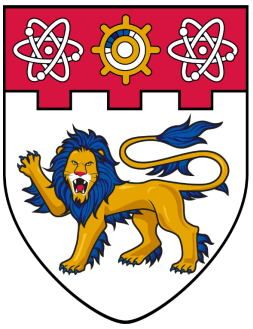

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

EmotionRNN - Integrative Multimodal Learning for Speech Emotion Recognition

GROUP PROJECT

AY 2023/24 SEMESTER-1
CZ4042 - Neural Networks and Deep Learning

Delivered by—
Pathak Siddhant (U2023715K)
Anand Arjun Pramod (U2023574B)
Mahtolia Ronan (U2023144J)

Submitted on —
November 10, 2023

Table of Contents

1. Abstract

In this project, we tackle the challenging task of accurately detecting emotions in conversational data involving multiple speakers. Leveraging the intricate interplay of emotions within dialogues, we employ a multi-modal approach, combining audio and text embeddings. We utilise pre-trained models to extract audio representations from independent utterances and implement an attentive bi-directional GRU to capture contextual dependencies and speaker-specific interactions dynamically. Our experiments, conducted on the MELD conversational dataset, demonstrate the effectiveness of our method in speech emotion recognition.

2. Introduction

2.1 Speech Emotion Recognition

Speech Emotion Recognition (SER) is vital for dialogue generation, social media analysis, and human-computer interaction, particularly in the evolving field of Emotion Recognition in Conversation (ERC) that requires nuanced context modelling. Capturing emotional subtleties in speech signals poses a challenge. Communication often involves navigating ambiguity, like interpreting phrases like “What a wonderful surprise” which can convey varied emotions. Despite the surge of deep learning in SER, striking a balance between its power and leveraging domain knowledge for interpretable models is crucial. A holistic approach, integrating deep learning with an understanding of context and modalities, remains essential.

An example of the dataset (MELD) with SER is given below:

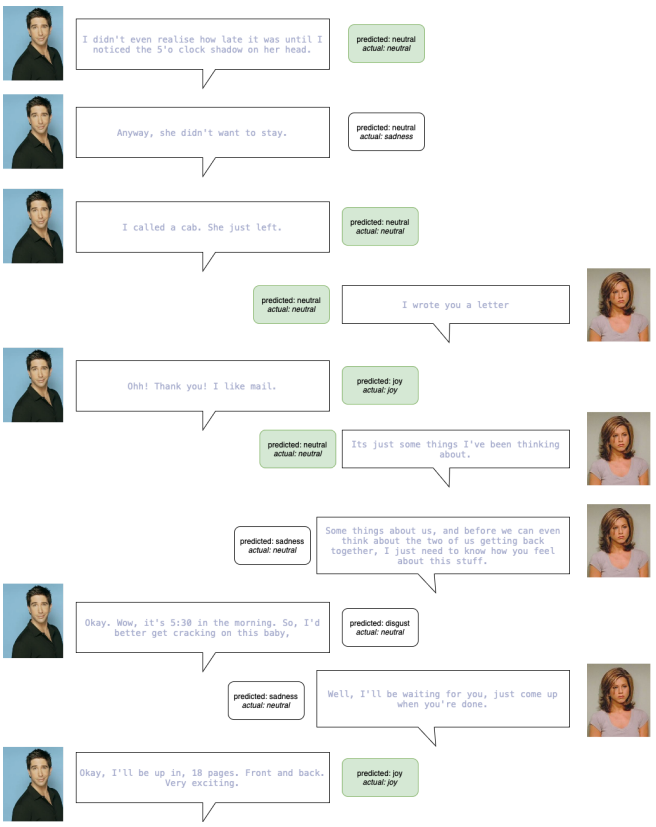

Figure 1. Sample conversation from MELD (Train split)

2.2 Multimodal Learning

In the realm of Artificial Intelligence, the quest to emulate human cognition has led to the exploration of multimodal machine learning. Just as humans seamlessly integrate information from their five senses, the future of AI lies in processing diverse data modalities. SER becomes a prime example, where understanding emotions requires decoding cues from both spoken words and textual content. Multimodal Deep Learning, a cutting-edge subfield, focuses on training AI models to navigate and establish connections across different modalities, such as audio and text. This approach enables a more comprehensive comprehension of the emotional landscape, where nuances captured in voice tones and textual expression collectively contribute to a deeper understanding, surpassing the limitations of individual modalities. Previous works such as [1]–[3] have showcased the tremendous power in the hands of two or more modes of information.

2.3 Problem Statement

This refinement involves integrating sophisticated algorithms that can accurately discern nuanced emotional cues within bi-directional dialogue, thereby ensuring that the emotional context of each conversational exchange is more accurately captured and interpreted. By doing so, it enables a more precise and contextually aware analysis of emotions, leading to a significant improvement in the accuracy and relevance of emotion recognition systems in complex conversational environments.

3. Methodology

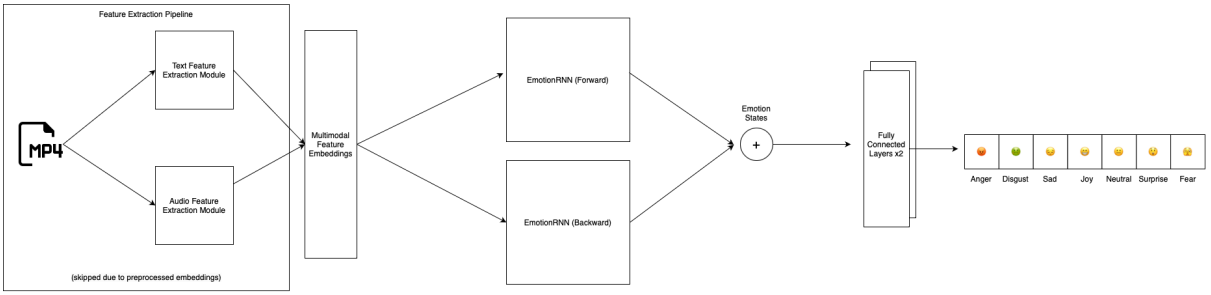

Figure 2. Model Pipeline

Our model pipeline consists of three major modules: 1) embedding vector generation and 2) context-aware module that aims to model the interactions in the dialogue and 3) classification head that makes the emotional state prediction.

3.1 Embedding Vector Generation

Feature extraction process to generate embeddings was borrowed from [4]. Two different approaches were taken to segment extract features from text and audio components from the utterance clip of a dialogue. Due to lack of computational resources, we borrowed the pre-processed embeddings generated by [5], available as a part of open-source code available at https://github.com/declare-lab/conv-emotion.

3.1.1 Textual Features

Convolutional neural networks (CNN) are employed for the extraction of textual features. N-gram features are obtained from each utterance by using three distinct convolution filters of sizes 3, 4, and 5. The outputs from these filters are subjected to max-pooling and then processed through ReLU. Following this, the activations are concatenated and fed into a dense layer with 100 dimensions, which serves as the representation of the textual utterance.

3.1.2 Audio Features

As suggested by the official project handout, OpenSMILE toolkit [6] supported and maintained by Audeering for acoustic feature extraction.

3.2 Model Architecture

Based on the acoustic and textual embeddings generated in sections 3.1, they are concatenated together to form the final input to the model architecture. These embeddings are represented as X_t at timestamp t. The model architecture is composed of four submodules consisting of Gated Recurrent Unit (GRU) cells, which are utilised to capture the global context, intra- and inter-speaker states, as well as the influence and emotion state of the participant. These GRU cells are designed to effectively manage the flow of information, allowing the network to retain or forget data through its update and reset gates. This capability enables the nuanced modelling of temporal sequences and dependencies within the data.

3.2.1 Global Context

In conversational emotion recognition, the emotional state of an utterance at timestamp t is determined by considering the preceding utterances as its cumulative context. The global contextual state G_t is based on updated on the basis of previous utterances' self-speaker state S_{t-1}, intra-speaker states I_{t-1}, global state G_{t-1} and the embedding vector X_t of the current utterance at timestamp t. Formally, the update can be calculated using the formula below:

$$G_t = GRU_G(G_{t-1}, (S_{t-1} \oplus I_{t-1} \oplus X_t))$$

Here $\oplus$ implies concatenation. At time t = 0, random initialization is used for states.

In order to amplify the contribution of the context-rich information, we employ matching attention from the history interactive context to combine long-context speaker interaction influences and conversational dependence. We pool the attention vector a_t from the surrounding context history using soft-attention.

$$\alpha_i = \frac{exp(u_i^T)}{\sum_{i=1}^{t-1} exp(u_i^T)} \text{ where } u_i = tanh(WG_i + b)$$

$$a_t = \sum_{i=1}^{t-1} \alpha_i G_i$$

3.2.2 Self Speaker and Listener States

Self-influence modules consists of two GRU cells: GRU_{SS} and GRU_{SL}. S denotes the speaker and L denotes the listener. The aim is to memorize the emotional inertia of speaker/listener, which represents the emotional dependency of the person with their own previous states. Emotional inertia refers to the situations where speakers may not always express explicitly their feelings or outlook through reactions. It is updated on the basis of global state G_t and the embedding vector X_t of the current utterance at timestamp t.

$$S_{SS,\,t} = GRU_{SS}(S_{SS,\,t-1}, (G_{S,t} \oplus X_{S,t}))$$
$$S_{SL,\,t} = GRU_{SL}(S_{SL,\,t-1}, (G_{L,t} \oplus X_{L,t}))$$

3.2.3 *Intra Speaker and Listener States*

The intra-speaker and listener states are passively observed, felt, and understood by other participants in a natural setting. Typically, this state encompasses expressions, reactions, and responses. Similar to 3.2.2 in terms of design and use of two GRU cells, it is updated on the basis of intra speaker state I_{t-1}, attentive contextual vector a_t, the global vector G_t of the current utterance at timestamp t.

$$I_{SS,\,t} = GRU_{SS}(I_{SS,\,t-1}, (G_{S,t} \oplus a_{S,t}))$$
$$I_{SL,\,t} = GRU_{SL}(I_{SL,\,t-1}, (G_{L,t} \oplus a_{L,t}))$$

3.2.4 *Emotion State*

The emotional mood of the speaker and the emotion class of the utterance are determined by the emotional state. It is posited that the emotional state is dependent upon the utterance and the composite state of the speaker, which encompasses the internal, external, and intent states. The current emotion state is also naturally dependent on the speaker's previous emotion state. It is updated on the basis of self-speaker state S_t, the global vector G_t and the current utterance representation X_t.

$$E_t = GRU_E(E_{t-1}, (G_t \oplus X_t \oplus S_t))$$

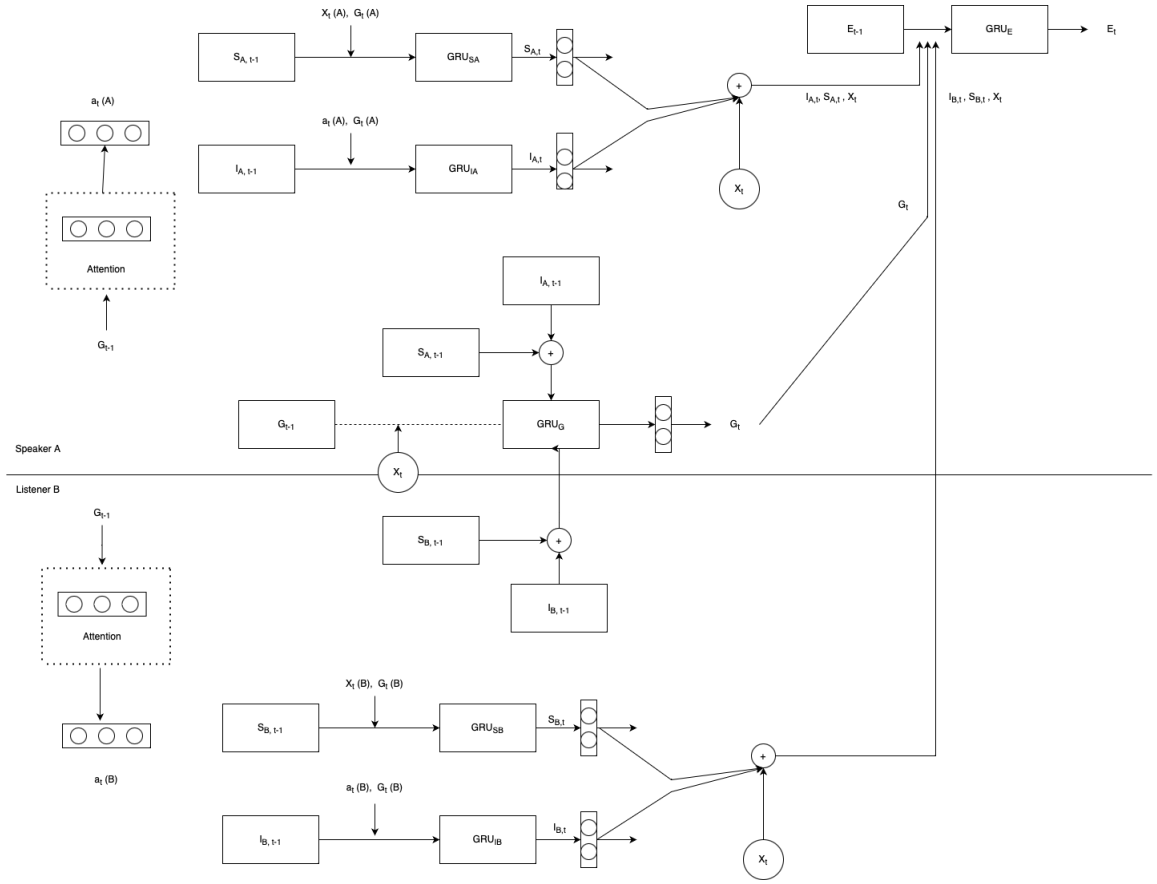

Figure 3. Model Architecture

3.3 Classification Head

The final output emotion state is then calculated in a bidirectional manner, which is analogous to bidirectional RNNs where two different RNNs are used for backward and forward passes of the input sequence. The outputs from both are concatenated. This helps the model in learning from both past and future utterances in the dialogue through the two models, thus providing better context for emotion classification. This approach was adopted from [2]. The final output emotion state is fed into two fully-connected linear layers with a log-softmax function to get log-probabilities as the final output out of the model.

3.4 Attention Mechanisms

In the realm of neural network architectures, attention mechanisms play a crucial role in enhancing the model's ability to process and interpret data efficiently. Two notable types are Simple Attention and Matching Attention. Simple Attention is foundational, focusing on weighting input features to highlight relevant information, ideal for capturing long-distance dependencies within a dataset. It uses a combination of linear transformations and non-linear activations to assign importance to different parts of the input. In contrast, Matching Attention is more specialized, designed for scenarios where two distinct sets of data need to be compared or aligned, such as in aligning a query with relevant information in a text corpus. It comes in various forms, including dot-product, general, and concatenation-based mechanisms, each tailored to different dimensions and types of data alignment. The approach has been borrowed from [7]

3.4.1 Simple Attention

The Simple Attention mechanism is essential in neural networks for modeling dependencies within input data, irrespective of distance. It calculates a weighted sum of input features, where weights are dynamically derived from the input, indicating the importance of each feature. Typically, the process involves a linear layer followed by a tanh activation and a softmax function to generate normalized weights. This results in a context vector that encapsulates the most relevant information from the input sequence. Predominantly used in various domains like natural language processing and image processing, its ability to capture long-range dependencies makes it a popular choice in many applications.

```
class SimpleAttention(nn.Module):

    def __init__(self, input_dim):
        super(SimpleAttention, self).__init__()
        self.input_dim = input_dim
        self.scalar = nn.Linear(self.input_dim,1,bias=False)

    def forward(self, C, x=None):
        scale = F.tanh(self.scalar(C))
        alpha = F.softmax(scale, dim=0).permute(1,2,0)
        attn_pool = torch.bmm(alpha, C.transpose(0,1))[:,0,:]
        return attn_pool, alpha
```

Figure 4. Code Snippet of Simple Attention Module

3.4.2 Matching Attention

The Matching Attention mechanism is more complex, designed for scenarios that involve the comparison of two distinct sets of features, often termed as "memory" and "candidate" elements. This mechanism is particularly useful in tasks like question-answering, where it's necessary to evaluate the relevance between different information sources. It includes variations like dot-product, general, and concatenation-based attentions. Dot-product attention computes direct similarity for features of the same dimension, while general attention allows different dimensions through a linear transformation. Concatenation-based attention merges features before processing for weight determination.

```python
class MatchingAttention(nn.Module):

    def __init__(self, mem_dim, cand_dim, alpha_dim=None, att_type='general'):
        super(MatchingAttention, self).__init__()
        self.mem_dim = mem_dim
        self.cand_dim = cand_dim
        self.att_type = att_type
        self.transform = nn.Linear(cand_dim, mem_dim, bias=False)

    def forward(self, M, x, mask=None):
        if type(mask)==type(None):
            mask = torch.ones(M.size(1), M.size(0)).type(M.type())
            M_ = M.permute(1,2,0) # batch, mem_dim, seqlen
            x_ = self.transform(x).unsqueeze(1) # batch, 1, mem_dim
            alpha = F.softmax(torch.bmm(x_, M_), dim=2) # batch, 1, seqlen

        attn_pool = torch.bmm(alpha, M.transpose(0,1))[:,0,:] # batch, mem_dim
        return attn_pool, alpha
```

Figure 5. Code Snippet of Matching Attention Module

3.5 Model Variants

We have two attention mechanisms and two different types of embeddings (unimodal and multimodal), there are six different combinations of model types:

Variant	Embedding Type	Attention Mechanism
AudioSimple	Audio	Simple Attention
AudioMatching	Audio	Matching Attention
TextSimple	Text	Simple Attention
TextMatching	Text	Matching Attention
AudioTextSimple	Audio + Text	Simple Attention
AudioTextMatching	Audio + Text	Matching Attention

Table 1. Model Variants based on modality and attention mechanisms

4. Experiments

Dataset: The Multi-modal EmotionLines Dataset (MELD) was utilised, which is a multi-modal and multi-speaker conversational dataset. MELD includes acoustic, textual, and visual information pertaining to utterances and conversations from the TV series “Friends”. It encompasses seven emotion categories: anger, disgust, sadness, joy, neutral, surprise, and fear. The dataset consists of 13,798 utterances within 1,433 conversations. The dataset has been divided into three subsets for the purposes of machine learning: training, validation, and testing, with 9,946 utterances allocated for training, 1,241 for validation, and 2,610 for testing. On average, there are about 9-10 utterances in one conversation, most of them being dyadic in nature.

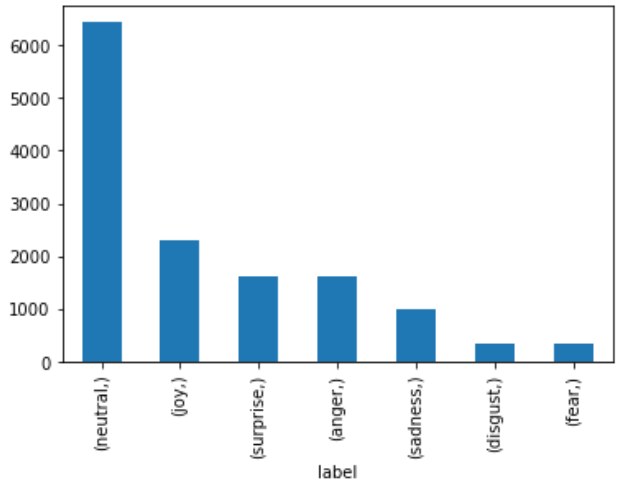

Figure 7. Dataset Label Distribution

Evaluation Metrics: Due to the natural imbalance across various emotions as shown in Figure 7, we chose a weighted average F1 measure as the evaluation metric.

Parameter Settings: The following parameter settings were used to train the network. Some of the settings, such as the loss function were implemented manually such as the loss function due to lack of implementation in native PyTorch 2.0 setup.The learning rate is set to 1e-4,. A batch size of 30 is chosen, balancing the need for computational efficiency and the granularity of the gradient updates. An L2 regularization weight of 3e-4 is applied, which helps prevent overfitting by penalizing large weights. A dropout rate of 0.2 indicates that 20% of the nodes are randomly ignored during training, introducing redundancy in the network to improve generalization. The model is trained for 50 epochs. The loss function used is Masked Negative Log-Likelihood (NLL) Loss, which is to be handling sequences with variable lengths, only considering the prediction error of active parts of the output. Lastly, Stochastic Gradient Descent (SGD) is used as the optimizer.

5. Results and Discussion

5.1 Results

Model	**MELD (Train)**		**MELD (Test)**	
	Accuracy	*Weighted-F1 score*	*Accuracy*	*Weighted-F1 score*
AudioSimple	0.2301	0.2547	0.2498	0.2703
AudioMatching	0.2384	0.2695	0.2789	0.3015
TextSimple	0.5309	0.5638	0.5207	0.564
TextMatching	0.5422	0.5702	0.5241	0.5273
AudioTextSimple	0.5103	0.5488	0.5195	0.5164
AudioTextMatching	0.5780	0.5880	0.5337	0.5420

Table 2. Performance evaluation against test and train split

The Table (shown below) shows the performance of recommendation by the methods described in Section 2. The highest measures are highlighted in red and the second-highest measures are underlined for reference.

5.2 Discussion

The results in Table 2 suggest that multimodal models, which combine audio and text data, significantly outperform unimodal models that rely on a single type of data. The AudioTextMatching model achieves the best results, indicating that the fusion of audio and textual information provides a more comprehensive context for the machine learning task at hand, possibly because it captures a wider range of features that can be leveraged for better performance. The superiority of the TextMatching model over AudioSimple and AudioMatching models could be due to the richer and potentially more nuanced information contained in text, such as syntax and semantics, which are not present in audio data. Audio data alone might not capture the full extent of the emotional or contextual cues that text can provide. Conversely, the relatively lower performance of the audio-based models suggests that audio features alone may not be as discriminative or may be more challenging to model effectively for the task compared to text. The limited performance could also be attributed to factors like background noise, variations in speech, or the complexity of processing and extracting meaningful features from raw audio.

The provided graphs in Figure 7 depict the evolution of the model's performance over 50 epochs, reflecting an optimistic trend of learning and improvement. The decreasing training loss signifies the model's successful learning from the training data. In tandem, the F-score and accuracy graphs demonstrate an encouraging increase for both training and validation sets, with the training curve leading. The convergence of validation and test curves, while slightly lagging behind the training curves, suggests that the model generalizes well to new data.

Figure 7. Model Training Graphs (L->R: Loss, F1-score and Accuracy)

The slight plateauing of validation and test metrics indicates a good fit, although there may be room for enhancement in generalization, possibly through methods such as regularization or more diverse training data, to achieve even higher performance on unseen data.

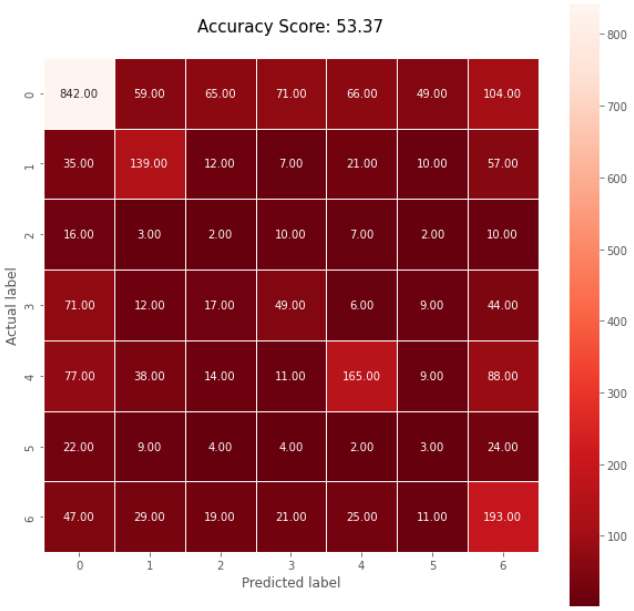

Figure 8. Confusion Matrix

The confusion matrix in Figure 8 displays a promising classification performance with an accuracy score of 53.75%. The substantial numbers on the diagonal indicate that the model is effective in identifying a good portion of true positives, especially in classes 0 (neutral), 1 (surprise), and 6 (anger). However, there is a visible confusion between some classes, such as between classes 0 and 6 (neutral vs anger), and 4 and 6 (joy vs anger), where a relatively high number of instances were incorrectly classified. The misclassifications present an opportunity for model optimization, which could be achieved by fine-tuning or additional training. The overall pattern in the matrix suggests that the model has a solid foundation for accurate predictions, with potential for enhancements to achieve even greater precision. Further analysis on why our model prediction is better in some cases while worse-off for some pairs could shed light on contextual dependency (say, sarcasm) and speaker-sensitive influence, utilising works for emotion cause recognition tasks.

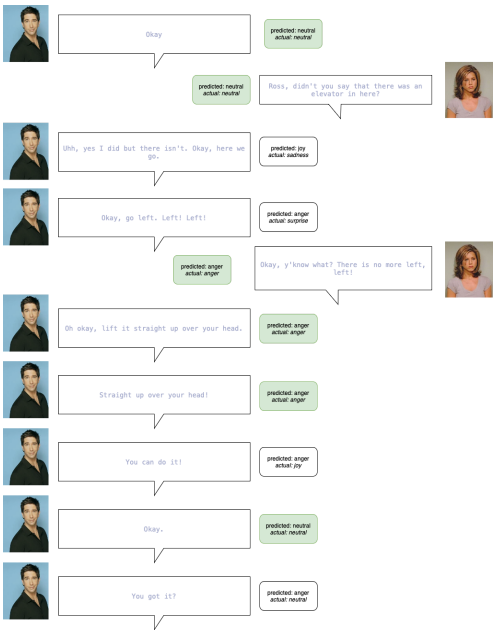

Figure 10. Model performance against test split

Shown here, in Figure 9, is a snippet from the test split of the MELD dataset to evaluate and visualise the model's performance. The model performs and generalises well to unseen data. As highlighted above about misclassificatiom, such instances are available below as well.

6. Conclusion

The project suggested a new way to use small parts of speech spectrograms to figure out the emotions being expressed in conversation pieces. This method focuses on understanding the context, including how people influence themselves and each other while speaking. Two attention mechanisms were explored and the matching attention mechanism was used to pick out the most important parts of the conversation from past exchanges. A bi-directional GRU, which is a kind of model that looks at information in two ways, was used to understand how these influences work together. The tests done showed that this new method works well.

7. References

[1] V. Chudasama, P. Kar, A. Gudmalwar, N. Shah, P. Wasnik, and N. Onoe, “M2FNet: Multi-modal Fusion Network for Emotion Recognition in Conversation.” arXiv, Jun. 05, 2022. Accessed: Oct. 26, 2023. [Online]. Available: http://arxiv.org/abs/2206.02187

[2] D. Hazarika, S. Poria, A. Zadeh, E. Cambria, L.-P. Morency, and R. Zimmermann, “Conversational Memory Network for Emotion Recognition in Dyadic Dialogue Videos,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2122–2132. doi: 10.18653/v1/N18-1193.

[3] J. Luo, H. Phan, and J. Reiss, “deep learning of segment-level feature representation for speech emotion recognition in conversations.” arXiv, Feb. 05, 2023. Accessed: Oct. 27, 2023. [Online]. Available: http://arxiv.org/abs/2302.02419

[4] D. Hazarika, S. Poria, R. Zimmermann, and R. Mihalcea, “Conversational Transfer Learning for Emotion Recognition.” arXiv, May 19, 2020. Accessed: Oct. 26, 2023. [Online]. Available: http://arxiv.org/abs/1910.04980

[5] N. Majumder, S. Poria, D. Hazarika, R. Mihalcea, A. Gelbukh, and E. Cambria, “DialogueRNN: An Attentive RNN for Emotion Detection in Conversations.” arXiv, May 25, 2019. Accessed: Oct. 26, 2023. [Online]. Available: http://arxiv.org/abs/1811.00405

[6] F. Eyben, M. Wöllmer, and B. Schuller, “Opensmile: the munich versatile and fast open-source audio feature extractor,” in *Proceedings of the 18th ACM international conference on Multimedia*, Firenze Italy: ACM, Oct. 2010, pp. 1459–1462. doi: 10.1145/1873951.1874246.

[7] D. Ghosal, N. Majumder, A. Gelbukh, R. Mihalcea, and S. Poria, “COSMIC: COmmonSense knowledge for eMotion Identification in Conversations.” arXiv, Oct. 06, 2020. Accessed: Oct. 29, 2023. [Online]. Available: http://arxiv.org/abs/2010.02795

8. Appendix

A: Inference Pipeline

We used OpenSMILE toolkit, MoviePy and PyTorch and the pre-processing suggested by [5]. This helps us to make a deployable, and scalable application as well. Any open-source software such as Gradio or HuggingFace can be used to test the performance as well.

```
def opensmile_feature(audio_file_path, feature_type):
    if feature_type == "emobase":
        smile = opensmile.Smile(
            feature_set=opensmile.FeatureSet.emobase,
            feature_level=opensmile.FeatureLevel.Functionals,
        )
    elif feature_type == "ComParE":
        smile = opensmile.Smile(
            feature_set=opensmile.FeatureSet.ComParE_2016,
            feature_level=opensmile.FeatureLevel.Functionals,
        )
    else:
        smile = opensmile.Smile(
            feature_set=opensmile.FeatureSet.eGeMAPSv02,
            feature_level=opensmile.FeatureLevel.Functionals,
        )
    return np.array(smile.process_file(audio_file_path))

def get_audio_from_mp4(video_filename):
    filename, _ = os.path.splitext(video_filename)
    clip = VideoFileClip(video_filename)
    clip.audio.write_audiofile(f"{filename}.wav", verbose=False, logger=None)
    return f"{filename}.wav"

def get_audio_emb_from_one_mp4(video_filename):
    audio_filename = get_audio_from_mp4(video_filename)
    audio_array = opensmile_feature(audio_filename, 'ComParE')
    return get_emb_from_audio(audio_array)

def make_embs_for_dialogue(dialogue_dir):
    video_filenames = [f for f in os.listdir(dialogue_dir) if os.path.isfile(os.path.join(dialogue_dir, f))]
    video_filenames.sort()
    video_filenames = [os.path.join(dialogue_dir, f) for f in video_filenames]
    video_filenames = [f for f in video_filenames if f.endswith('.mp4')]

    audio_embs = np.array([get_audio_emb_from_one_mp4(video_filename) for video_filename in video_filenames])
    return audio_embs
```