Speech Evaluation With Multi modal

Junseo Kim
Department of Statistics,
SungKyunKwan University
hazzisss5249@gmail.com

MinSeok Seo
Department of Software,
SungKyunKwan University
minsukdamin@gmail.com

JungHwan Lee
Department of Statistics,
SungKyunKwan University
ljh640370@gmail.com

MinSeok Choi
Department of Software,
SungKyunKwan University
snowdolf99@gmail.com

ChaeYoon Han
Department of Library and Information
Science, SungKyunKwan University
chaeyoon@gmail.com

ABSTRACT
In today's world, the significance of effective speech is increasingly recognized. However, some individuals experience presentation jitters or anxiety when speaking in public. To address this issue, we developed a model that classifies speeches as either good or bad. While existing models have focused on analyzing text or audio separately, we took a novel approach by incorporating multiple modalities simultaneously.
To train our model, we utilized the TEDLIUM dataset, which contains a diverse range of TED speeches, and the MOSI dataset, which includes speeches of poor quality. By combining these two datasets, we performed binary classification using a multi-modal approach. The baseline model, trained solely on speech-to-text data, achieved an accuracy of 0.82. In contrast, our model, leveraging multiple modalities, demonstrated an accuracy that consistently approached 1, indicating significant improvements in classification performance.

Keywords
Speech; assessment; TEDLIUM; MOSIl; BERT; Cosine Similarity; Next Senctence Probability; Speech to text; openSMILE; jitterLocal; OpenCV; L2CS NET; Cubic Spline interpolation, MaxScaling; Multi Modal; Feature Vector Concatenation; LSTM;

1. INTRODUCTION
We learn public speaking and evaluate our presentations at various stages of life. Despite learning public speaking from a young age, many people still feel nervous and anxious about it. This phenomenon is known as "presentation jitters" or "stage fright." To address this, we believe it is beneficial to practice self-assessment by delivering multiple presentations and providing feedback to ourselves. Consequently, we are aiming to develop an AI that can directly evaluate our own presentations, rather than relying on others for feedback.

2. prior research
2.1 JobKorea's AI Interview Evaluation Service
This is a service that evaluates the fluency of responses and the likelihood of passing based on conducting interviews with given questions. It has the limitation of evaluating only within the context of the given question, as it is an evaluation service targeting interviews rather than presentations. Additionally, due to the nature of the interview situation, it only captures the upper half of the body in the video recording, making it difficult to evaluate overall body movements or gestures, which makes it challenging to apply in general situations.

2.2 Ringle's AI Analysis for Video English Lessons
This service analyzes video English lessons conducted with actual people, measuring factors such as the amount of speech and frequently used words. It incurs high costs as it requires the involvement of human instructors, making complete AI substitution unfeasible. It does not consider the context of responses and relies on evaluating based on simple speech volume, making it difficult to assess the fluency of presentations accurately.

2.3 Speak's AI OPIC Preparation Service
This service is designed to prepare for the English speaking test, OPIC, by engaging in conversations with an AI chatbot that corrects grammar and expression errors. It lacks a comprehensive diagnostic system beyond error correction. Furthermore, it does not utilize video information, so it does not consider posture or appropriateness of gestures in its evaluation.

3. Dataset
3.1 Good Presentation: TEDLIUM
TEDLIUM is a dataset consisting of TED (Technology, Entertainment, Design) lecture videos. It is composed of high-quality English pronunciations and provides accompanying transcripts, making it easily accessible for text analysis. Additionally, the lecture videos are freely available, allowing for video processing as well. Our Good Presentation Dataset length is 89.

3.2 Poor Presentation: MOSI
MOSI (Multimodal Corpus of Sentiment Intensity) provides a dataset of flawed presentations that includes various modalities such as emotions in videos, emotions in speech, and emotions in text. This dataset encompasses a range of errors and flawed presentations that occur when speakers make mistakes or are emotionally unstable. By using this dataset, models can be made more robust and powerful when deployed in real-world scenarios. Our Poor Presentation Dataset length is 92.

4. Data PreProcessing

4.1 Text Dataset

4.1.1.1 Change file extension

We will convert the audio file in “sph” format to wav format and then upload it to the Google Cloud bucket. The reason for uploading it to the Google Cloud bucket is that when using the *Speech-to-Text API*, there is a time limit of 1 minute for local audio files. By utilizing the bucket, we can use the audio file without the limitation of its length.

We extracted the text from *TEDLIUM's* subtitles, which are in the “stm” file format, and removed any text that did not correspond to the script, such as applause sounds or static.

4.1.1.2 Google Cloud APIs: Speechtotext API

The *Speech-to-text API* is an API that provides the functionality to convert speech into text. By using this API, it is possible to take in speech input and return the result in text format. This allows for the analysis, processing, and storage of speech data in text form.

By comparing the obtained text from the *Speech-to-text API* with the script, it is possible to determine how closely the speaker followed the script during the presentation.

4.1.1.3 Extra

A problem was discovered in the Speech-to-text data where sentence delimiters were missing. To address this issue, a period ('.') was added for sentence segmentation.

As a result, a CSV file was created for multimodal analysis, containing the audio file names, script, *speech-to-text* data, and labels, using the text data.

4.2 Audio Dataset

The audio data in TEDLIUM is in the “sph” file format. The *openSMILE* API does not support “sph” files. To address this, we used the “sph” file Python package to convert the “sph” files from TEDLIUM to the wav format. This conversion allows the audio files to be compatible with *openSMILE* for further processing and analysis.

4.3 Vision Dataset

The input of the *L2CS net* in this project is sequential data, specifically videos. However, the environment for this project is *Google Colab*, and there is concern about decreased model training speed due to limited computing power. To overcome this issue, I decided to extract frames at 1-second intervals from the videos and use them as input for the *L2CS net*. I used the *OpenCV* Python package to extract frames from the dataset's videos at the desired intervals.

4.4 Data Scaling

The gaze model utilizes the output of the *L2CS net* and converts it to Euclidean distance, resulting in values ranging from -inf to inf. On the other hand, the outputs of the other modalities range from 0 to 1. Therefore, I applied *MinMax* scaling only to the output of the *L2CS net*. The reason for choosing *MinMax* scaling among various scaling techniques is that it ensures the scaled values fall within the range of 0 to 1, matching the range of other feature vectors.

5. UNIModel

5.1 Text Model

5.1.1 Cosine Similarity

To calculate similarity between a script and text obtained through a *Speech-to-Text API*, the following steps were performed:

First, the pretrained BERT model was used to tokenize the script and the text. Second, each token was converted into an integer. Third, to match the vector sizes, the sentences were split into the first 512 tokens and the last 512 tokens, and these segments were saved. Fourth, the embedding of each sentence was generated to summarize it into a single vector. The sentence embeddings were computed by transforming the sentences into embedding vectors and calculating the average of these vectors. Fifth, the cosine similarity between the preprocessed sentence embedding vectors was calculated. Finally, the calculated cosine similarity vectors were concatenated, resulting in a tensor of size 768*2 (1536), which was then saved.

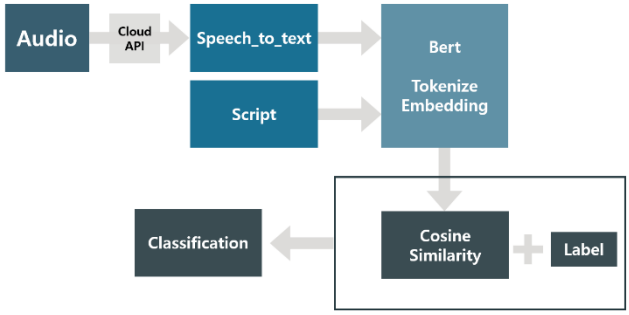


Figure 1. Flow Chart of Cosine Similarity

5.1.2 BertTopic

Using Pre-trained *BERT* model with script data, we calculated topic vector. Next, calculate cosine similarity of each sentence and the topic vector. We evaluate about probability about the script contains useless contents. The output of this model is one-dimensional vector with the value of elements 0 to 1.

Expression 2. Cosine Similarity

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

5.1.3 Next Sentence Probability

When we perceive language, we can observe that even if two sentences imply similar topics, their meaning and comprehension can vary depending on the order. Therefore, the order of utterances is crucial, and in speech acts such as presentations, the sequence becomes even more important in conveying information. Through *BerTopic*, we can determine whether the presentation sentences are similar to the topic, regardless of the order of the presentation sentences. Thus, we wanted to add metrics related to the order of presentation sentences.

BERT is pretrained on a large corpus of text data, and Next Sentence Prediction (NSP) enhances the model's ability to understand the relationship between two sentences. NSP refers to the task of predicting whether two given sentences are actually consecutive.

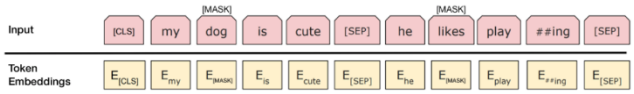


Figure 2. Example of BERT Input

In this context, [*CLS*] represents the token indicating the start of a sentence, while [*SEP*] represents the token indicating the boundary between sentences. In the NSP task, when two given sentences are actually consecutive, they are labeled as "IsNext," and if they are not consecutive, they are labeled as "NotNext." During this process, the *BERT* model generates the next sentence probability, which is the probability of whether each sentence follows the other. We conducted fine-tuning on the *RoBERTa* pretrained model using our dataset. From this model, we extracted the NSP representation and created a 1-dimensional vector of shape (number of sentences, 1).

5.2 Audio Model

We intended to extract speech feature vectors using the *openSMILE* API, which takes speech as input and provides various features extracted from the input speech. During our team meeting, we explored various features and concluded that using the *jitterLocal* feature, which measures the stability of speech, would be reasonable. We decided to utilize the *jitterLocal* feature as a modality for analyzing speech.

jitterLocal_sma3nz represents a feature that indicates the frequency variation ratio related to the intensity fluctuation of the speech. It estimates the stability of speech by measuring the frequency variation. The *jitterLocal* feature is composed of a one-dimensional vector. Each element's value ranges from 0 to 1. The values are derived at a resolution of 0.01 seconds, so if the audio is 3 minutes long, a vector of length 18,000 will be obtained.

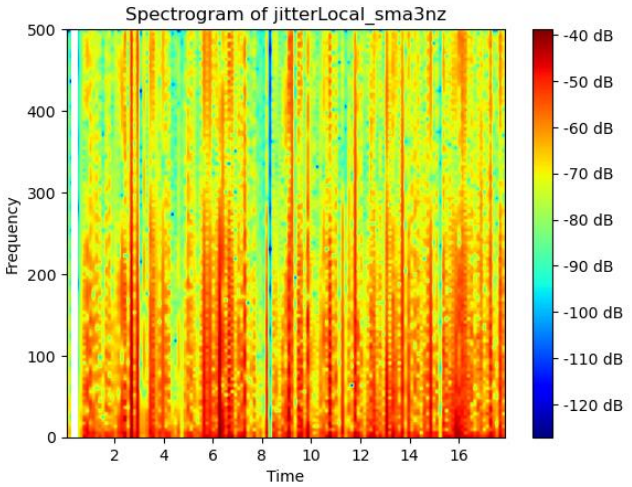


Figure 3. Spectrogram of JitterLocal Feature

5.3 Vision Model

5.3.1 L2CS-NET

L2CS-Net utilizes a multi-loss approach for estimating 3D gaze angles from RGB images. It consists of two fully-connected layers and defines separate loss functions to regress the individual gaze angles (yaw and pitch). By performing gaze binary classification using a *softmax* layer and *cross-entropy* loss, the network robustly estimates the gaze angles.

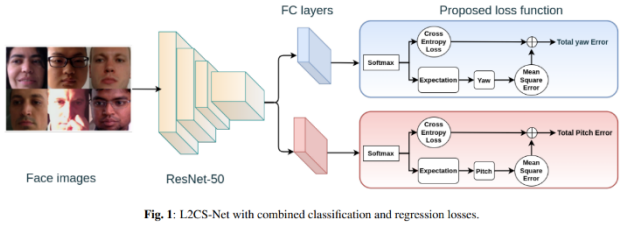


Fig. 1: L2CS-Net with combined classification and regression losses.

Figure 4. Flow Chart of L2CS-Net

To make *L2CS-Net* capable of handling sequentially extracted image values at 1-second intervals using the *OpenCV* module, modifications have been made. While *L2CS-Net* typically expects continuous data in the form of videos, the sequential input of image values allows for measurements as well.

L2CS-Net takes facial images as input and utilizes *ResNet-50* as its backbone to extract spatial gaze features. It employs two fully-connected layers with shared convolution layers from the backbone to individually predict the gaze angles. The same two loss functions, one for yaw and one for pitch, are applied.

The outputs of the FC layers are transformed into probability distributions using *softmax* layers to obtain output logits. The bin classification loss is computed by employing the cross-entropy loss between the probabilities and the target bin labels. Additionally, the mean squared error (MSE) for the predictions is calculated and added to the classification loss.

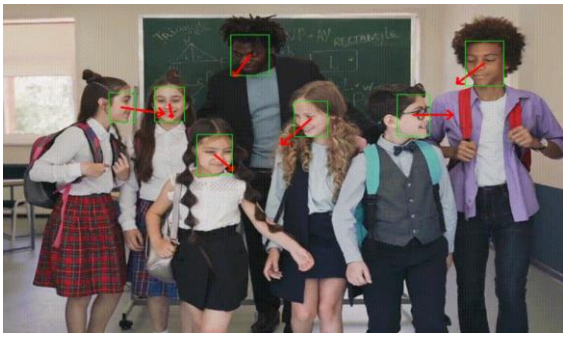


Figure 5. Result of L2CS-Net

5.3.2 *Euclidean Distance*

To measure how much the gaze direction is unstable by using the extent of gaze movement, the L2CS-Net was run, and the resulting pitch_predicted_list (horizontal) and yaw_predicted_list (vertical) vectors were utilized to calculate the Euclidean distance.

Expression 2. Euclidean Distance

$$distance = (x2-x1)2 + (y2-y1)2$$

5.3.3 *Cubic Spline interpolation*

Expression 3. Linear Spline Model

$$p_{1,i}(x) = f_{i-1} + (\frac{f_i - f_{i-1}}{x_i - x_{i-1}})(x - x_{i-1}), i = 1, ...n$$

When approximating an arbitrary function y = f(x), if it is approximated with linear segments at regular intervals, the graph at each intersection of the approximation intervals will appear as a piecewise linear shape. This can result in a large approximation error, and when considering the overall polynomial, it becomes a function that is not continuously differentiable.

Mathematically, it can be represented as described above, and the method of connecting with straight lines as mentioned above is called a linear spline model.

Expression 3. Cubic Spline Model

$$s_{3,i}(x) = f_{i-1} + a_i(x - x_{i-1}) + b_i(x - x_{i-1})^2 + c_i(x - x_{i-1})^3$$

Increasing the degree of the polynomial is one approach to address this issue. Even considering a quadratic function can compensate for the line's sharpness and create a smoother function at each point. By using a cubic function, as shown in the equation above, we can further improve the smoothness of the interpolation by compensating for the sharpness of the lines. As the degree of the polynomial increases and becomes differentiable up to the nth order, the piecewise connection becomes smoother. However, if the degree is too high, oscillation issues may arise. To avoid oscillation problems, we used the most commonly used method, which is the Cubic Spline Model.

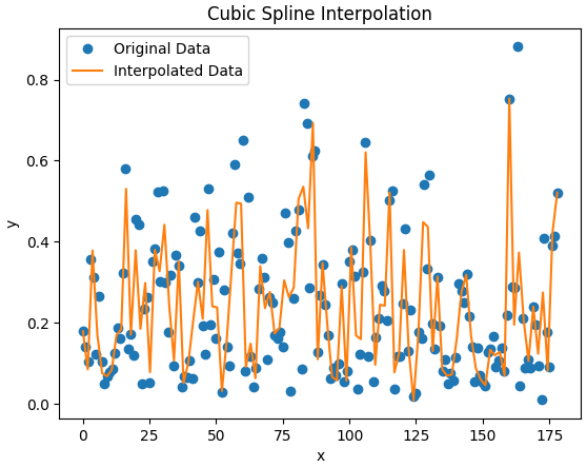


Figure 6. Cubic Spline Interpolation

6. MultiModel

Based on *"Automatic Driver Stress Level Classification Using Multimodal Deep Learning" by Mohammad Naim Rastgoo*, we conducted multimodal fusion by incorporating additional novelties to adapt the model to our final dataset vector.

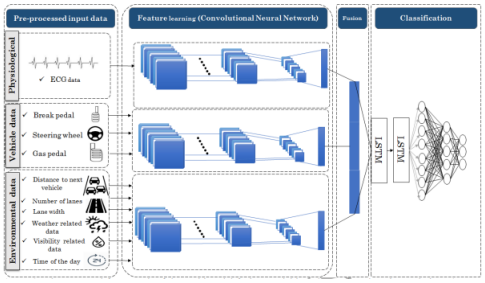


Figure 7. Flow Chart of Multimodal Fusion

6.1 Feature vector Max Scaling

We applied weighting based on the length of each feature vector. Let's call it *Max Scaling*. First, for a given data instance, we calculate the total length of the feature vectors obtained from each unimodal source. Then, we divide the length of each feature vector by the total length. Since there are 5 features, a total of 5 values are obtained. Among these 5 values, we divide all values by the largest value. The resulting values are multiplied by each corresponding feature vector.

The smallest value is multiplied by the feature vector with the longest length, while 1 is multiplied by the feature vector with the shortest length. During *LSTM* training, we observed that longer feature vectors had a significant impact on model learning. By multiplying a smaller value to longer feature vectors, we aimed to equalize the influence of each feature.

For example, Assuming the lengths of each feature are as follows:

Table 1. Each Feature Info

Kind	length	extra
Bert Topic Feature Vector	88	Number of sentences in the script
Cosine Similarity	1536	Fixed
Next Sentence Probability	88	Number of sentences in the script
L2CS net	1270	Video length in seconds
jitterLocal feature	119530	Video length in seconds * 100
Sum	122512	The sum of the lengths of all features

The sum of the lengths of all features is 122512

When we divide the total length by the length of each feature, the following results are obtained:

Table 2. Length of Each Feature Vector

Kind	length
Bert Topic Feature Vector	1392.18
Cosine Similarity	79.7
Next Sentence Probability	1392.18
L2CS net	96.4
jitterLocal feature	1.02
Maximum value	1392.18

The max of the total length by the length of each feature of features is 1392.18

By performing *Max Scaling* (each feature divided by the maximum value) based on these values, the following values are obtained:

Table 3. Result of Max Scaling

Kind	length
Bert Topic Feature Vector	1
Cosine Similarity	0.057
Next Sentence Probability	1
L2CS net	0.069
jitterLocal feature	0.000736

We then multiply these values by the corresponding feature. *BerT opic* and NSP, which have the shortest lengths, are multiplied by 1, so the original values of the feature vectors remain unchanged for training. On the other hand, the jitterLocal vector, which has the longest length, is multiplied by 0.000736, resulting in a smaller value being multiplied with the original feature vector values. We consider this as one of our novelties.

6.2 Feature vector Concatenation

After applying Max Scaling to the vectors, we concatenate them into a 1-dimensional vector. We considered expanding them into a 2-dimensional vector and using concatenation, but since we couldn't find a suitable method, we opted to concatenate them into a 1-dimensional vector and then apply *Max Scaling*.

6.3 Fusion With LSTM

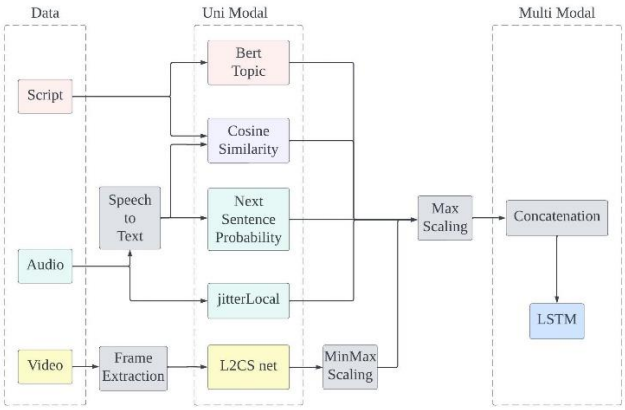


Figure 8. Flow Chart of Overall Model

In summary, we extracted the features from a total of 181 data points, including 89 good data and 92 bad data. After scaling the features, we combined them into a 1-dimensional feature vector. We then fed this feature vector into an *LSTM* model to perform fusion.

6.3.1 First Model: Stacked LSTM

The first model used a *stacked LSTM* with 2 layers, similar to the referenced paper. It had a hidden state size of 16, and a total of 181 data points were used in the dataset. The results are shown in the table below. Due to the limited amount of data, it was determined that the *stacked LSTM* was prone to overfitting. Therefore, instead of using a *stacked LSTM*, a *single-layer LSTM* was employed for fusion.

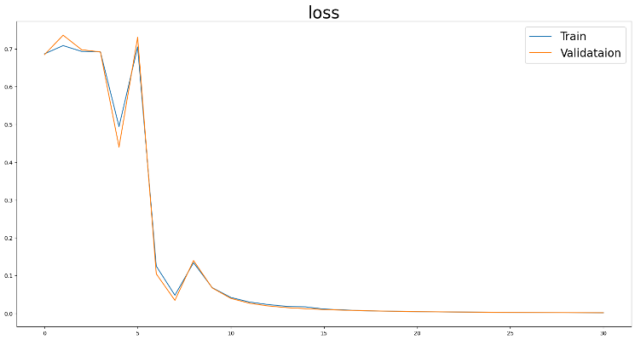


Figure 9. Stacked LSTM Loss

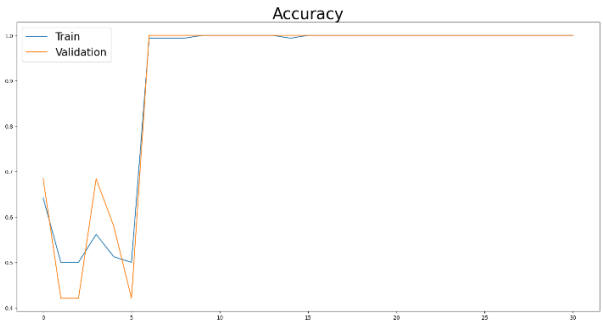


Figure 10. Stacked LSTM Accuracy

6.3.2 Second Model: Only One LSTM

6.3.2.1 Step1: #Hidden State = 16, #Data = 181

The results showed that the model was mostly predicting zeros and had a high occurrence of coincidental matches between the validation values and the predicted values. It was concluded that the model was not accurately predicting the results. To address this issue, the number of hidden states was increased to make the model slightly more complex. The following are the results for the model's loss and accuracy.

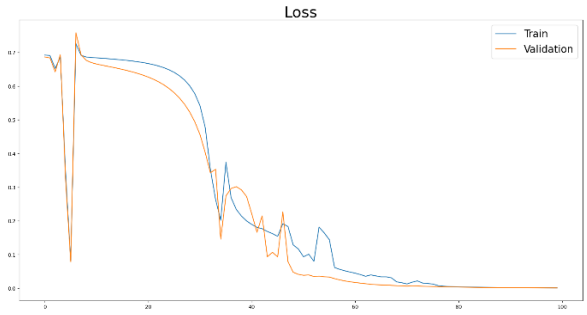


Figure 11. Step1 Loss

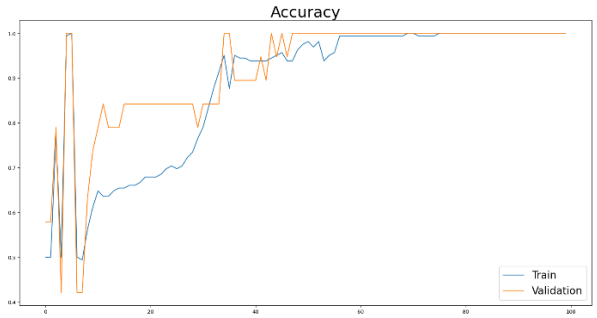


Figure 12. Step1 Accuracy

6.3.2.2 Step2: #Hidden State = 25, #Data = 181

In the previous model, the hidden state was modified to 25. Looking at the loss graph, it can be observed that the validation graph closely followed the train graph. Additionally, there was significant oscillation in the loss. It was identified that this was due to the lack of data in the dataset. Therefore, it was determined that the dataset needed to be increased.

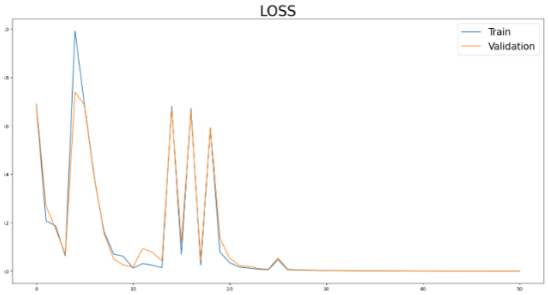


Figure 13. Step2 Loss

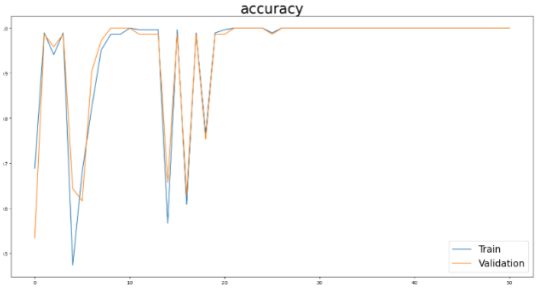
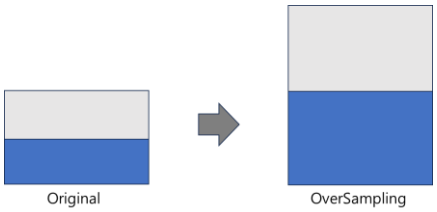


Figure 14. Step2 Accuracy

6.3.2.3 Step3: #Hidden State = 25, #Data = 362

As a result, we performed oversampling on the original data. We doubled the total number of data from 181 to 362, and fed it into the model with a hidden state of 25.



We can clearly observe that the oscillation in the loss graph has significantly decreased compared to the previous one. However, we still see the validation loss following the train loss, indicating the persisting issue of limited data with only 362 samples.

We selected this model to final multimodal model.

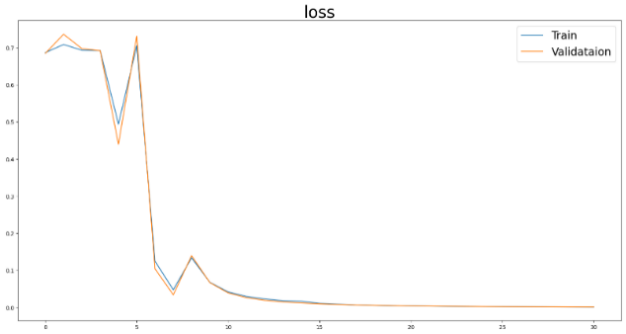


Figure 15. Step3 Loss

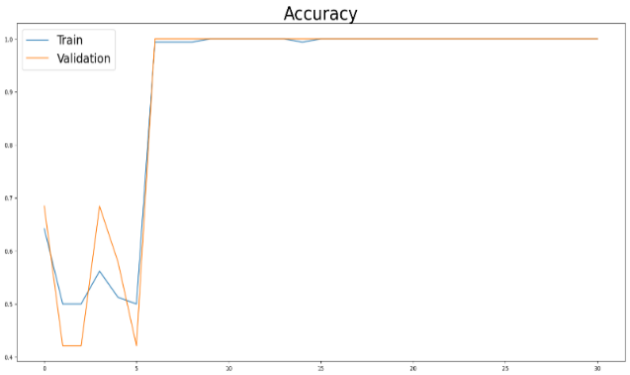


Figure 16. Step3 Accuracy

6.4 Interpretation of the Final Model:

Compared to the previous models, the final model exhibits reduced oscillation in the loss graph. However, it still shows a trend where the validation loss closely follows the training loss. We attribute this behavior to the issue of limited data. Looking at the accuracy, we observe that as the training accuracy increases, the validation accuracy also improves. The accuracy on the test data is also close to 1, which further indicates the impact of data scarcity on the model's performance.

Therefore, we conclude that the challenges of limited data contribute to these observations. The lack of diverse and abundant data hinders the model's ability to generalize well beyond the training set. To address these issues, it is crucial to augment the

dataset with more samples or explore alternative data generation techniques.

Despite these limitations, the final model shows promising results in terms of reduced oscillation and high accuracy. It highlights the importance of dataset expansion and further emphasizes the need for a larger, more diverse dataset to improve the model's performance and enhance its ability to generalize to unseen data.

7. Novelty & Limitations

7.1 Novelty

We have made modifications to the input format of the L2cs net to accommodate video data in a sequential format obtained at a rate of one frame per second. To address the issue of varying lengths between feature vectors, we applied max scaling. Additionally, for the multimodal feature vector concatenation, we replaced the stacked LSTM used in the previous model with a single LSTM.

7.2 Limitations

First, Difficulty in capturing eye movements during significant screen transitions: It is challenging to track eye movements accurately when there are large screen transitions. The model performs best when the screen remains relatively stable, limiting its effectiveness in scenarios with frequent and significant screen changes.

Second, Challenges in measuring gaze for multiple presenters: Measuring gaze and creating feature vectors based on the gaze of multiple presenters is complex. Since the model requires gaze measurements from a single presenter to generate the feature vector, it becomes challenging to handle multiple presenters accurately. Additionally, distinguishing between the presenter and other individuals in the scene is problematic, making it practically necessary to have only one presenter.

Third, Insufficient training data: The limited number of data samples poses a challenge for model training. Although we attempted oversampling by doubling the dataset size to 362 samples, the limitations of having a small dataset persist. To mitigate this issue, it would be beneficial to collect more diverse data, particularly by increasing the number of "bad" presentation samples. Furthermore, the original reference paper used a stacked LSTM, but due to the limited data, we switched to a single-layer LSTM to prevent overfitting.

Fourth, Difficulty in processing video data with smaller frames: Limited computing power and time constraints make it difficult to process video data at smaller frame sizes. Using smaller frames could potentially provide more detailed information about gaze movement trends, but it comes with computational and time limitations.

Fifth, Difficulty in identifying significant features for presentation assessment: It is challenging to determine which features are meaningful for presentation assessment. Since the model utilizes concatenated data feature vectors, it becomes difficult to discern which specific features contribute significantly to the measurement of presentations.

Sixth, Limitations in aligning the lengths of data features: Aligning the lengths of data features proves to be challenging. If the data features had consistent lengths, it would have enabled parallel multimodal learning. However, due to time constraints, we were unable to explore methodologies to align all data feature lengths uniformly.

Lastly, Designed for immediate service but challenging to obtain immediate results: Developing and obtaining results from various models to generate the feature vector requires a considerable amount of time. Additionally, the time investment required for conversion and processing is significant. As a result, obtaining immediate results for assessment is challenging.

8. REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* (Oct.2018)

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin *Attention Is All You Need* (Dec.2017)

[3] Haşim Sak, Andrew Senior, Françoise Beaufays *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition* (Feb.2014)

[4] Mohammad Naim Rastgoo, Bahareh Nakisa, Federic Mire, Andry Rakotonirainy, Vinod Chandran. Automatic driver stress level classification using multimodal deep learning (Dec.2019)

[5] Ahmed A.Abdelrahman, Thorsten Hempel, Aly Khalifa, Ayoub Al-Hamadi *L2CS-Net: Fine-Grained Gaze Estimation in Unconstrained Environments* (Mar.2022)

[6] Florian Eyben, Martin Wollmer, Bjorn Schuller. *Opensmile: the munich versatile and fast open-source audio feature extractor* (Oct.2010)

9. REFERENCES

Git hub : Git hub: https://github.com/ai-project-2023/Speech_grading_multimodal.git