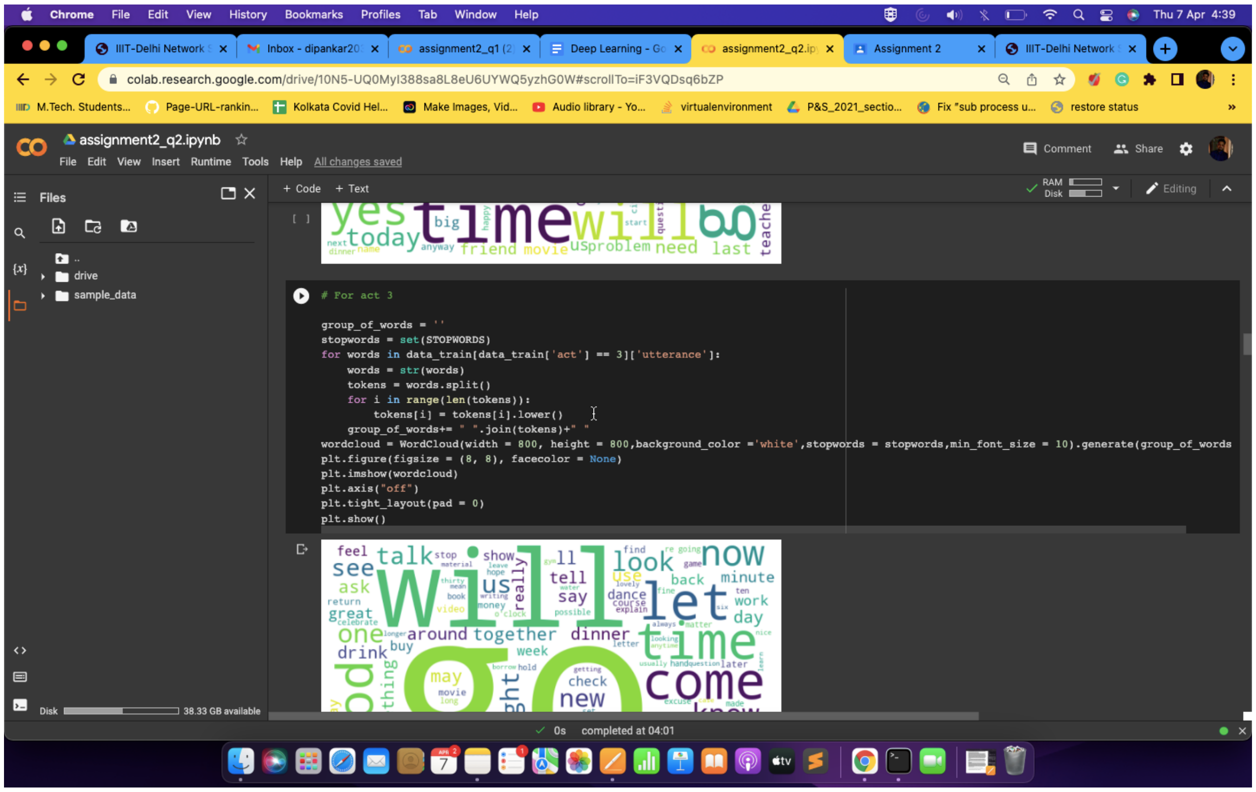
For Act 2



```
# For act 2

group_of_words = ''
stopwords = set(STOPWORDS)
for words in data_train[data_train['act'] == 2]['utterance']:
    words = str(words)
    tokens = words.split()
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()
    group_of_words+= " ".join(tokens)+" "
wordcloud = WordCloud(width = 800, height = 800,background_color ='white',stopwords = stopwords,min_font_size = 10).generate(group_of_words
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

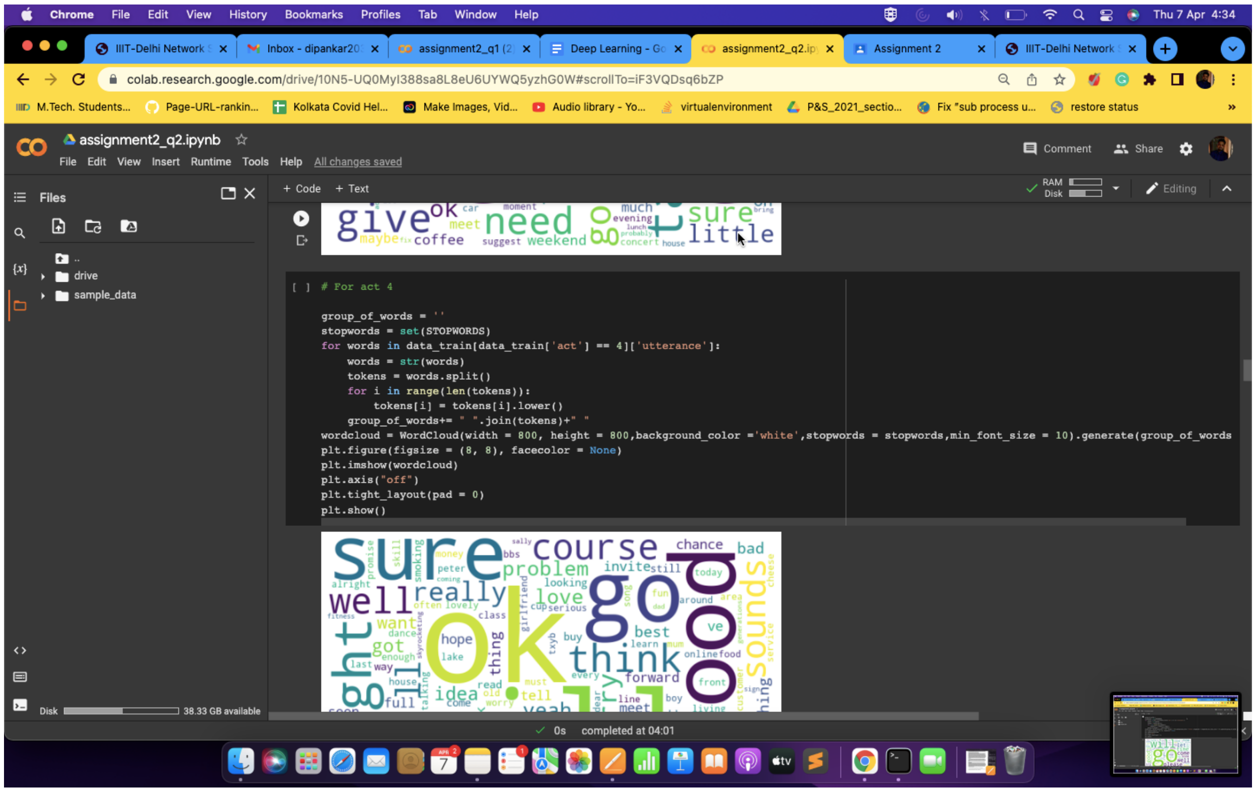
For Act 3



```
# For act 3

group_of_words = ''
stopwords = set(STOPWORDS)
for words in data_train[data_train['act'] == 3]['utterance']:
    words = str(words)
    tokens = words.split()
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()
    group_of_words+= " ".join(tokens)+" "
wordcloud = WordCloud(width = 800, height = 800,background_color ='white',stopwords = stopwords,min_font_size = 10).generate(group_of_words
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

For Act 4



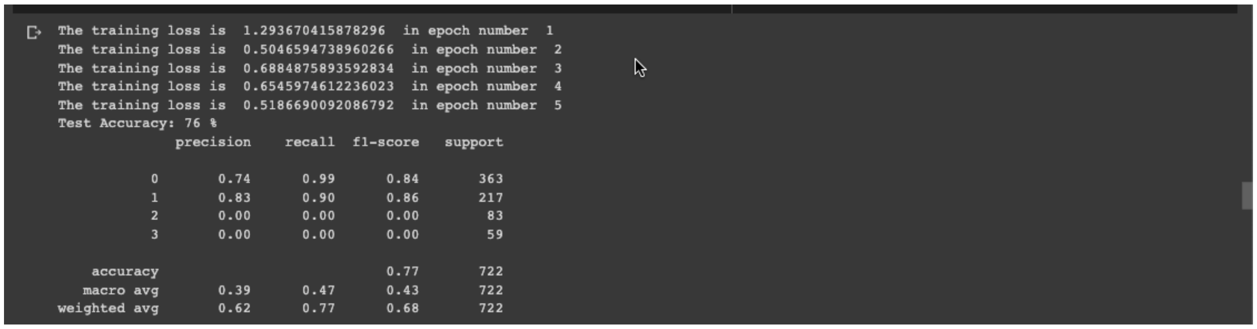
```
# For act 4

group_of_words = ''
stopwords = set(STOPWORDS)
for words in data_train[data_train['act'] == 4]['utterance']:
    words = str(words)
    tokens = words.split()
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()
    group_of_words+= " ".join(tokens)+" "
wordcloud = WordCloud(width = 800, height = 800,background_color ='white',stopwords = stopwords,min_font_size = 10).generate(group_of_words
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

assignment2_q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
sure course chance bad problem invite still today well really love go good sounds ok think best idea tell thing hope got want forward front
0s completed at 04:01

3)

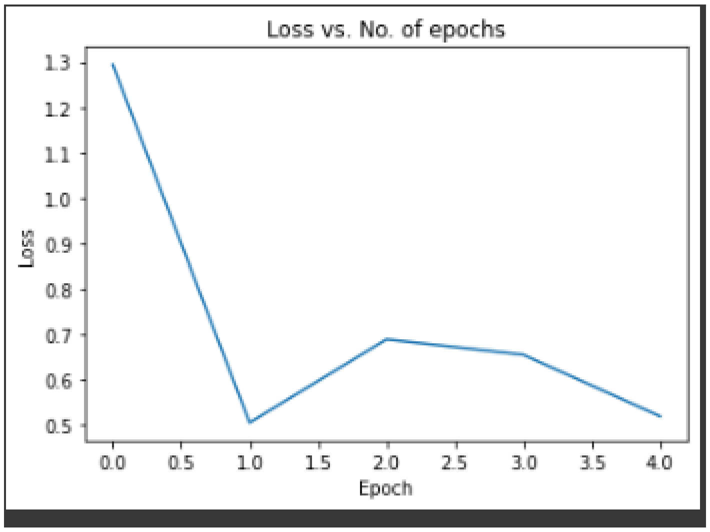
Accuracy and weighted F1 for X=0



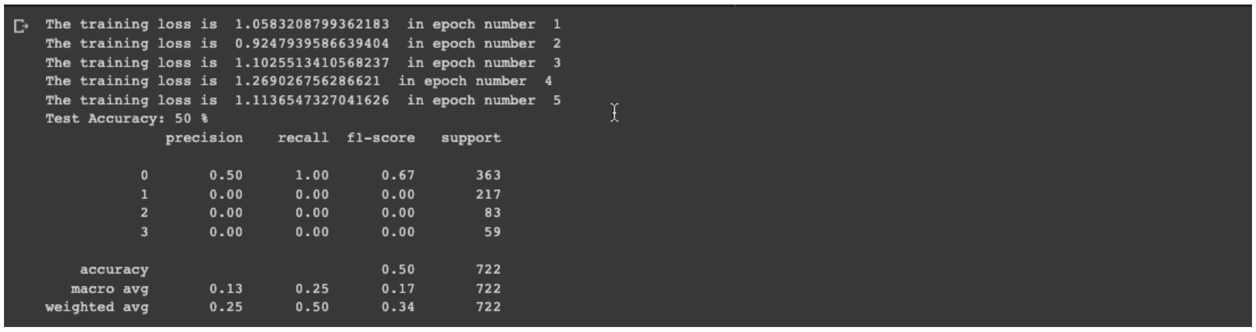
```
The training loss is  1.293670415878296  in epoch number  1
The training loss is  0.5046594738960266  in epoch number  2
The training loss is  0.6884875893592834  in epoch number  3
The training loss is  0.6545974612236023  in epoch number  4
The training loss is  0.5186690092086792  in epoch number  5
Test Accuracy: 76 %
              precision    recall  f1-score   support

           0       0.74      0.99      0.84       363
           1       0.83      0.90      0.86       217
           2       0.00      0.00      0.00        83
           3       0.00      0.00      0.00        59

    accuracy                           0.77       722
   macro avg       0.39      0.47      0.43       722
weighted avg       0.62      0.77      0.68       722
```



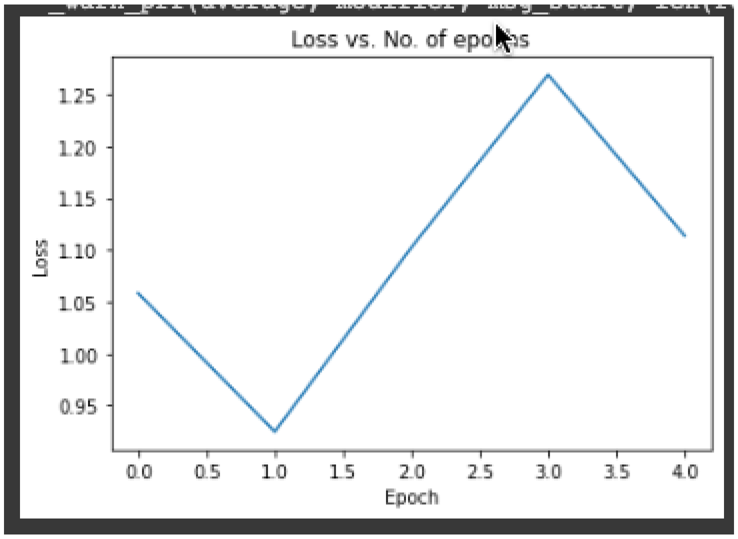
Accuracy and weighted F1 for X=1



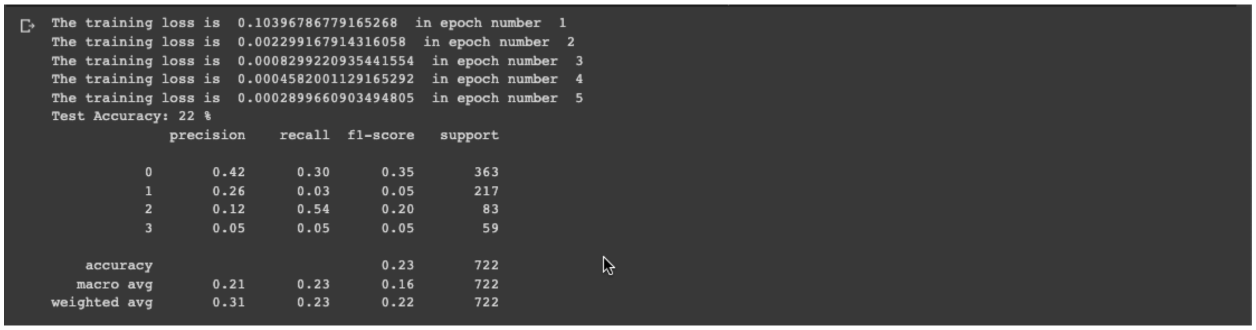
```
The training loss is  1.0583208799362183  in epoch number  1
The training loss is  0.9247939586639404  in epoch number  2
The training loss is  1.1025513410568237  in epoch number  3
The training loss is  1.269026756286621  in epoch number  4
The training loss is  1.1136547327041626  in epoch number  5
Test Accuracy: 50 %
              precision    recall  f1-score   support

           0       0.50      1.00      0.67       363
           1       0.00      0.00      0.00       217
           2       0.00      0.00      0.00        83
           3       0.00      0.00      0.00        59

    accuracy                           0.50       722
   macro avg       0.13      0.25      0.17       722
weighted avg       0.25      0.50      0.34       722
```



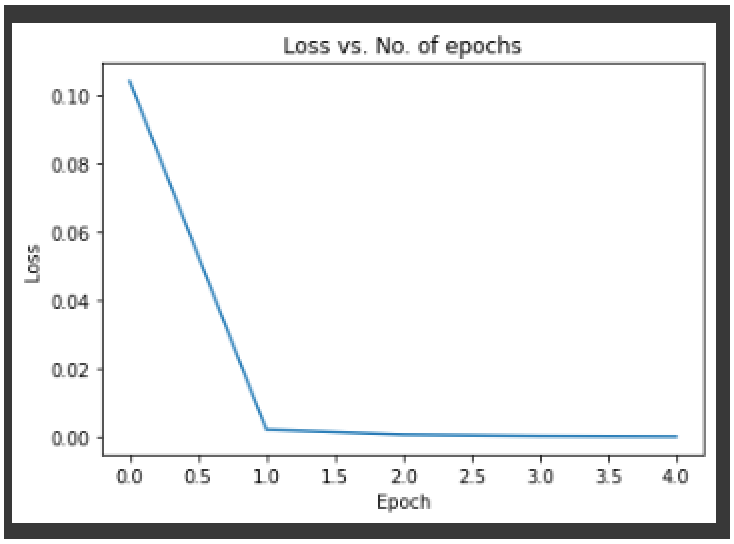
Accuracy and weighted F1 for X=2



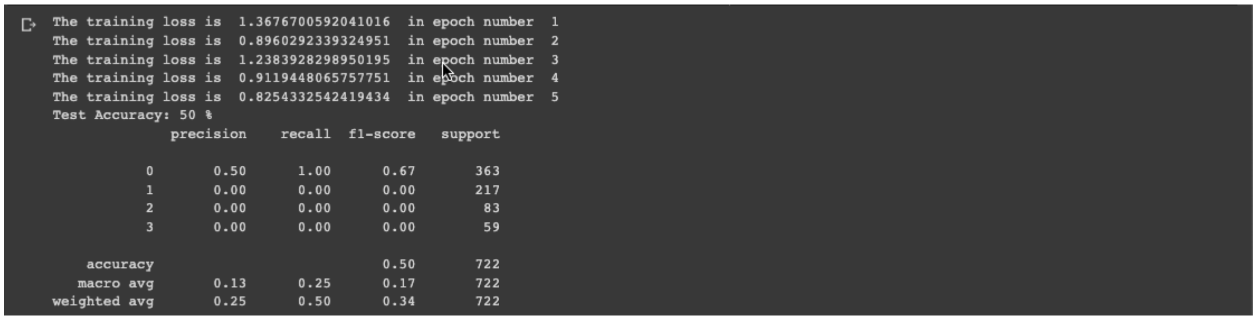
```
The training loss is  0.10396786779165268  in epoch number  1
The training loss is  0.002299167914316058  in epoch number  2
The training loss is  0.0008299220935441554  in epoch number  3
The training loss is  0.0004582001129165292  in epoch number  4
The training loss is  0.0002899660903494805  in epoch number  5
Test Accuracy: 22 %
              precision    recall  f1-score   support

           0       0.42      0.30      0.35       363
           1       0.26      0.03      0.05       217
           2       0.12      0.54      0.20        83
           3       0.05      0.05      0.05        59

    accuracy                           0.23       722
   macro avg       0.21      0.23      0.16       722
weighted avg       0.31      0.23      0.22       722
```



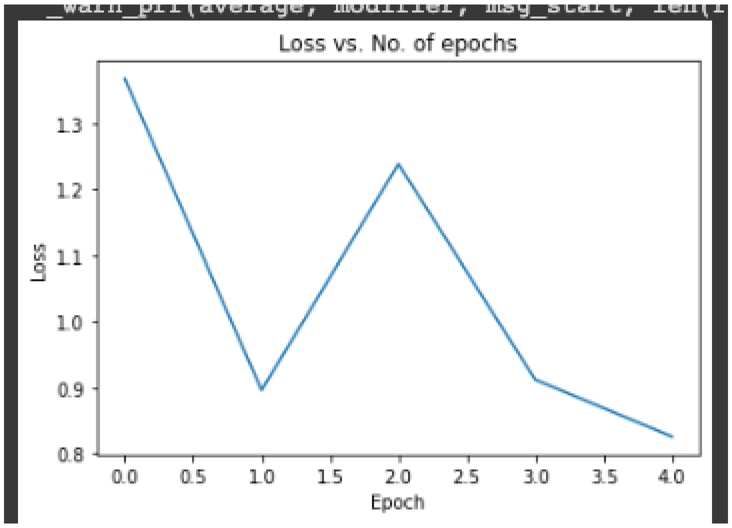
Accuracy and weighted F1 for X=3



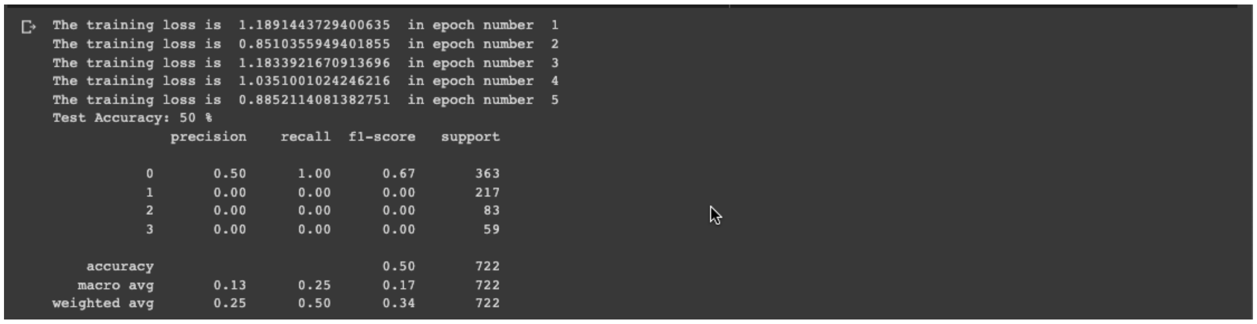
```
The training loss is  1.3676700592041016  in epoch number  1
The training loss is  0.8960292339324951  in epoch number  2
The training loss is  1.2383928298950195  in epoch number  3
The training loss is  0.9119448065757751  in epoch number  4
The training loss is  0.8254332542419434  in epoch number  5
Test Accuracy: 50 %
               precision    recall  f1-score   support

           0       0.50      1.00      0.67       363
           1       0.00      0.00      0.00       217
           2       0.00      0.00      0.00        83
           3       0.00      0.00      0.00        59

    accuracy                           0.50       722
   macro avg       0.13      0.25      0.17       722
weighted avg       0.25      0.50      0.34       722
```



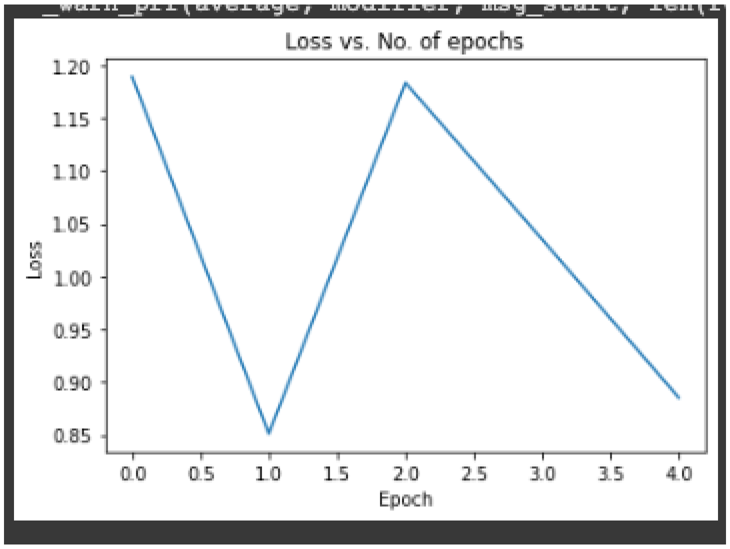
Accuracy and weighted F1 for X=4



```
The training loss is  1.1891443729400635  in epoch number  1
The training loss is  0.8510355949401855  in epoch number  2
The training loss is  1.1833921670913696  in epoch number  3
The training loss is  1.0351001024246216  in epoch number  4
The training loss is  0.8852114081382751  in epoch number  5
Test Accuracy: 50 %
               precision    recall  f1-score   support

           0       0.50      1.00      0.67       363
           1       0.00      0.00      0.00       217
           2       0.00      0.00      0.00        83
           3       0.00      0.00      0.00        59

    accuracy                           0.50       722
   macro avg       0.13      0.25      0.17       722
weighted avg       0.25      0.50      0.34       722
```


```
  _warn_prf(average, modifier, msg_start, len(r
```

4) The performance of the model does not increase with increase in X as seen above sometime it increases and in accordance with the model above it decreases and also increases randomly at times. So we cannot say that it will increase only and not decrease.