Utilizing Multi-Modal Cues for detection of **Mental Depression**

More Than a Thought

Balancing academics and social life - tips?

Future career options?

Student loans!

I just broke up, what's left in life now!?

Am I doing the right thing?

How do I better manage my time?

Is it all worth it?

What will my family think?

Am I working hard enough?

I don't have any friends!

I just don't have time for anything!

I feel homesick!

I am soo fat, nobody even looks at me!

Peer pressure!

I have no money left!

The problem addressed is detecting depression using **multimodal data**, voice, text and visual features. We aim to develop a model **to detect depression** based on the participant's voice, facial expression and text responses. The focus is on multimodal data and depression detection making it a significant contribution to the field of depression research.

POTENTIAL APPLICATIONS

- Enhances mental depression diagnosis accessibility.
- Serves as a self-diagnostic tool for depression.
- Expandable to other mental health conditions.
- Facilitates early detection and intervention.

POTENTIAL IMPACT

Break the Stigma

- Improves mental health outcomes in under-served areas.
- Assists in depression diagnosis and management.
- Reduces stigma associated with mental health issues.
- Promotes mental well-being in society.

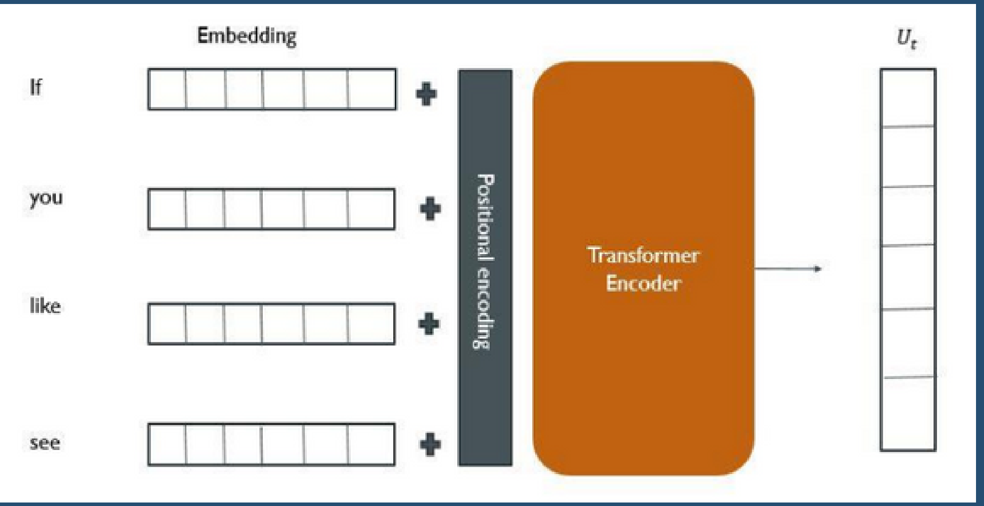
Literature review 1

Depression Detection Model Using Multimodal Deep Learning

Two types of features were extracted:

TEXT

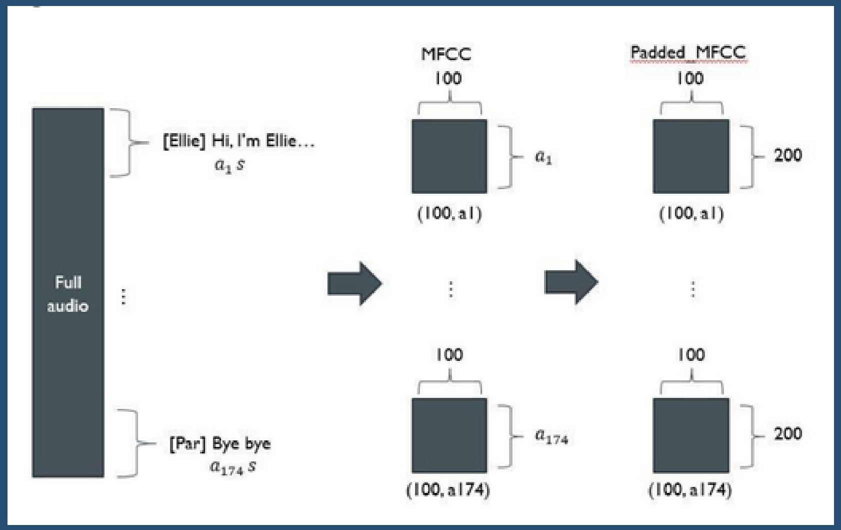
- Script Tokenization and Cleaning (using NLTK's Word Tokenizer)
- BERT Model for Feature Extraction (This involved token embedding, segment embedding, and positional embedding to provide comprehensive word context, ultimately flattening these features into a single vector representation.)



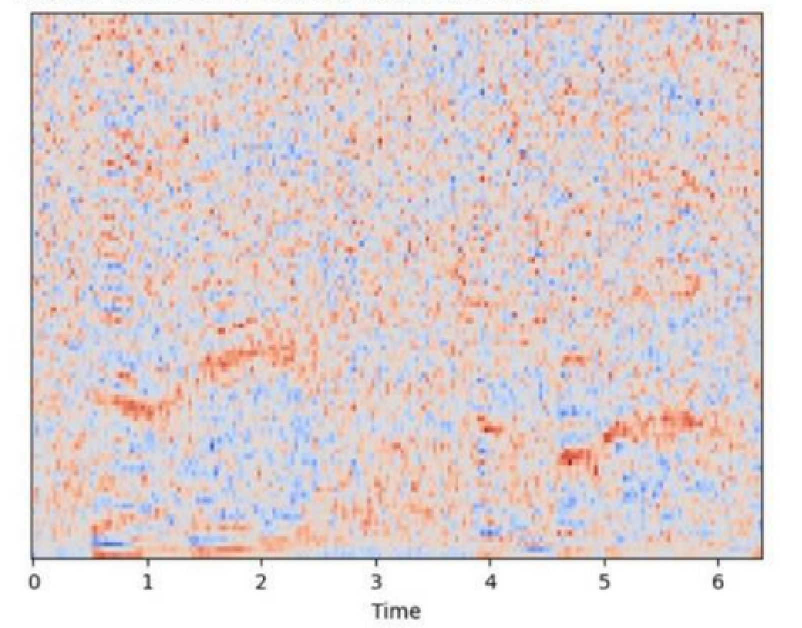
DATASET USED: DAIC-WOZ DATASET

VOICE

- Use of LIBROSA Library (Librosa enables direct handling of .wav files and voice conversion functions.)
- MFCC Feature Extraction



Example of participant 300 audio preprocessing



Model Fusion

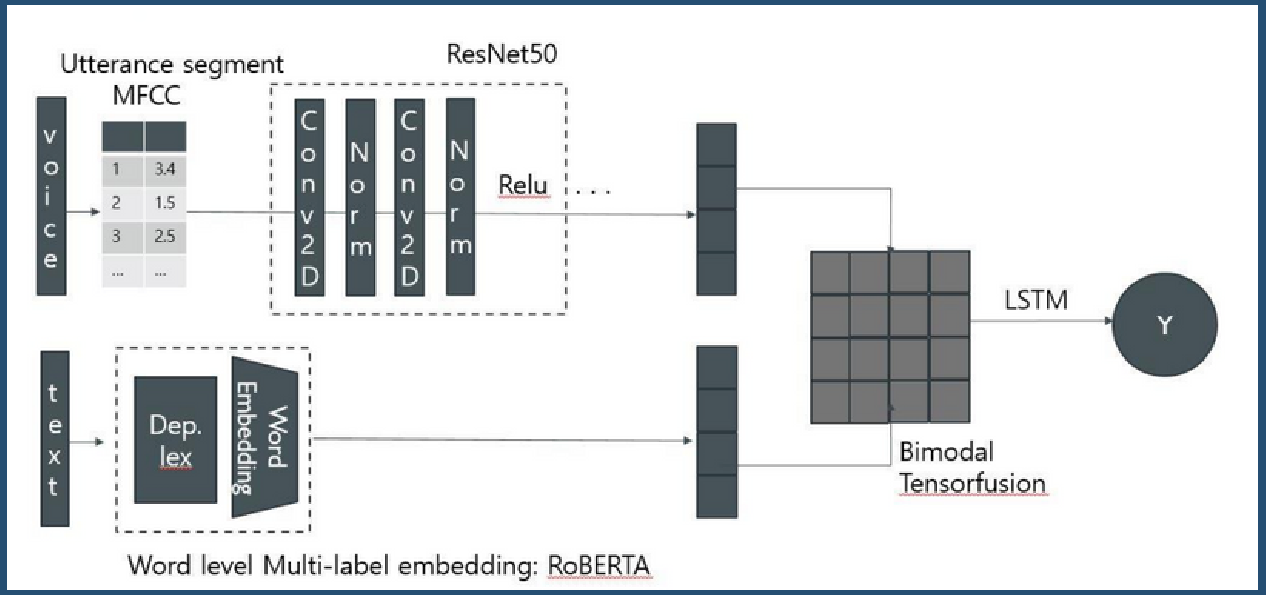
- Tensor Fusion Network (connecting features from each modality (voice and text) into a fully connected layer)

Fusion method	Accuracy
Add	0.6275
Concat	0.6837
Multiply	0.6878
TFN	0.8012

- Depression Detection Using LSTM

Model	Accuracy
SVM	0.5873
RF	0.5692
LR	0.6931
XGB	0.6024
LSTM	0.8012

- Depression Detection Using LSTM
  LSTM, an advanced RNN model, is used due to its ability to consider long-term context, essential in classifying depression based on the entire interview context.



Depression_detection_model_using_multimodal_deep_l%20(1).pdf

Literature review 2

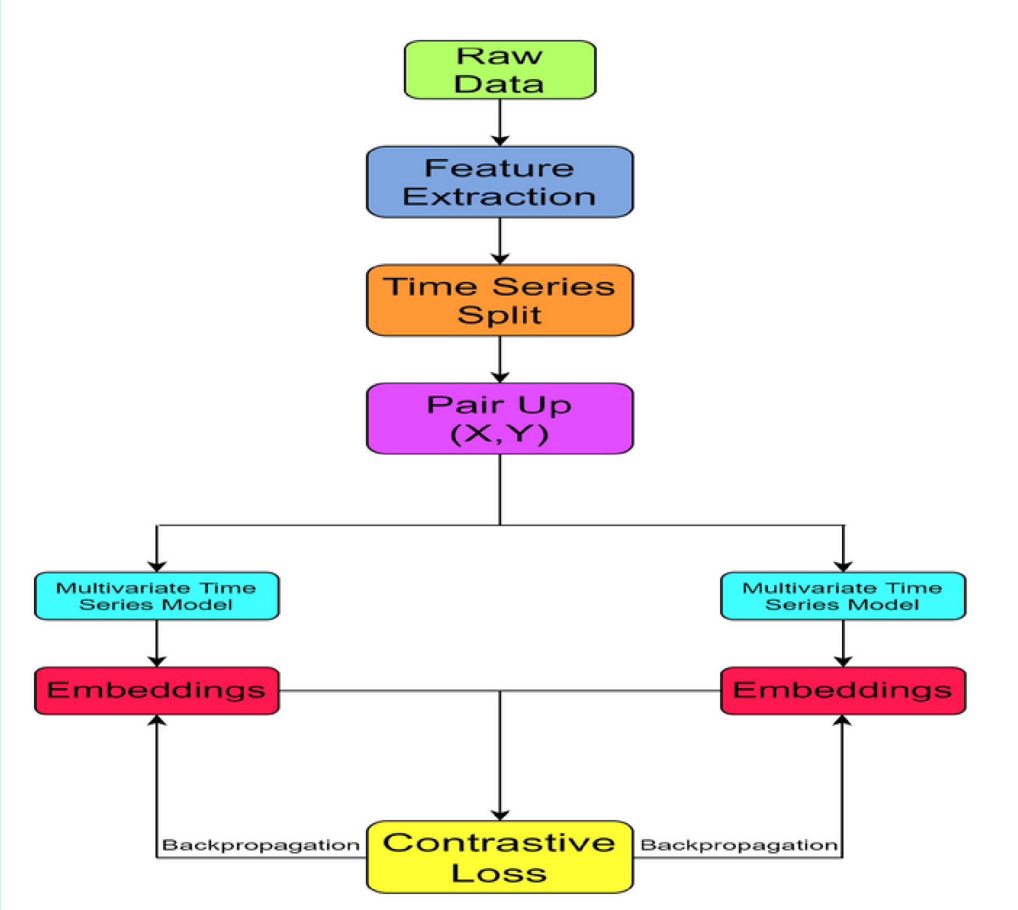
Depression Status Estimation by Deep Learning based Hybrid Multi-Modal Fusion Model

Feature Extraction and Preprocessing:

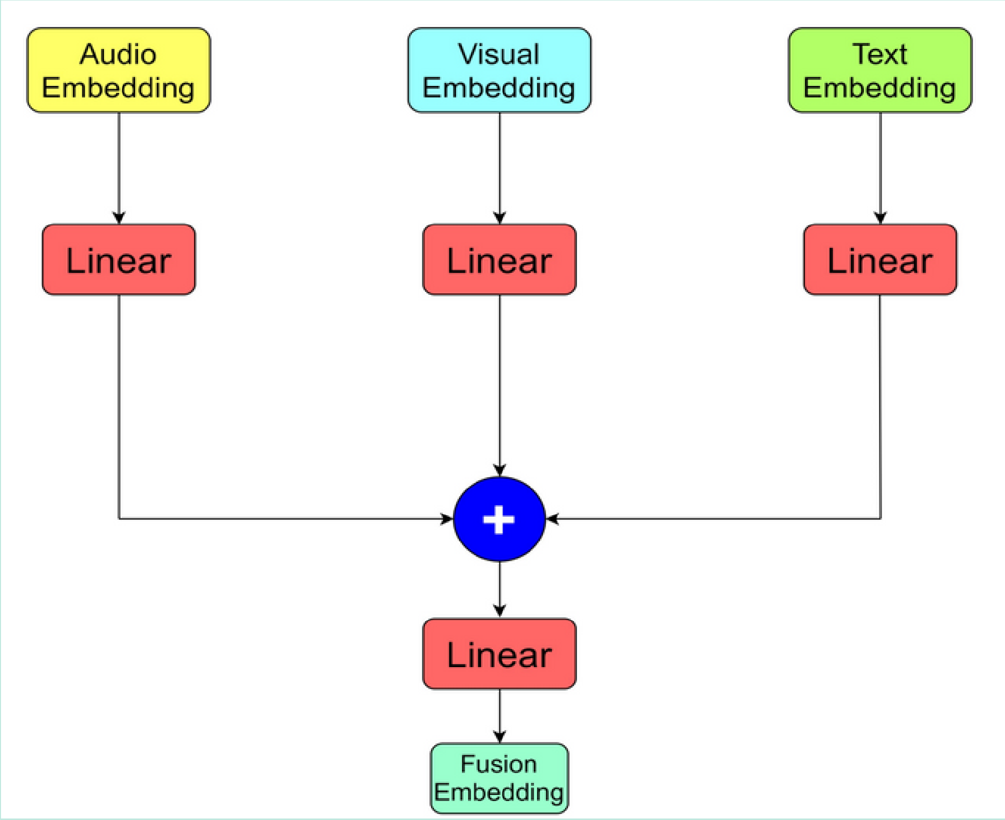
TEXT Preprocessed and analyzed using Sentence-BERT for capturing semantic representations.

VISUAL Facial features extracted using OpenFace, focusing on head pose, facial action units, and gaze.

AUDIO Robust noise filtering with CycleGAN ANF, feature extraction via COVAREP.



Siamese Training Procedure for representation enforcement



Late fusion of multi-modal input

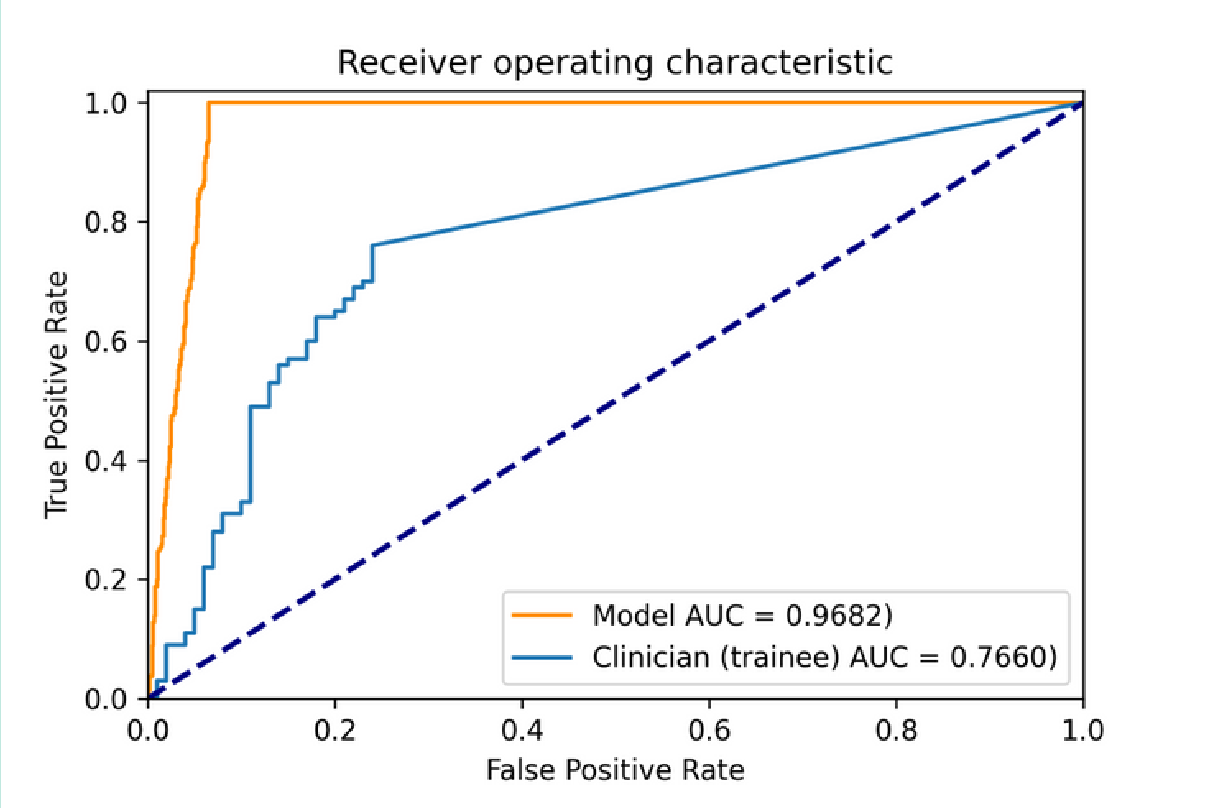
Model Architecture

- Employed a multivariate time-series analysis with attention mechanisms.
- Siamese network pre-training for individual modalities.
- Late fusion model combining audio, visual, and text features for final classification.

Human Aided Enhancement

- Semi-live feature allowing continual model learning from new patient data, enhancing adaptability and accuracy.

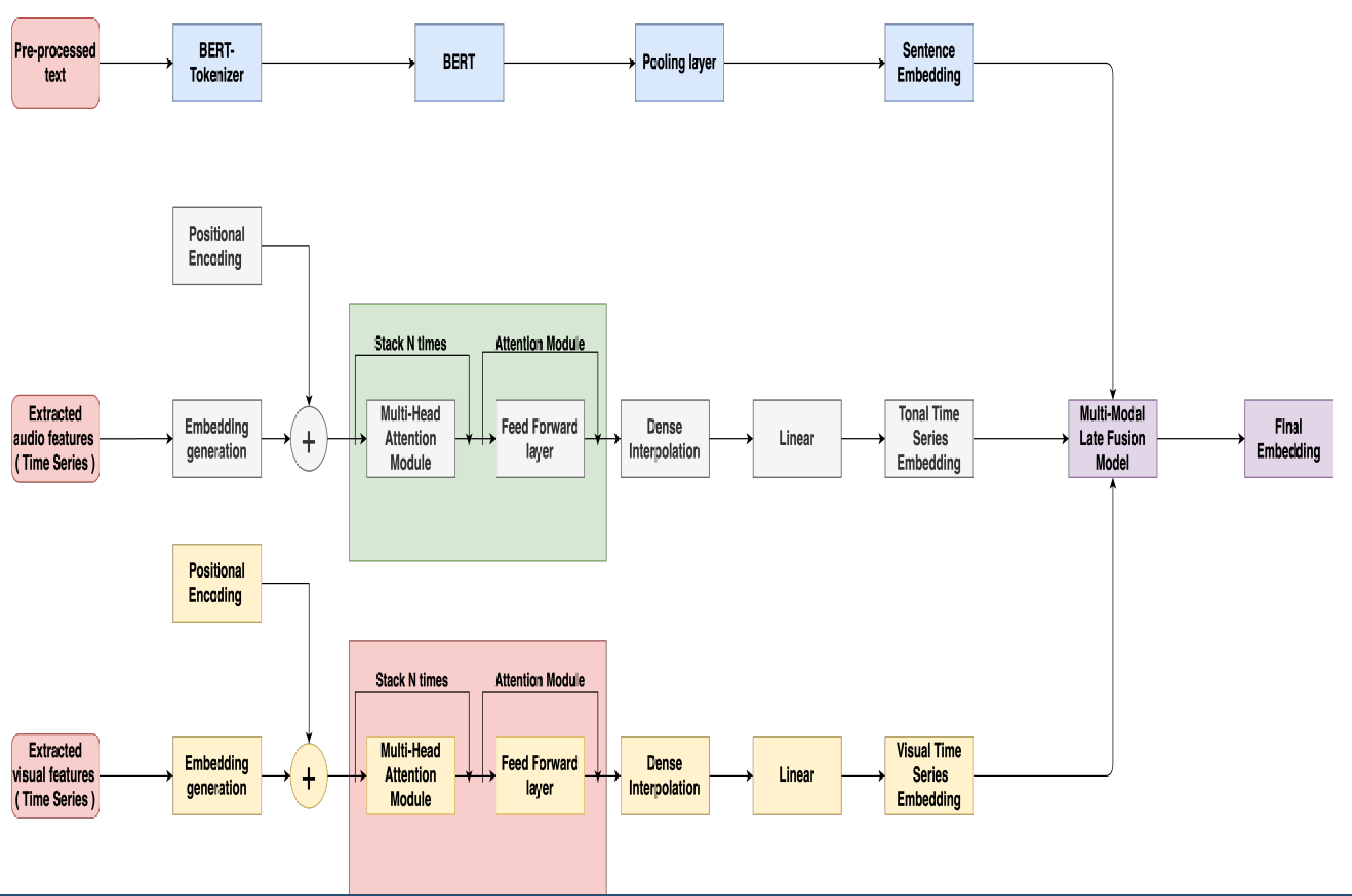
DATASET USED: DAIC-WOZ DATASET



Achieved high accuracy (96.3%) and AUC (0.9682), indicating robustness in real-world scenarios.

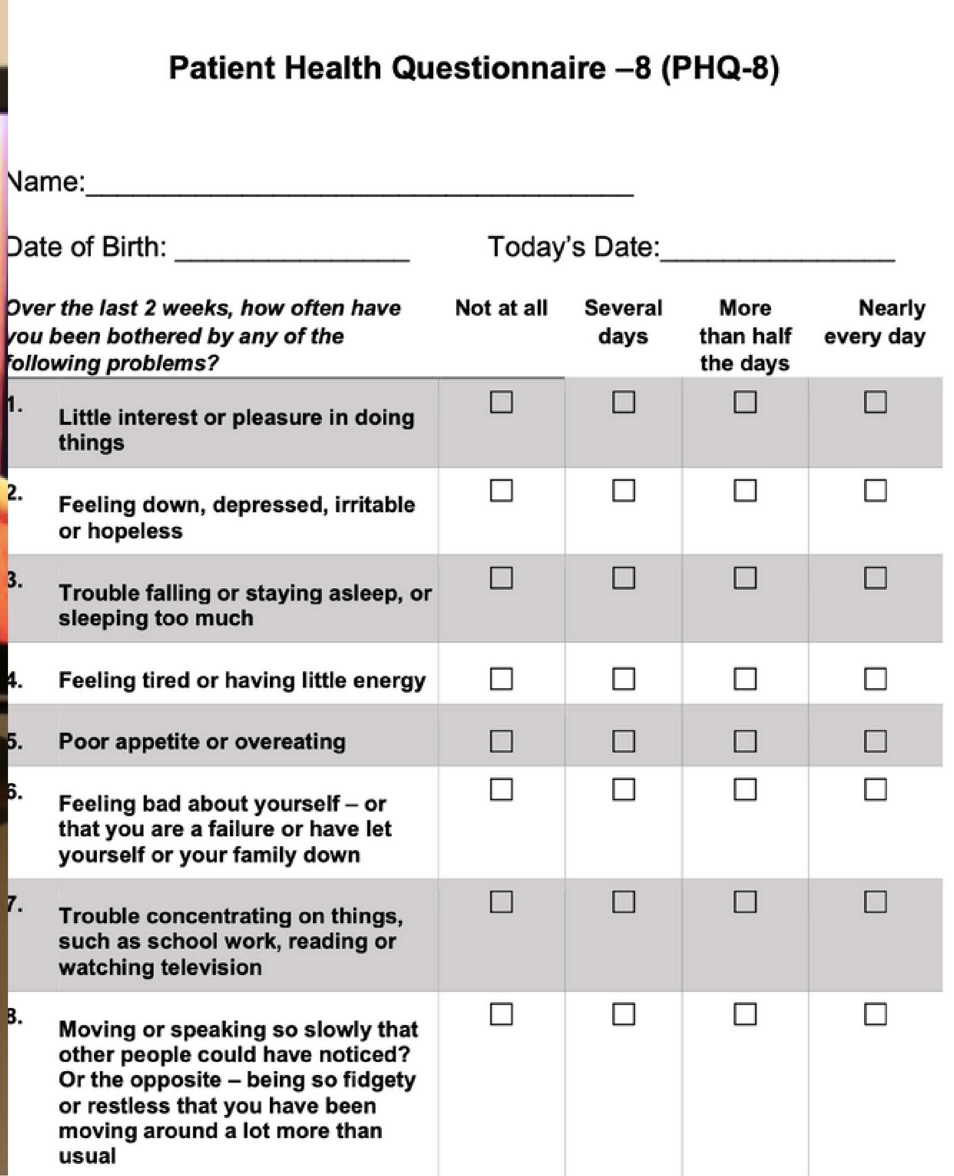
https://arxiv.org/pdf/2011.14966.pdf

FULL MODEL ARCHITECTURE



DATASET

The DAIC-WOZ (Distress Analysis Interview Corpus - Wizard of Oz) Database is a multimodal dataset containing clinical interviews designed to support the diagnosis of psychological distress conditions such as anxiety, depression, and post-traumatic stress disorder. The dataset was chosen for its **multimodal nature,** including audio and video recordings, **extensive questionnaire responses, trustworthiness and ease of accessibility,** making it well-suited for research in the field of mental health.



Patient Health Questionnaire –8 (PHQ-8)

Name:_______________________________

Date of Birth: ______________ Today's Date:______________

Over the last 2 weeks, how often have you been bothered by any of the following problems?	Not at all	Several days	More than half the days	Nearly every day
1. Little interest or pleasure in doing things	☐	☐	☐	☐
2. Feeling down, depressed, irritable or hopeless	☐	☐	☐	☐
3. Trouble falling or staying asleep, or sleeping too much	☐	☐	☐	☐
4. Feeling tired or having little energy	☐	☐	☐	☐
5. Poor appetite or overeating	☐	☐	☐	☐
6. Feeling bad about yourself – or that you are a failure or have let yourself or your family down	☐	☐	☐	☐
7. Trouble concentrating on things, such as school work, reading or watching television	☐	☐	☐	☐
8. Moving or speaking so slowly that other people could have noticed? Or the opposite – being so fidgety or restless that you have been moving around a lot more than usual	☐	☐	☐	☐

The dataset consists of the following features and data points:

- 189 sessions, each averaging 16 minutes
- Audio and video recordings
- Extensive questionnaire responses
- 189 folders of sessions 300-492
- Excluded sessions: 342, 394, 398, 460
- Included sessions with special notes: 373, 444, 451, 458, 480, 402
- 68 2D and 3D facial points, gaze, head orientation, and facial action units
- Transcripts of the interviews
- COVAREP audio features, including F0, VUV, formants, and other audio characteristics

Concerns

All ethical concerns were taken care of, and prior permission to extract audio,video and textual data from the interviewees were taken.

Data Preprocessing

- Stopword removal from transcript data
- Background noise removal from audio files
- Undersampling for removing class imbalance
- Silent Noise Removal
- Convert text to lowercase

Feature Extraction

- For textual data, Word2Vec used for generating embeddings (vectorization) - chosen primarily due to lighter computational expenses.
- Pre-trained models such as BERT and Longformer were tried but rejected owing to them computationally very heavy and expensive.
- Owning to unavailability of actual interview footage, pre-extracted features already provided with the dataset were used for dealing with image/video data.

1. CLNF features - 68 2D points on the face
2. CLNF AUs
3. CLNF features 3D - 68 3D points on the face
4. CLNF gaze - Gaze depiction using head and eye position
5. CLNF hog - HOG features
6. CLNF pose - 6 number feature to keep track of movement using rotation matrices

Feature Extraction

- For audio data, performed after extensive research and literature review on similar projects as also the concept of audio preprocessing in general.

1. MFCCs (Mean): Mean values of Mel-frequency cepstral coefficients, capturing spectral characteristics.
2. MFCCs (Standard Deviation): Standard deviation of Mel-frequency cepstral coefficients, indicating variability in spectral features.
3. MFCCs (Maximum): Identifies the maximum value of Mel-frequency cepstral coefficients, highlighting spectral peaks.
4. MFCCs (Minimum): Reveals the minimum value of Mel-frequency cepstral coefficients, indicating spectral troughs.
5. Chroma: Represents the tonal content, highlighting pitch class distribution in the audio.
6. Spectral Contrast: Captures the difference in amplitude between peaks and valleys in the audio spectrum

Feature Extraction

1. Zero Crossing Rate: Measures the rate of signal crossings through zero, indicating noisiness or percussiveness.
2. Energy: Quantifies the energy in audio frames, showing variations in signal intensity.
3. BPM (Beats Per Minute): Identifies the tempo or rhythm of the audio in beats per minute.
4. Tempo: Represents the tempo information with a different computation method.
5. Onset Strength: Measures the strength of note onsets in the audio, indicating musical events.

Textual Data

```
from sklearn.ensemble import RandomForestClassifier
clf1 = RandomForestClassifier()
param_grid = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_features': ['sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy','log_loss']
}
CV_rfc = GridSearchCV(estimator=clf1, param_grid=param_grid, cv=9)
CV_rfc.fit(X_train_1,Y_train_1)
```

```
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'log2', 'n_estimators': 200}
0.6049382716049383
```

```
Random forests:
              precision    recall  f1-score   support

         0.0       0.79      0.45      0.58        33
         1.0       0.36      0.71      0.48        14

    accuracy                           0.53        47
   macro avg       0.57      0.58      0.53        47
weighted avg       0.66      0.53      0.55        47

Train: 1.0
Test: 0.5319148936170213
```

Random Forest with Hyperparameter Tuning

Textual Data

```python
from sklearn.svm import SVC

classifier = SVC()
param_grid = {
    'C': [0.1, 1, 10, 100, 1000],
    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf', 'poly', 'sigmoid'],
    'class_weight': ['balanced', None],
    'verbose': [True]
}
CV_rfc = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=9)
CV_rfc.fit(X_train_1,Y_train_1)
```

```
{'C': 0.1, 'class_weight': 'balanced', 'gamma': 1, 'kernel': 'rbf', 'verbose': True}
0.5555555555555556
```

SVM:

	precision	recall	f1-score	support
0.0	0.60	0.18	0.28	33
1.0	0.27	0.71	0.39	14
accuracy			0.34	47
macro avg	0.44	0.45	0.34	47
weighted avg	0.50	0.34	0.31	47

Train: 1.0
Test: 0.3404255319148936

SVM with Hyperparameter Tuning

Textual Data

```
XGBoost:
              precision    recall  f1-score   support

         0.0       0.71      0.45      0.56        33
         1.0       0.31      0.57      0.40        14

    accuracy                           0.49        47
   macro avg       0.51      0.51      0.48        47
weighted avg       0.59      0.49      0.51        47

Train: 1.0
Test: 0.48936170212765956
```

XGB Classifier

Textual Data

```
CatBoost:
              precision    recall  f1-score   support

         0.0       0.75      0.36      0.49        33
         1.0       0.32      0.71      0.44        14

    accuracy                           0.47        47
   macro avg       0.54      0.54      0.47        47
weighted avg       0.62      0.47      0.48        47

Train: 1.0
Test: 0.46808510638297873
```

CatBoost Classifier

Audio Data

```
Random forests:
              precision    recall  f1-score   support


         0.0       0.67      0.42      0.52        33
         1.0       0.27      0.50      0.35        14


    accuracy                           0.45        47
   macro avg       0.47      0.46      0.43        47
weighted avg       0.55      0.45      0.47        47


Train: 1.0
Test: 0.44680851063829785
```

Random Forest Classifier

Audio Data

```
SVM:
              precision    recall  f1-score   support

         0.0       0.55      0.36      0.44        33
         1.0       0.16      0.29      0.21        14

    accuracy                           0.34        47
   macro avg       0.35      0.32      0.32        47
weighted avg       0.43      0.34      0.37        47

Train:  0.6666666666666666
Test:  0.3404255319148936
```

SVM Classifier

Audio Data

XGBoost:

	precision	recall	f1-score	support
0.0	0.73	0.48	0.58	33
1.0	0.32	0.57	0.41	14
accuracy			0.51	47
macro avg	0.52	0.53	0.50	47
weighted avg	0.61	0.51	0.53	47

Train: 1.0
Test: 0.5106382978723404

XGB Classifier

Audio Data

```
CatBoost:
              precision    recall  f1-score   support

         0.0       0.61      0.33      0.43        33
         1.0       0.24      0.50      0.33        14

    accuracy                           0.38        47
   macro avg       0.43      0.42      0.38        47
weighted avg       0.50      0.38      0.40        47

Train: 1.0
Test: 0.3829787234042553
```

CatBoost Classifier

Video Data

```
Random forests:
              precision    recall  f1-score   support

           0       0.69      0.67      0.68        33
           1       0.27      0.29      0.28        14

    accuracy                           0.55        47
   macro avg       0.48      0.48      0.48        47
weighted avg       0.56      0.55      0.56        47

Train: 1.0
Test: 0.5531914893617021
```

Random Forest Classifier

Video Data

```
SVM:
              precision    recall  f1-score   support

         0.0       0.70      1.00      0.82        33
         1.0       0.00      0.00      0.00        14

    accuracy                           0.70        47
   macro avg       0.35      0.50      0.41        47
weighted avg       0.49      0.70      0.58        47

Train: 0.5961538461538461
Test: 0.7021276595744681
```

SVM Classifier

Video Data

```
XGBoost:
              precision    recall  f1-score   support

           0       0.80      0.73      0.76        33
           1       0.47      0.57      0.52        14

    accuracy                           0.68        47
   macro avg       0.64      0.65      0.64        47
weighted avg       0.70      0.68      0.69        47

Train: 1.0
Test: 0.6808510638297872
```

XGBoost Classifier

Video Data

```
CatBoost:
              precision    recall  f1-score   support

           0       0.74      0.61      0.67        33
           1       0.35      0.50      0.41        14

    accuracy                           0.57        47
   macro avg       0.55      0.55      0.54        47
weighted avg       0.62      0.57      0.59        47

Train: 1.0
Test: 0.574468085106383
```

CatBoost Classifier

Individual Best Weights

- We find the best combination of weights for three sets of predictions of the same classifier.
- For each combination of weights, we calculate a weighted average of the three prediction sets
- Then, we convert the weighted average into binary predictions using a threshold of 0.5 and calculate the F1 score

```
# find the best weights
best_weights = [0, 0, 0]
best_score = 0
for i in range(0, 100):
    for j in range(0, 100):
        for k in range(0, 100):
            if(i + j + k == 0):
                continue
            weights = [i/100, j/100, k/100]
            final_preds = (weights[0] * Y_pred_xgb_1 + weights[1] * Y_pred_xgb_2 + weights[2] * Y_pred_xgb_3) / sum(weights)
            final_binary_preds = np.where(final_preds > 0.5, 1, 0)
            # score = f1_score(final_binary_preds, Y_test_3)
            score = f1_score(Y_test_3, final_binary_preds)
            if(score > best_score):
                best_score = score
                best_weights = weights
```

Final Random Forest Classifier

```
[0.05, 0.04, 0.01]
```

```
Final Random Forest:
              precision    recall  f1-score   support

           0       0.83      0.45      0.59        33
           1       0.38      0.79      0.51        14

    accuracy                           0.55        47
   macro avg       0.61      0.62      0.55        47
weighted avg       0.70      0.55      0.57        47
```

Final SVM Classifier

```
[0.01, 0.0, 0.0]
```

Final SVM:

	precision	recall	f1-score	support
0	0.60	0.18	0.28	33
1	0.27	0.71	0.39	14
accuracy			0.34	47
macro avg	0.44	0.45	0.34	47
weighted avg	0.50	0.34	0.31	47

Final XGBoost Classifier

```
[0.01, 0.01, 0.02]
```

Final XGBoost:

	precision	recall	f1-score	support
0	0.83	0.88	0.85	33
1	0.67	0.57	0.62	14
accuracy			0.79	47
macro avg	0.75	0.73	0.73	47
weighted avg	0.78	0.79	0.78	47

Final CatBoost Classifier

```
[0.01, 0.0, 0.0]
```

```
Final CatBoost:
              precision    recall  f1-score   support

           0       0.75      0.36      0.49        33
           1       0.32      0.71      0.44        14

    accuracy                           0.47        47
   macro avg       0.54      0.54      0.47        47
weighted avg       0.62      0.47      0.48        47
```

Final Best Weight

- We find the best combination of weights for three sets of predictions: Y_pred_rf_1, Y_pred_xgb_2, and Y_pred_xgb_3 (RF, XGB, XGB)
- Apply the same principle as before

```
# find the best weights
best_weights = [0, 0, 0]
best_score = 0
for i in range(0, 100):
    for j in range(0, 100):
        for k in range(0, 100):
            if(i + j + k == 0):
                continue
            weights = [i/100, j/100, k/100]
            final_preds = (weights[0] * Y_pred_rf_1 + weights[1] * Y_pred_xgb_2 + weights[2] * Y_pred_xgb_3) / sum(weights)
            final_binary_preds = np.where(final_preds > 0.5, 1, 0)
            score = f1_score(Y_test_3, final_binary_preds)
            if(score > best_score):
                best_score = score
                best_weights = weights
```

```
[0.01, 0.01, 0.02]
```

Final Model

- Random Forest for Text Classification and XG Boost for Audio and Video Classification is used as the final model.

Final Model:

	precision	recall	f1-score	support
0	0.82	0.85	0.84	33
1	0.62	0.57	0.59	14
accuracy			0.77	47
macro avg	0.72	0.71	0.71	47
weighted avg	0.76	0.77	0.76	47

LSTM without Gating (SL)

- **Model Definition**
  - Input Branches:
    - 250 time steps
    - Audio: 74 features, processed by 1 dense layer.
    - Text: 5100 features, processed by 3 dense layers (1000, 500, 250).
  - LSTM Layer: 128 units with 20% dropout.
  - Output Layer: 1 unit with sigmoid activation for binary classification.
- **Model Compilation and Optimization**
  - Adam Optimizer: Learning rate = 0.0001
    Loss Function: Binary Cross-Entropy

LSTM without Gating (SL)

- **Model Definition**

  Input Branches:
    - 250 time steps
    - Video: 388 features, processed by 1 dense layer.
    - Text: 5100 features, processed by 2 dense layers (1000 and 500).
  - LSTM Layer: 128 units with 20% dropout.
  - Output Layer: 1 unit with sigmoid activation for binary classification.
- **Model Compilation and Optimization**
  - Adam Optimizer: Learning rate = 0.0001
  - Loss Function: Binary Cross-Entropy

LSTM without Gating (SL)

- **Model Definition**
  - Input Branches:
    - 250 time steps
    - Audio: 74 features, processed by 1 dense layer.
    - Video: 388 features, processed by 2 dense layers (200 and 74).
    - Text: 5100 features, processed by 4 dense layers (1000, 500, 250, and 74).
  - LSTM Layer: 128 units with 20% dropout.
  - Output Layer: 1 unit with sigmoid activation for binary classification.
- **Model Compilation and Optimization**
  - Adam Optimizer: Learning rate = 0.0001
  - Loss Function: Binary Cross-Entropy

LSTM with Gating

- "Gating" in LSTM (Long Short-Term Memory) networks is crucial for controlling the flow of information through the network.
- It helps the network to decide how much of the past information needs to be passed along to the future.

Word-level Gating

Processes and applies gating mechanisms at the level of individual words in a text.

Sentence-level Gating

Treats entire sentences as units for processing and applies gating mechanisms at this higher level.

LSTM with Gating (SL)

- **Highway Layer (Custom Layer)**
  - Transform gate
  - Activation Gate
  - Carry Gate: Computed as 1.0−Transform Gate
- **Model Definition**
  - Input Branches:
    - 250 time steps
    - Video: 388 features, processed through 3 consecutive Highway layers.
    - Text: 5100 features, processed by 2 dense layers (1000 and 500 units).
  - LSTM Layer: 128 units with 20% dropout.
  - Output Layer: 1 unit with sigmoid activation for binary classification.
- **Model Compilation and Optimization**
  - Adam Optimizer: Learning rate = 0.0001
  - Loss Function: Binary Cross-Entropy

LSTM with Gating (SL)

- **Highway Layer (Custom Layer)**
  - Transform gate
  - Activation Gate
  - Carry Gate: Computed as 1.0−Transform Gate
- **Model Definition**
  - Input Branches:
    - 250 time steps
    - Audio: 74 features, processed through 3 consecutive Highway layers.
    - Text: 5100 features, processed by 2 dense layers (1000 and 500 units).
  - LSTM Layer: 128 units with 20% dropout.
  - Output Layer: 1 unit with sigmoid activation for binary classification.
- **Model Compilation and Optimization**
  - Adam Optimizer: Learning rate = 0.0001
  - Loss Function: Binary Cross-Entropy

LSTM with Gating (SL)

- **Highway Layer (Custom Layer)**
  - Transform gate
  - Activation Gate
  - Carry Gate: Computed as 1.0−Transform Gate
- **Model Definition**
  - Input Branches:
    - 250 time steps
    - **Audio:** 74 features, processed through 3 consecutive **Highway** layers.
    - Video: 388 features, processed through 3 consecutive Highway layers.
    - Text: 5100 features, processed by 4 dense layers (1000, 500, 250, and 74).
  - LSTM Layer: 128 units with 20% dropout.
  - Output Layer: 1 unit with sigmoid activation for binary classification.
- **Model Compilation and Optimization**
  - Adam Optimizer: Learning rate = 0.0001
  - Loss Function: Binary Cross-Entropy

Model Training

- Validation Split: 0.2
- EarlyStopping Callback
  - Monitors validation loss for improvement.
  - Minimum change in loss (min_delta): 0
  - Patience: 15 epochs (stops if validation loss does not improve for 15 consecutive epochs).
  - Restores best weights when training stops.
- Epochs: 50
- Batch Size: 32

LSTM with Gating (WL)

- **Highway Layer (Custom Layer)**
  - Transform gate
  - Activation Gate
  - Carry Gate: Computed as 1.0−Transform Gate
- **Model Definition**
  - Input Branches:
    - 1700 time steps
    - **Audio:** 74 features, processed through 3 consecutive **Highway** layers.
    - Video: 388 features, processed through 3 consecutive Highway layers.
    - Text: 5100 features, processed by 4 dense layers (1000, 500, 250, and 74).
  - LSTM Layer: 128 units with 20% dropout.
  - Output Layer: 1 unit with sigmoid activation for binary classification.
- **Model Compilation and Optimization**
  - Adam Optimizer: Learning rate = 0.0001
  - Loss Function: Binary Cross-Entropy

Model Training

- Validation Split: 0.2
- EarlyStopping Callback
  - Monitors validation loss for improvement.
  - Minimum change in loss (min_delta): 0
  - Patience: 15 epochs (stops if validation loss does not improve for 15 consecutive epochs).
  - Restores best weights when training stops.
- Epochs: 50
- Batch Size: 2

PERFORMANCE METRICS

Model	Modality	Precision	Recall	F1-Score	Accuracy
LSTM without Gating	Text + Audio	0.69	0.34	0.46	0.42
	Text + Video	0.61	0.43	0.51	0.48
	Text + Audio + Video	0.58	0.52	0.55	0.54
LSTM with sentence-level Gating	Text + Audio	0.64	0.63	0.63	0.59
	Text + Video	0.63	0.57	0.6	0.57
	Text + Audio + Video	0.68	0.67	0.68	0.66
LSTM with word-level Gating	**Text + Audio + Video**	0.51	0.64	0.57	0.54

Challenges

- Scalability
- Data Scarcity
- Cultural Bias
- Data Privacy and Security
- Ethical Concerns
- User Acceptance and Stigma
- Diversity and Inclusivity

Deployment at a Plaksha

Integration with Counseling Services

Student Health Monitoring

Research and Development Collaboration

Training and Awareness Programs

THANK YOU!