Nanyang Technological University

Academic Year 2023/2024 Semester 2

SC4003: Neural Network and Deep Learning

Assignment 2: Speech Emotion Recognition (Group Project)

Date of Submission: 14 April 2024

Name:	**Matriculation Number:**
Hoo Jian Le	U2120301C
Ma Wenyue	U2120727K
Wong Wei Kai	U2121476K

1. Introduction to assignment........ 3
2. Review of existing techniques........ 3
  2.1. Convolutional Neural Network (CNN):........ 3
  2.2. Transfer Learning:........ 3
3. Datasets........ 4
  3.1. EmoDB Dataset........ 4
  3.2. RAVDESS Dataset........ 4
  3.3. Overview........ 4
4. Methods........ 5
  4.1. CNN Implementation........ 5
  4.1.1. Preprocessing........ 5
  4.1.2. Model and Training........ 5
  4.2. Transfer learning........ 5
  4.2.1. Preprocessing........ 6
  4.2.1. Model (Wav2Vec2)........ 6
  4.2.3. Training........ 6
5. Experiment and Results........ 8
6. Detecting emotions dynamically:........ 10
7. Conclusion:........ 11
References........ 12

1. Introduction to assignment

Speech Emotion Recognition (SER) is the process of predicting human emotions from audio signals using Artificial Intelligence (AI) and Deep Learning techniques. Although humans can discern emotions conveyed through speech, AI lacks the same capability. Through this assignment, we aim to create a model that can recognize emotions through speech intervals.

2. Review of existing techniques

SER is not a novel concept, but rather a field that has been extensively investigated over the past decade. In this section, we will be discussing some of the existing techniques used for SER.

2.1. Convolutional Neural Network (CNN):

CNN, even though more commonly associated with image processing tasks, is also used for SER. CNN uses convolutional layers to extract features from input, pooling layers to reduce spatial dimensions on feature maps, and finally the fully connected layer to perform classifications [3]. In the context of SER, the input takes in spectrograms of the audio, which is treated as an image where the intensity of each pixel represents the magnitude of corresponding frequency components.

2.2. Transfer Learning:

Transfer learning is a method widely used to improve performance while reducing the need for large amounts of data. By leveraging on models pre-trained on large amounts of dataset online, these pre-trained models are suited to doing very general things. This also means that for very specific purposes (SER in this case), the model must be further fine-tuned on smaller emotion-specific datasets.

One of the more popular approaches to transfer learning involves the use of HuggingFace, which is a company and open-source community that specializes in machine learning technologies. Their platform provides access to pre-trained models, fine-tuning capabilities, and collaborative spaces for sharing datasets and insights, all aimed at advancing the field of audio processing and making it more accessible to a wider audience.

3. Datasets

We are provided with 2 datasets, both of which contain audio clips spoken with a clear emotion from professional speakers.

3.1. EmoDB Dataset

The EmoDB [2] database contains German audio clips comprising of seven emotions, those being: Anger (W), Boredom (L), Disgust (E), Anxiety/Fear (A), Happiness (F), Sadness (T), Neutral (N) An example of the file name would be: 03a01Fa.wav We can observe that the 6th position of the filename is mapped to one of the seven emotions mentioned above where in this case that particular .wav file is an audio clip spoken with the Happiness emotion.

3.2. RAVDESS Dataset

The RAVDESS [1] database contains English audio clips comprising of eight emotions, those being: Neutral (01), Calm (02), Happy (03), Sad (04), Angry (05), Fearful (06), Disgust (07), Surprised (08). An example of the file name would be: **03-01-01-01-01-01-01.wav** (375.72 kB) .Based on the documentation provided, the third segment of the filename is mapped to one of the eight emotions mentioned above where in this case that particular .wav file is an audio clip spoken with the Neutral emotion.

3.3. Overview

These observations are crucial when it comes to labelling our dataset, which must be done in order to properly train the model via supervised learning.

Another thing to note is that while it is tempting to combine the two datasets into one giant dataset which provides more training data, certain implications will arise when we combine the two conflicting datasets.

- EmoDB dataset is spoken in German, while RAVDESS dataset is spoken in English. This difference in language results in different expression of emotions through tones, intonations etc...
- EmoDB dataset is sampled at an audio rate of 16000Hz, whereas RAVDESS dataset is sampled at 48000Hz. The difference in sampling rate results in RAVDESS dataset containing more nuanced information, which the model can recognize subtle variations in emotion for RAVDESS but not EmoDB. To use both datasets together, the RAVDESS dataset needs to be resampled to 16000Hz beforehand.

For our model training, we have decided to train the model separately first, then train it with the combined dataset to see the differences.

4. Methods

In this section, we will explore our CNN implementation for SER as well as the utilisation of pre-trained models for transfer learning, refining them through fine-tuning to suit our specific classification requirements.

4.1. CNN Implementation

For the CNN implementation, we decided to focus on MFCCs as the main input to be used to train our model, which represents the spectral characteristics of a sound, thereby emphasizing features of an audio signal that are important for speech processing while discarding irrelevant information.

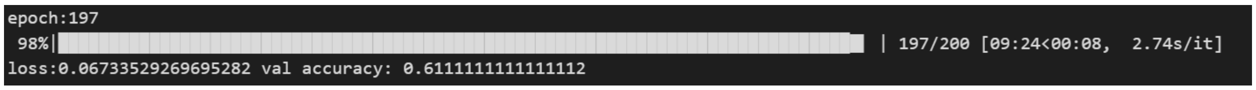
4.1.1. Preprocessing

For preprocessing, we ensured that every single audio file was of equal length. This would mean that we would trim down audio files longer than the max duration, and zero-pad audio files shorter than the max duration. The MFCC is then extracted to be used as the input for training our model.

4.1.2. Model and Training

The model we settled on consists of 5 convolutional layers, 1 fully connected layer and an output layer, with each layer consisting of a 2D convolutional layer, an ELU activation function and a 2D max pooling. We also implemented each layer with dropout as a regularization technique to prevent overfitting of the training data.

We trained our model for 200 epochs, checking the validation accuracy along the way. The results show that our highest validation accuracy came from epoch 197, with an accuracy of 61.1%



```
epoch:197
 98%|█████████████████████████████████████████████████████████████▌ | 197/200 [09:24<00:08,  2.74s/it]
loss:0.06733529269695282 val accuracy: 0.6111111111111112
```

4.2. Transfer learning

For transfer learning, we decided to focus on a pre-trained model Wav2Vec2 from Hugging Face.

4.2.1. Preprocessing

For preprocessing, Hugging Face comes equipped with a pre-existing feature extractor class (AutoFeatureExtractor). This saves us the hassle of having to manually preprocess the data like what was done for the CNN implementation. Instead, we can just call a function within the feature extractor class to get our input.

4.2.1. Model (Wav2Vec2)

Wav2Vec2 is a SOTA model proposed in wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations by Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, Michael Auli from Facebook AI. The authors proposed a framework for self-supervised learning of representations from raw audio data. The method encodes speech audio via a multi-layer convolutional neural network followed by masks spans of the resulting latent speech representations like masked language modeling. These masked latent representations are fed to a Transformer network to build contextualized representations. The model is trained via a contrastive task where the true latent is to be distinguished from distractors. Discrete speech units are learnt during training via a gumbel softmax to represent the latent representations in the contrastive task. After pre-training on unlabeled speech, the model is fine-tuned using labeled data with a Connectionist Temporal Classification (CTC) loss to be used for downstream speech recognition tasks. The figure below shows the framework of Wav2Vec2.

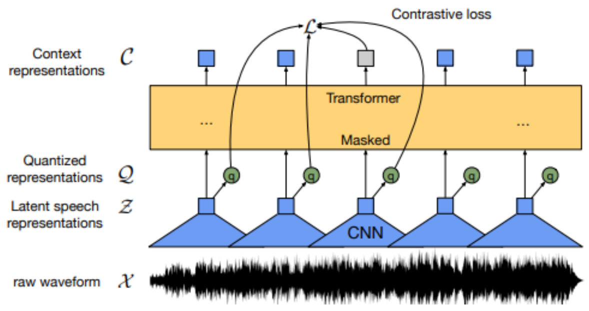


Figure 1. Framework of Wav2Vec2

4.2.3. Training

This can be done in a few steps using the existing transformers library:

1. Split our data into train validation test splits as usual
2. Extract the features need for the model using the AutoFeatureExtractor class:

```
from transformers import AutoFeatureExtractor
feature_extractor = AutoFeatureExtractor.from_pretrained("facebook/wav2vec2-large-xlsr-
53")
```

```
def preprocess_function(examples):
    audio_arrays = [x["array"] for x in examples["audio"]]
    inputs = feature_extractor(
        audio_arrays, sampling_rate=feature_extractor.sampling_rate
    )
    return inputs
```

3. Replace or add classification layers at the top of the model to match the number of classes in our dataset. This can be done using the AutoModelForClassification class:

```
from transformers import AutoModelForAudioClassification, TrainingArguments, Trainer
num_labels = len(label2id)
model = AutoModelForAudioClassification.from_pretrained(
    "facebook/wav2vec2-large-xlsr-53", num_labels=num_labels, label2id=label2id,
id2label=label2id
)
```

4. Instantiate the trainer class with hyperparameters of our choosing to prepare the model for training:

```
training_args = TrainingArguments(
    output_dir="english_new_emotion_model",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    learning_rate=0.0001,
    per_device_train_batch_size=4,
    gradient_accumulation_steps=2,
    per_device_eval_batch_size=4,
    num_train_epochs=25,
    warmup_ratio=0.1,
    logging_steps=10,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    push_to_hub=False,
    report_to="wandb"
)


trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=encoded_english_dataset["train"].with_format("torch"),
    eval_dataset=encoded_english_dataset["val"].with_format("torch"),
    tokenizer=feature_extractor,
    compute_metrics=compute_metrics,
)


trainer.train()
```

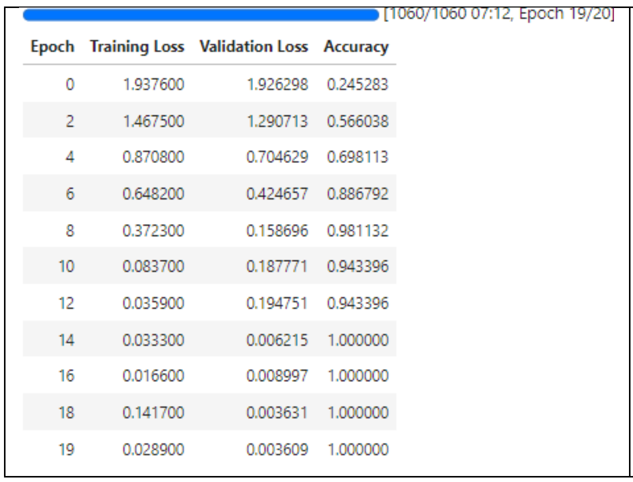
5. Evaluate the model as needed:

```
trainer.evaluate(encoded_english_dataset["test"].with_format("torch"))
```

5. Experiment and Results

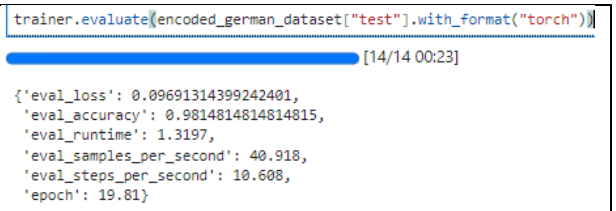
We have employed the methods described to train the datasets individually and in combination using the transfer learning model, to observe their respective effects.

Training on EmoDB dataset only (german model):



[1060/1060 07:12, Epoch 19/20]

Epoch	Training Loss	Validation Loss	Accuracy
0	1.937600	1.926298	0.245283
2	1.467500	1.290713	0.566038
4	0.870800	0.704629	0.698113
6	0.648200	0.424657	0.886792
8	0.372300	0.158696	0.981132
10	0.083700	0.187771	0.943396
12	0.035900	0.194751	0.943396
14	0.033300	0.006215	1.000000
16	0.016600	0.008997	1.000000
18	0.141700	0.003631	1.000000
19	0.028900	0.003609	1.000000

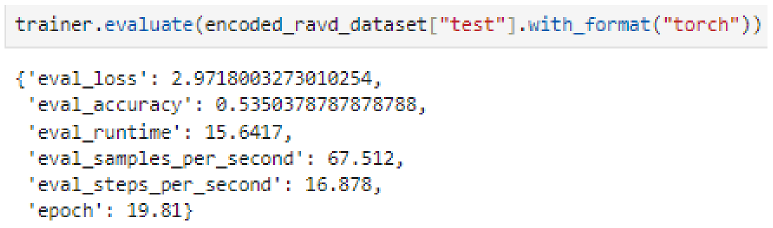


```
trainer.evaluate(encoded_german_dataset["test"].with_format("torch"))
```

[14/14 00:23]

```
{'eval_loss': 0.09691314399242401,
 'eval_accuracy': 0.9814814814814815,
 'eval_runtime': 1.3197,
 'eval_samples_per_second': 40.918,
 'eval_steps_per_second': 10.608,
 'epoch': 19.81}
```

The validation accuracy hit 100% with very small validation loss. The test accuracy on EmoDB test split is also very high at 98.1%. We will investigate the model's performance on RAVDESS dataset for data with common labels to check its robustness.

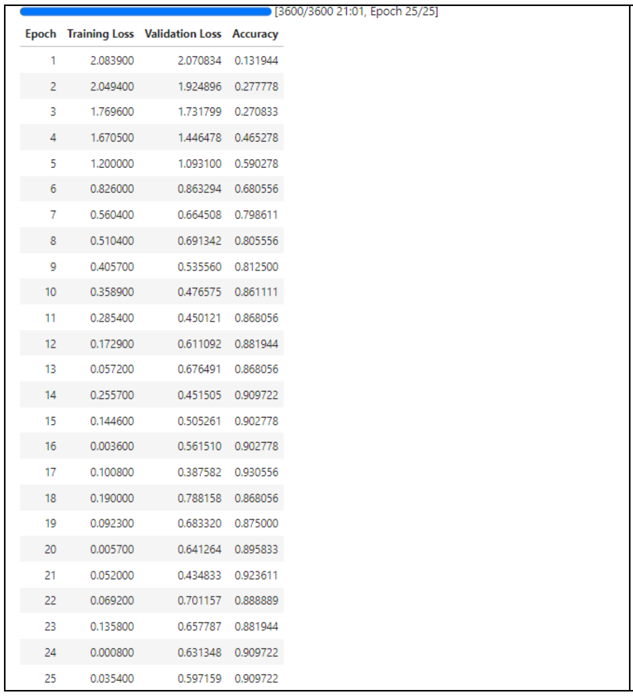


```
trainer.evaluate(encoded_ravd_dataset["test"].with_format("torch"))
```

```
{'eval_loss': 2.9718003273010254,
 'eval_accuracy': 0.5350378787878788,
 'eval_runtime': 15.6417,
 'eval_samples_per_second': 67.512,
 'eval_steps_per_second': 16.878,
 'epoch': 19.81}
```

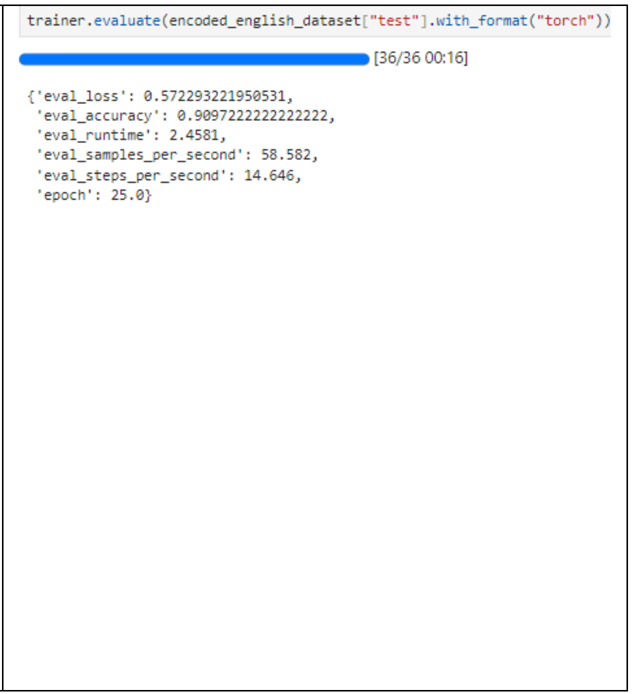
The test accuracy is low at 53.5%. The german model does not perform well when tested on the english RAVDESS dataset, this could be due to differences in each language's expression of emotions and overfitting.

Training on RAVDESS dataset only (english model)



[3600/3600 21:01, Epoch 25/25]

Epoch	Training Loss	Validation Loss	Accuracy
1	2.083900	2.070834	0.131944
2	2.049400	1.924896	0.277778
3	1.769600	1.731799	0.270833
4	1.670500	1.446478	0.465278
5	1.200000	1.093100	0.590278
6	0.826000	0.863294	0.680556
7	0.560400	0.664508	0.798611
8	0.510400	0.691342	0.805556
9	0.405700	0.535560	0.812500
10	0.358900	0.476575	0.861111
11	0.285400	0.450121	0.868056
12	0.172900	0.611092	0.881944
13	0.057200	0.676491	0.868056
14	0.255700	0.451505	0.909722
15	0.144600	0.505261	0.902778
16	0.003600	0.561510	0.902778
17	0.100800	0.387582	0.930556
18	0.190000	0.788158	0.868056
19	0.092300	0.683320	0.875000
20	0.005700	0.641264	0.895833
21	0.052000	0.434833	0.923611
22	0.069200	0.701157	0.888889
23	0.135800	0.657787	0.881944
24	0.000800	0.631348	0.909722
25	0.035400	0.597159	0.909722

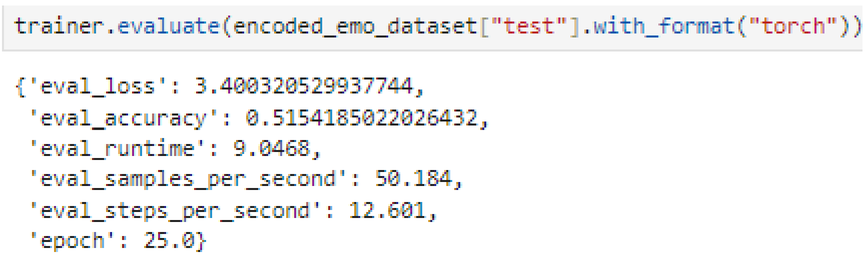


```
trainer.evaluate(encoded_english_dataset["test"].with_format("torch"))
```

[36/36 00:16]

```
{'eval_loss': 0.572293221950531,
 'eval_accuracy': 0.9097222222222222,
 'eval_runtime': 2.4581,
 'eval_samples_per_second': 58.582,
 'eval_steps_per_second': 14.646,
 'epoch': 25.0}
```

The validation accuracy hit 93% with low validation loss. The test accuracy on RAVDESS test split is also very high at 90.9%. We will investigate the model’s performance on EmoDB dataset for data with common labels to check its robustness.

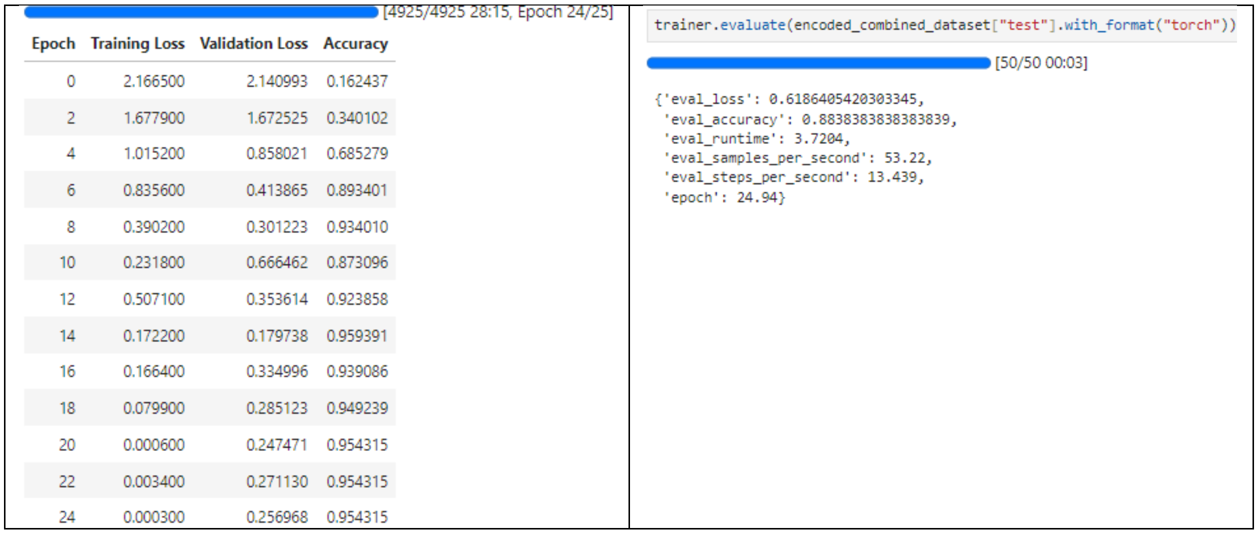


```
trainer.evaluate(encoded_emo_dataset["test"].with_format("torch"))
```

```
{'eval_loss': 3.400320529937744,
 'eval_accuracy': 0.5154185022026432,
 'eval_runtime': 9.0468,
 'eval_samples_per_second': 50.184,
 'eval_steps_per_second': 12.601,
 'epoch': 25.0}
```

The test accuracy is low at 51.5%. The english model does not perform well when tested on the German EmoDB dataset, this could be due to differences in each language's expression of emotions and overfitting.

Training on Combined dataset (EmoDB + RAVDESS)


```
[4925/4925 28:15, Epoch 24/25]
```

Epoch	Training Loss	Validation Loss	Accuracy
0	2.166500	2.140993	0.162437
2	1.677900	1.672525	0.340102
4	1.015200	0.858021	0.685279
6	0.835600	0.413865	0.893401
8	0.390200	0.301223	0.934010
10	0.231800	0.666462	0.873096
12	0.507100	0.353614	0.923858
14	0.172200	0.179738	0.959391
16	0.166400	0.334996	0.939086
18	0.079900	0.285123	0.949239
20	0.000600	0.247471	0.954315
22	0.003400	0.271130	0.954315
24	0.000300	0.256968	0.954315

```
trainer.evaluate(encoded_combined_dataset["test"].with_format("torch"))
```

```
[50/50 00:03]
{'eval_loss': 0.6186405420303345,
 'eval_accuracy': 0.8838383838383839,
 'eval_runtime': 3.7204,
 'eval_samples_per_second': 53.22,
 'eval_steps_per_second': 13.439,
 'epoch': 24.94}
```

The validation accuracy hit 95.4% with very small validation loss. The test accuracy is high at 88.4%. The combined model that was trained on combined dataset performed well for emotion detection.

6. Detecting emotions dynamically:

Dynamic emotion detection in speech involves analyzing emotional content within specific subintervals of a speech utterance rather than just assessing the overall emotional tone. To achieve this, we divide the .wav file into 3-second segments with the correct sampling rate (16KHz) and analyze each segment individually using a saved model. This process continues until the entire file is processed.

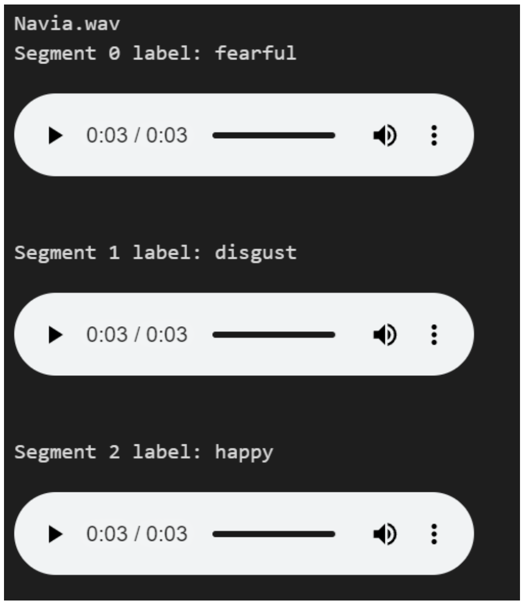
We will be using our best performing model from our results to carry out this task.

```
def dynamic_recognition(input_wav_path, model_checkpoint):
    model = AutoModelForAudioClassification.from_pretrained(model_checkpoint)
    id2label = model.config.id2label
    output_paths = split_wav(input_wav_path)

    for i, path in enumerate(output_paths):
        # Preprocess
        input = preprocess_function(path, model_checkpoint)
        input_values = torch.tensor(input["input_values"])
        attention_mask = torch.tensor(input["attention_mask"])

        # Predict
        with torch.no_grad():
            results = model(input_values, attention_mask=attention_mask)
            probabilities = F.softmax(results.logits, dim=1)
            # Get the index of the maximum value
        argmax_index = torch.argmax(probabilities)
        print(f'Segment {i} label:', id2label[argmax_index.item()])
        display_audio(path)
        print()
```

Our test on a Genshin Impact voice line showed positive results, with each segment clearly showing the emotion our model predicted and matching the actual emotion displayed, albeit having no official emotion label for that segment. Below is an example of what the output looks like:



```
Navia.wav
Segment 0 label: fearful
```

```
Segment 1 label: disgust
```

```
Segment 2 label: happy
```

Furthermore, given that our inference speed is much faster than the length of the segmented audio clip, we can safely deduce that our model can perform emotion detection dynamically.

7. Conclusion:

To train a proper CNN model, we realised that there was a lot of additional feature engineering that had to be done to process audio files into usable inputs. This is not the case for transfer learning, given that many platforms that provide transfer learning models (such as Hugging Face) already have classes and functions to pre-process audio files into inputs. Furthermore, the recognition rates for our CNN model were roughly 61.1%, which pales in comparison to the Wav2Vec2 transformer model provided by Hugging Face which had recognition rates of 88.4% on the combined dataset. A reason for this could be because transformers are specifically designed to model long-range dependencies in sequential data, unlike CNNs which are proficient at capturing local patterns in grid-like data such as images. This difference makes transformers more well-suited for tasks where temporal ordering of information is crucial, such as speech emotion recognition. To leverage on the strengths of both CNN and transformers, a hybrid model that combines both architectures could potentially achieve higher accuracy in SER. Recent research [4] has explored the advantages of integrating CNN with transformers resulting in a more comprehensive representation of the input data. It is without a doubt that further experimentation and research is needed to optimize such architectures, but the potential for an improved performance is promising.

References

[1] S. R. Livingstone, “Ravdess emotional speech audio Dataset,” Kaggle, https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio (accessed Apr. 11, 2024).

[2] P. Agnihotri, “EmoDB dataset,” Kaggle, https://www.kaggle.com/datasets/piyushagni5/berlin-database-of-emotional-speech-emodb (accessed Apr. 11, 2024).

[3] Mandal, M. (2024, February 23). *Introduction to convolutional neural networks (CNN)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-

[4] Tang, X., Lin, Y., Dang, T., Zhang, Y., & Cheng, J. (2024). Title of the report [Speech emotion recognition via CNN-transformer and multidimensional attention mechanism]. Retrieved from URL https://arxiv.org/html/2403.04743v1#bib.bib8