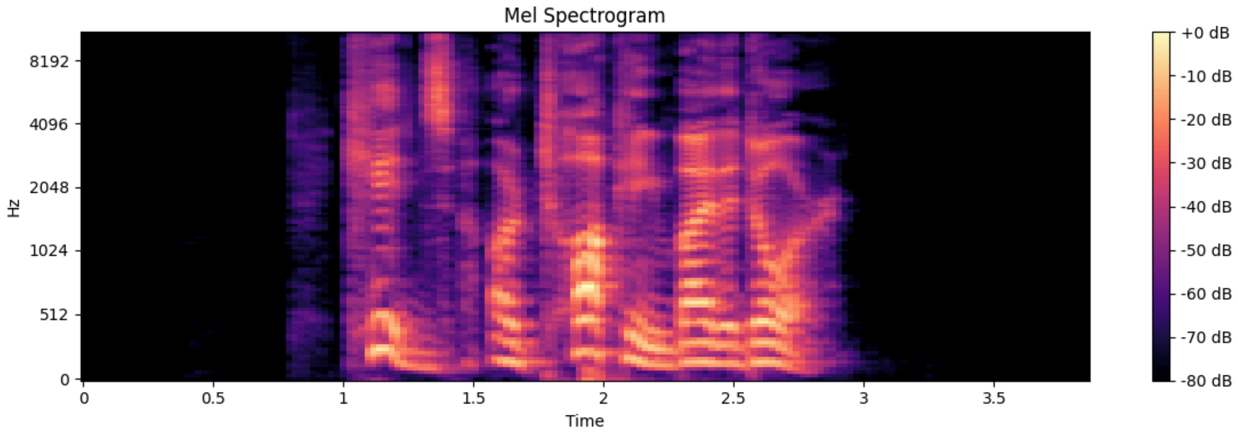
Note : The dataset of RAVDESS and the X_spectrograms.npy file cannot be pushed into GitHub due to large size.

Output for converting audio clips into spectrograms for training CNN:

```
0it [00:00, ?it/s]
100%|██████████| 60/60 [00:59<00:00,  1.02it/s]
100%|██████████| 60/60 [00:50<00:00,  1.20it/s]
100%|██████████| 60/60 [00:50<00:00,  1.20it/s]
100%|██████████| 60/60 [00:47<00:00,  1.27it/s]
100%|██████████| 60/60 [00:48<00:00,  1.24it/s]
100%|██████████| 60/60 [00:47<00:00,  1.25it/s]
100%|██████████| 60/60 [00:46<00:00,  1.29it/s]
100%|██████████| 60/60 [00:51<00:00,  1.16it/s]
100%|██████████| 60/60 [00:50<00:00,  1.18it/s]
100%|██████████| 60/60 [00:49<00:00,  1.21it/s]
100%|██████████| 60/60 [00:46<00:00,  1.28it/s]
100%|██████████| 60/60 [00:46<00:00,  1.30it/s]
100%|██████████| 60/60 [00:48<00:00,  1.23it/s]
100%|██████████| 60/60 [00:48<00:00,  1.23it/s]
100%|██████████| 60/60 [00:47<00:00,  1.28it/s]
100%|██████████| 60/60 [00:48<00:00,  1.25it/s]
100%|██████████| 60/60 [00:45<00:00,  1.33it/s]
100%|██████████| 60/60 [00:55<00:00,  1.08it/s]
100%|██████████| 60/60 [00:45<00:00,  1.31it/s]
100%|██████████| 60/60 [00:52<00:00,  1.15it/s]
100%|██████████| 60/60 [00:47<00:00,  1.26it/s]
100%|██████████| 60/60 [00:53<00:00,  1.13it/s]
100%|██████████| 60/60 [00:51<00:00,  1.17it/s]
100%|██████████| 60/60 [00:48<00:00,  1.23it/s]

Spectrogram :
```



CNN Training with the obtained spectrograms :

```
Epoch 1, Loss: 2.0329, Train Acc: 19.27%, Test Acc: 21.18%
Epoch 2, Loss: 2.0112, Train Acc: 18.06%, Test Acc: 11.81%
Epoch 3, Loss: 1.9640, Train Acc: 24.13%, Test Acc: 13.19%
Epoch 4, Loss: 1.9763, Train Acc: 22.74%, Test Acc: 16.67%
Epoch 5, Loss: 1.9026, Train Acc: 24.91%, Test Acc: 19.79%
Epoch 6, Loss: 1.8814, Train Acc: 28.12%, Test Acc: 27.08%
Epoch 7, Loss: 1.8108, Train Acc: 31.25%, Test Acc: 23.96%
Epoch 8, Loss: 1.8085, Train Acc: 31.34%, Test Acc: 28.82%
Epoch 9, Loss: 1.7711, Train Acc: 30.56%, Test Acc: 30.56%
Epoch 10, Loss: 1.7501, Train Acc: 34.81%, Test Acc: 38.54%
```

```
This suggests that accuracy for testing data is low and needs further
stronger models or good improvements in testing.
This can be further improved.
```

1) Converting audio into text :

```
100%|████████████████████████████████████████| 139M/139M [00:01<00:00,
76.5MiB/s]
Transcribing actor: Actor_19
Transcribing actor: Actor_21
Transcribing actor: Actor_20
Transcribing actor: Actor_17
Transcribing actor: Actor_15
Transcribing actor: Actor_18
Transcribing actor: Actor_22
Transcribing actor: Actor_23
Transcribing actor: Actor_24
Transcribing actor: Actor_16
Transcribing actor: Actor_06
Transcribing actor: Actor_13
Transcribing actor: Actor_10
Transcribing actor: Actor_07
Transcribing actor: Actor_14
Transcribing actor: Actor_11
Transcribing actor: Actor_05
Transcribing actor: Actor_08
Transcribing actor: Actor_09
Transcribing actor: Actor_12
Transcribing actor: Actor_04
Transcribing actor: Actor_01
Transcribing actor: Actor_03
Transcribing actor: Actor_02
```

2) Adding simulated texts to sentences for training text RNN :

Here in the dataset , simulated sentences play a major role in transcripts because they signify the emotion for training text RNN.

3) Training text RNN :

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Epoch 1, Loss: 1.3968, Training Accuracy: 58.59%, Testing Accuracy: 81.25%
Epoch 2, Loss: 0.3010, Training Accuracy: 90.36%, Testing Accuracy:
100.00%
Epoch 3, Loss: 0.0408, Training Accuracy: 100.00%, Testing Accuracy:
100.00%
Epoch 4, Loss: 0.0107, Training Accuracy: 100.00%, Testing Accuracy:
100.00%
Epoch 5, Loss: 0.0060, Training Accuracy: 100.00%, Testing Accuracy:
100.00%
Epoch 6, Loss: 0.0041, Training Accuracy: 100.00%, Testing Accuracy:
100.00%
```

```
Epoch 7, Loss: 0.0030, Training Accuracy: 100.00%, Testing Accuracy:
100.00%
Epoch 8, Loss: 0.0023, Training Accuracy: 100.00%, Testing Accuracy:
100.00%
Epoch 9, Loss: 0.0019, Training Accuracy: 100.00%, Testing Accuracy:
100.00%
Epoch 10, Loss: 0.0016, Training Accuracy: 100.00%, Testing Accuracy:
100.00%
```

This is the output for testing data as per training data.

Phase 2 : Early fusion

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Epoch 1, Loss: 2.0587, Training Accuracy: 16.41%
Epoch 2, Loss: 1.5395, Training Accuracy: 36.28%
Epoch 3, Loss: 1.3182, Training Accuracy: 40.97%
Epoch 4, Loss: 1.1277, Training Accuracy: 47.14%
Epoch 5, Loss: 0.9037, Training Accuracy: 56.86%
Epoch 6, Loss: 0.7458, Training Accuracy: 65.10%
Epoch 7, Loss: 0.6819, Training Accuracy: 70.14%
Epoch 8, Loss: 0.5456, Training Accuracy: 75.43%
Epoch 9, Loss: 0.2755, Training Accuracy: 95.66%
Epoch 10, Loss: 0.0866, Training Accuracy: 100.00%
Test Accuracy: 100.00%
```

Phase 2: Late Fusion

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Epoch 1, Loss: 2.0724, Training Accuracy: 13.54%
Epoch 2, Loss: 1.9986, Training Accuracy: 18.14%
Epoch 3, Loss: 1.6924, Training Accuracy: 27.52%
Epoch 4, Loss: 1.6855, Training Accuracy: 26.56%
Epoch 5, Loss: 1.6543, Training Accuracy: 30.21%
Epoch 6, Loss: 1.2276, Training Accuracy: 44.97%
Epoch 7, Loss: 0.6036, Training Accuracy: 73.35%
Epoch 8, Loss: 0.2388, Training Accuracy: 87.67%
Epoch 9, Loss: 0.1579, Training Accuracy: 94.27%
Epoch 10, Loss: 0.0454, Training Accuracy: 99.39%
```

```
Test Accuracy: 99.65%
```

Transformer of Audio Model using AST :

```
Epoch 1: Train Loss = 2.1975, Accuracy = 11.81%
Test Accuracy: 12.50%
Epoch 2: Train Loss = 2.1072, Accuracy = 13.63%
Test Accuracy: 12.50%
Epoch 3: Train Loss = 2.0848, Accuracy = 12.67%
Test Accuracy: 14.24%
Epoch 4: Train Loss = 2.0811, Accuracy = 12.15%
Test Accuracy: 12.50%
Epoch 5: Train Loss = 2.0830, Accuracy = 13.11%
Test Accuracy: 15.28%
Epoch 6: Train Loss = 2.0896, Accuracy = 12.41%
Test Accuracy: 15.28%
Epoch 7: Train Loss = 2.0914, Accuracy = 12.93%
Test Accuracy: 15.28%
Epoch 8: Train Loss = 2.0791, Accuracy = 12.85%
Test Accuracy: 14.24%
Epoch 9: Train Loss = 2.0786, Accuracy = 13.37%
Test Accuracy: 13.19%
Epoch 10: Train Loss = 2.0846, Accuracy = 14.06%
Test Accuracy: 13.54%
```

Poetry : Dependency managing tool :

1) Install poetry :

```
curl -sSL https://install.python-poetry.org | python3 -
poetry --version
```

2) Create a new project

```
poetry new my_emotion_project
cd my_emotion_project
```

3) Add dependencies :

```
poetry add numpy pandas matplotlib seaborn torch torchvision torchaudio tensorflow transformers librosa nltk scikit-learn opencv-python
```

```
poetry add git+https://github.com/openai/whisper.git
```

4) Final pyproject.toml file :
    Refer to file for more details

5) Install all dependencies

```
poetry install
```

6) Run the code as follows :

```
poetry run python my_emotion_project/main.py
```

Requirements.txt file :
Use “pip install -r requirements.txt” command in Ubuntu terminal to install all the required libraries for requirements.txt file.

Additional improvements in the Multi modal emotion recognition from Audio and Transcripts project :

1) Usage of a Transformer based text model instead of Text RNN like DistilBERT , BERT , Whisper encoder .
2) Many more built in features for evaluation metrics can be used instead of directly calculating accuracy .

3) Instead of using same sentences again and again for generating simulated transcripts , one thing we can use to improve semantics and vocabulary is taking some bunch of sentences for each emotion and iterate over each bunch to add them in the transcripts(original) that match with the emotion so that the RNN gets exposed to more vocabulary and it may also help for training different dataset other than the given RAVDESS.
4) We can use something like soft labels instead of directly creating an emotion map and using it to get exact emotion.(hard labels). Then we can represent each audio clip label with % of each emotion in that clip instead of just focussing on dominant emotion.
5) Using pre-trained audio models can be also good for this task.
6) Also one more thing we can use is the process of Early stopping where we stop the epochs and training if the loss is not changing much i.e converging to a point in order to save computing power and time.
7) To improve accuracy for the models, we can use deeper neural networks and also use dropout etc. We can fine-tune the transformer model.