Deep Learning - CSE641 - Assignment 4

Name: Bharat Goyal (MT22024), Debnath Kundu (MT22026), Saloni Agrawal (MT22062)

Question1:

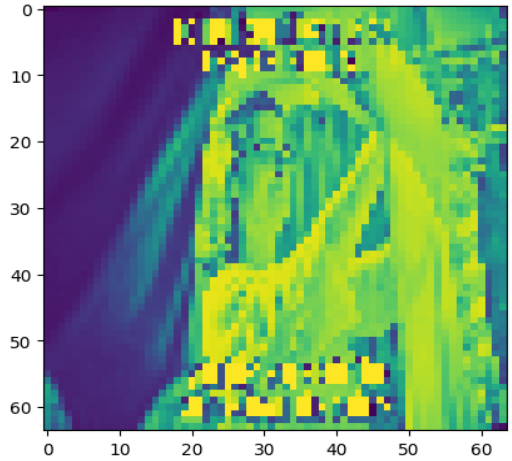
This Question is done by Debnath Kundu

1. *Pre-process the images using at least 2 techniques:*

Random Rotation + Color Jittering + Resize (224) for TRAIN SET:

Random Rotation + Color Jittering + Resize (224)

```
[20]  #TRAIN : Defining the transformation function
      transform_function_train = transforms.Compose([
          # transforms.ToTensor(),
          # transforms.ToPILImage(),
          transforms.RandomRotation(degrees=(-2,2)),
          # transforms.ColorJitter(brightness=0.5, contrast=0.2, saturation=0.4, hue=0.5),
          transforms.ToTensor(),
          transforms.Normalize(mean=[0.485, 0.456, 0.406],
                               std=[0.229, 0.224, 0.225]),
          transforms.Resize([64, 64]),

      ])
```



Normalise + Resize (224) for VALIDATION + TEST SET:

Normalise + Resize for Validation & Test data

```
#VAL + TEST : Defining the transformation
transform_function_val_test = transforms.Compose([
    # transforms.ToTensor(),
    # transforms.ToPILImage(),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225]),
    transforms.Resize([64, 64]),

])
```

The VGG16 takes (224,224) size images as input. So all the images are resized to that dimension.

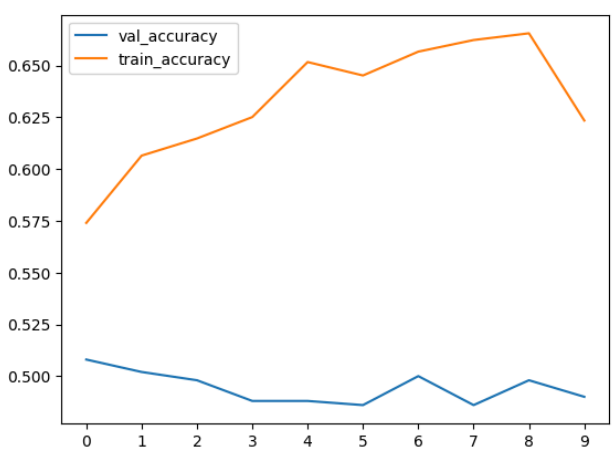
2. *Propose and implement an image-only model for classification.*

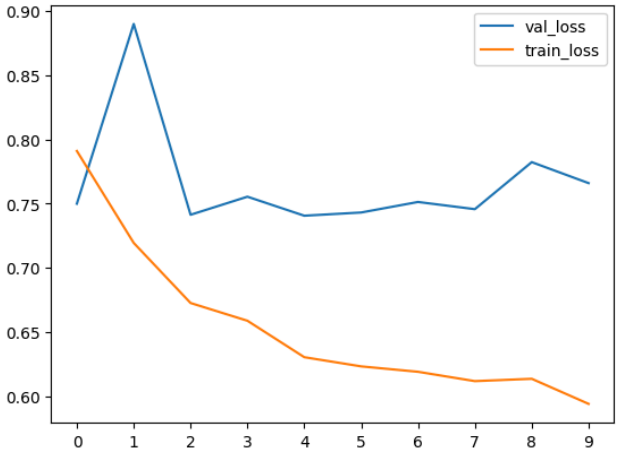
VGG 16

VGG 16

```
from torchvision import models, transforms

# load pretrained vgg16 from PyTorch as an instance
# need to make setting 'internet' to 'On'.
use_pretrained = True
net = models.vgg16(pretrained=use_pretrained)

# Replace output layer for 2 class classifier, 'HATEFUL' and 'NOT HATEFUL'.
net.classifier[6] = nn.Linear(in_features=4096, out_features=2)

net.train()
```

3. *Generate the following plots:*





*The training **accuracy** reached a peak of 0.65, while validation accuracy was low at 0.50. Since the training accuracy is higher than the validation accuracy, hence there must be over-fitting.*

*Since the validation **loss** is higher than the train loss, hence it clearly explains the over-fitting issue.*

4. *Report the overall **Accuracy, Precision, Recall, F1 score** for your test set. Also, report class-wise precision and recall and F1 score for test set.*

```
import sklearn.metrics as metrics
from sklearn.metrics import confusion_matrix

confusion_matrix(labels_l, predictions_l)
print("Classification report for V :\n%s\n"
      % (metrics.classification_report(labels_l, predictions_l)))
```

```
Classification report for V :
              precision    recall  f1-score   support

           0       0.51      0.90      0.65       508
           1       0.51      0.11      0.18       490

    accuracy                           0.51       998
   macro avg       0.51      0.50      0.42       998
weighted avg       0.51      0.51      0.42       998
```

Accuracy: 0.51

Classwise precision, recall & f1-score are dislayed above.

Question2:

This question is done by Saloni Agrawal

1. Data is preprocessed by removing unwanted spaces, symbols followed by tokenizing, padding it to max length.

   Some piece of codes:

```
      def preprocess_text(text):
          # Remove URLs
          text = re.sub(r'http\S+', '', text)
          # Remove hashtags
          text = re.sub(r'#\S+', '', text)
          # Remove special characters and punctuation
          text = re.sub(r'[^\w\s]', '', text)
          text = text.lower()
          #text = text.translate(str.maketrans('', '', string.punctuation))
          words = text.split()
          return words
```

```
def tokenizer_text(text, word2index, max_seq_length):
    words = preprocess_text(text)
    sequence = [word2index[w] for w in words if w in word2index]
    sequence = pad_sequences([sequence], maxlen=max_seq_length, padding='post',
truncating='post')
    return np.squeeze(sequence)
```

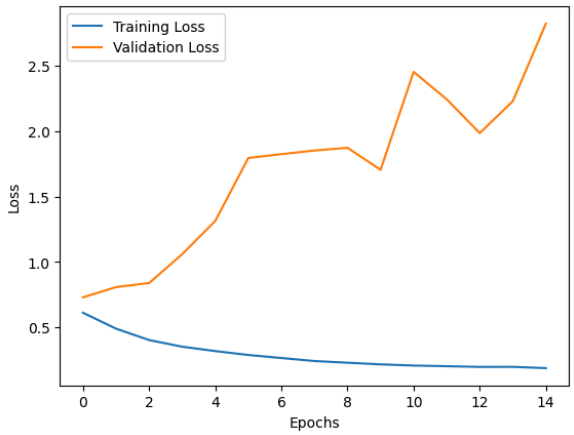
```
def create_vocab(data):
    vocab = set()
    for d in data:
        words = preprocess_text(d['text'])
        print(words)
        vocab.update(words)
    word2index = {w: i for i, w in enumerate(vocab)}
    return word2index
```

2. In this second part, I have applied Long short term memory(LSTM) model containing 2 hidden layer containing 64 neurons each. I have used sigmoid as activation function, binary cross entropy to calculate the loss and adam as an optimiser. Model is trained for 15 epochs.

```
model = Sequential([
        Embedding(vocab_size, embedding_dim, input_length=max_seq_length),
    LSTM(hidden_dim, return_sequences=True),
    LSTM(hidden_dim),
    Dense(1, activation='sigmoid')
])
```

```
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

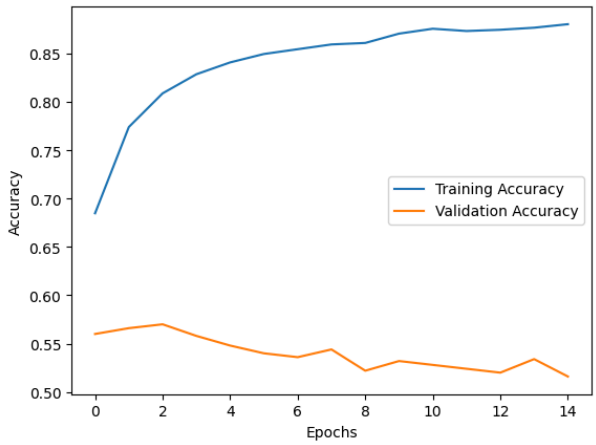
3. **● Loss plot - Training Loss and Validation Loss V/s Epochs.**




```
train_loss: 0.1869 and val_loss: 2.8232
```

Observation: As here, we can see that validation loss is increasing while training loss is decreasing; this shows that our model is trying to overfit the dataset.

● Accuracy plot - Training Accuracy, Validation Accuracy V/s Epochs




```
train_accuracy: 0.8804 and val_accuracy: 0.5160
```

Observation: As here, we can see that validation accuracy is decreasing while training accuracy is increasing. This shows that our model is trying to overfit the dataset.

.

4. On test set we get,

```
Accuracy: 0.557
Class 0 precision: 0.54
Class 0 recall: 0.84
Class 0 F1 score: 0.66
Class 1 precision: 0.61
Class 1 recall: 0.26
```

```
Class 1 F1 score: 0.37
```

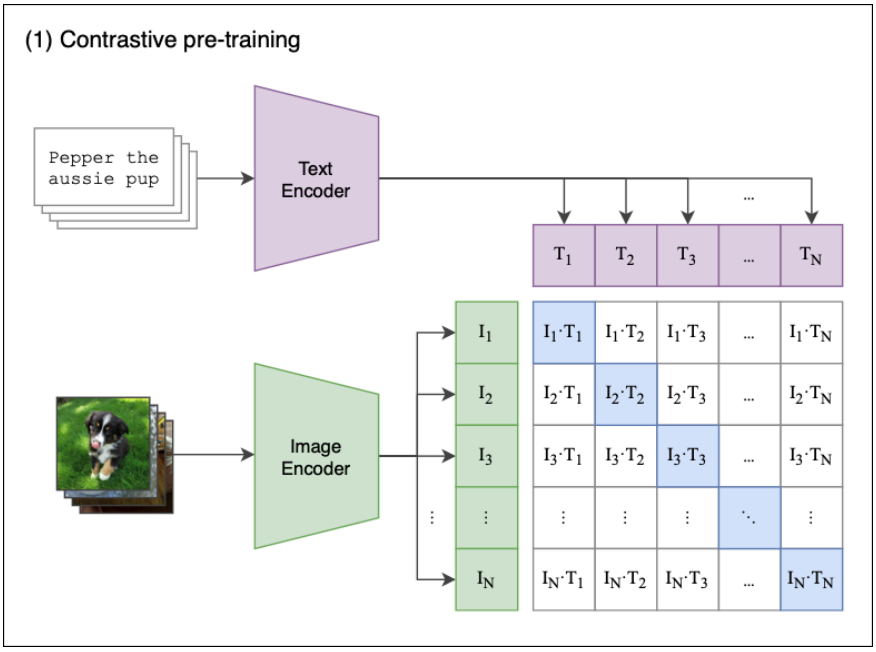
```
Classification Report:
                precision    recall  f1-score   support

           0       0.54      0.84      0.66       510
           1       0.61      0.26      0.37       490

    accuracy                           0.56      1000
   macro avg       0.58      0.55      0.51      1000
weighted avg       0.58      0.56      0.52      1000
```

Question3:

This question is done by Bharat Goyal

Part A)

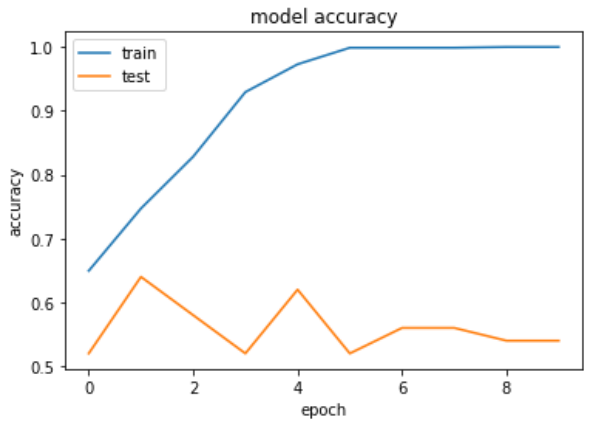
I have tried several architectures for this problem. It includes the LSTM + CNN model , VG16 + LSTM Model and final pretrained multi modal CLIP model. First 2 models seemed promising but didn't perform well. On the other hand, the CLIP model performed far better than the other two. This clip model uses early fusion techniques to tackle this kind of problem. CLIP (Contrastive Language-Image Pre-Training) is a state-of-the-art deep learning model developed by OpenAI in 2021 that can learn to understand both natural language and images at the same time. The CLIP model architecture is composed of a transformer-based neural network that can process both images and text simultaneously. The architecture consists of an encoder network for processing images, and another encoder network for processing text.



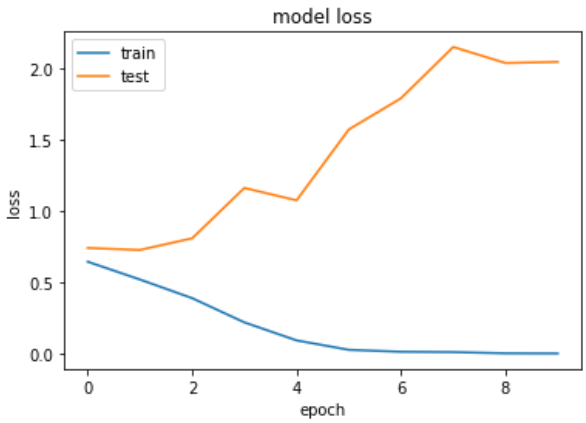
Part B)

I am attaching all the graphs for the results:

In the results through the graph it can be observed that the model is trying to overfit. As the training accuracy is very nice but validation accuracy is very low. Yet it shows that it is performing better than others. Even through loss it can be observed that the model is trying to overfit the data set. As validation loss is increasing and training loss is decreasing.



Accuracy vs Epoch Graph



Loss vs Epoch Graph

Part C)

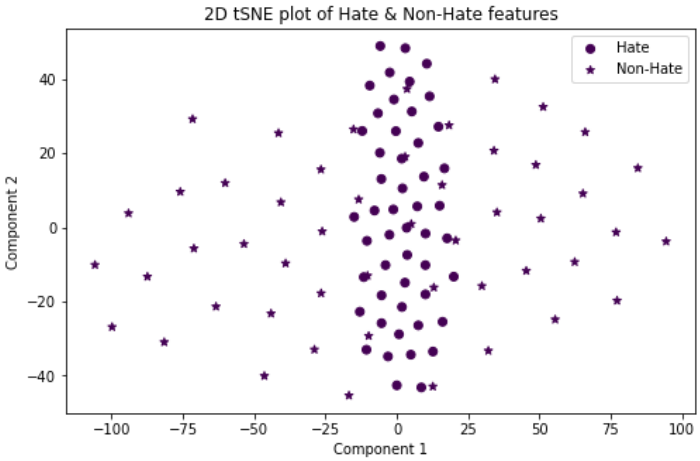
Class Wise accuracy,precision,F1-Score,recall for Model can be stated as,

	precision	**recall**	**f1-score**	**support**
0	0.56	0.69	0.62	505
1	0.57	0.43	0.49	485
accuracy			0.56	990
macro avg	0.57	0.56	0.55	990
weighted avg	0.57	0.56	0.56	990

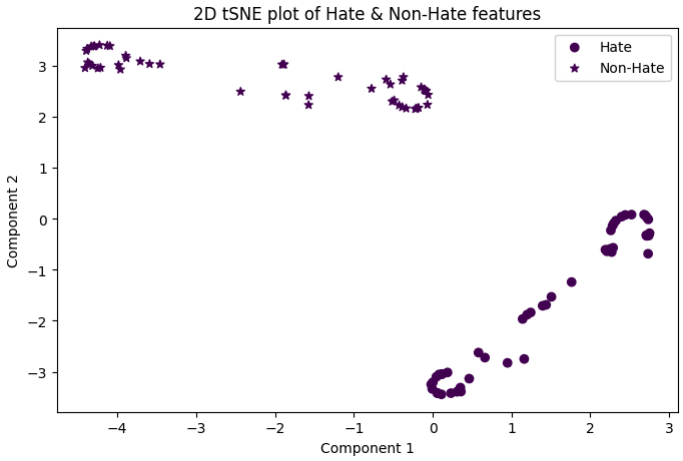
Part D)

TSNE visualization for all the three Models can be seen as :

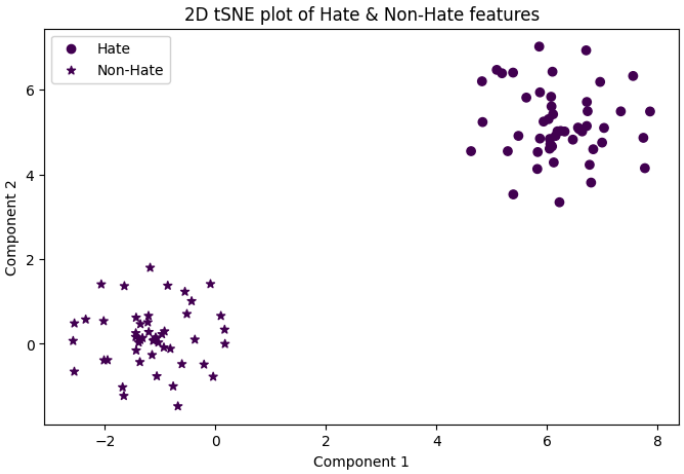
CLIP Model:



TSNE for LSTM Model :



TSNE for VGG16 Model:



Explanation :

Since the task is for image classification, so **VGG16 model** (vision based) works better than the **LSTM model** (text-based). For the classification of hateful memes, picture context (modality) is more important than text modality. So, the tSNE plot in vision model has more separable classes than the text model.

In the **multimodal** model, the accuracy has improved, and the features are also separable. Roughly from -25 to +25, the plot represents hateful memes. Beyond this range, the plot represents non-hateful images. It explains the fact in reality that hate is an extreme emotion, **i.e.,** the context of the meme will be either too negative or too positive to be classified as hateful.