Audio-Based Stress Detection through Deep Learning using DIAC WOZ dataset

Jai Moryani[1,2], Harsh Sharma[1,3], Sumit Prakash[1,4], Aryan Pandey[1,5], Saransh Yadav[1,6], Saransh Gupta[1,7], and Kushagra Jain[1,8]

[1] Indian Institute of Information Technology, Allahabad
[2] iit2021122@iiita.ac.in
[3] iit2021131@iiita.ac.in
[4] iit2021152@iiita.ac.in
[5] iit2021153@iiita.ac.in
[6] iit2021162@iiita.ac.in
[7] iit2021163@iiita.ac.in
[8] iit2021191@iiita.ac.in

1 Software Requirements Specification (SRS)

1.1 Objective:

Develop a audio-based stress detection system using deep learning techniques and the Extended DIAC WOZ dataset to provide automated and objective stress detection.

1.2 Benefits:

- Contact-free monitoring.
- Objective stress detection.
- Real-time insights.
- Holistic assessment.

1.3 Functional Requirements:

1. **Audio Data Processing:**
   - Preprocess and extract audio features and frequency cues from audio clips.
2. **Deep Learning Model:**
   - Implement a LSTM(Long Short-term Memory) Model.
3. **Stress Detection:**
   - Classify stress levels and evaluate performance using metrics.
4. **User Interface (UI):**
   - Allow video upload and display stress detection results.
5. **System Performance:**
   - Real-time processing and scalability.

1.4 Non-Functional Requirements:

1. **Performance:**
   - Real-time processing and high throughput.
2. **Usability:**
   - User-friendly interface and compatibility with major web browsers.
3. **Accuracy:**
   - Achieve at least 70% stress detection accuracy.
4. **Documentation:**
   - Provide user and developer documentation.

1.5 Dataset:

The Extended DAIC-WOZ dataset includes audiovisual recordings from over 200 subjects, featuring emotions like neutral, happy, sad, angry, and fearful. Interviews last 15-30 minutes, offering diverse multimodal data for mental health analysis. Researchers commonly employ it for tasks like emotion recognition, depression detection, and stress analysis. The dataset, anonymized and ethically handled, is publicly available for scientific research.

1.6 System Evaluation:

Focus on F1-Score, precision, recall, and accuracy using Extended DIAC WOZ dataset.

1.7 Related Work:

Previous research includes audio-based,image-based and video-based stress detection, as well as fusion of visual and thermal data for improved accuracy.

1.8 Implementation

Platform This section outlines the tools and libraries used in our machine learning project, focusing on video-based stress detection using the Extended DIAC WOZ dataset.

1. **scikit-learn (sklearn)**
   - **train_test_split** from sklearn.model_selection: This function plays a fundamental role in supervised machine learning by splitting the dataset into training and testing sets, ensuring model evaluation.
   - **LabelEncoder** from sklearn. preprocessing: The LabelEncoder is employed to encode categorical labels as integers, a crucial step in making data compatible with machine learning algorithms.
   - **confusion_matrix** from sklearn. metrics: This function helps assess the performance of classification models by tabulating true positives, true negatives, false positives, and false negatives.

   - **classification_report** from sklearn. metrics: The classification_report function generates a comprehensive report with precision, recall, F1-score, and support metrics, providing a holistic view of model performance.
2. **TensorFlow (tensorflow)**
   - **Sequential** from tensorflow. keras. models: The Sequential class is integral for building sequential neural network models, allowing the stacking of layers in a linear fashion.
   - **Dense, Dropout, LSTM, Flatten, Conv1D, MaxPooling1D** from tensorflow. keras. layers: These layers are essential for constructing neural network architectures. Dense represents fully connected layers, Dropout helps in mitigating overfitting, LSTM is used for recurrent neural networks, and Conv1D and MaxPooling1D are key components in 1D convolutional neural networks.
3. **Google Colab (google. colab)**
   - **drive** from google. colab: The drive module is utilized to interact with Google Drive within a Google Colab environment, simplifying data storage and retrieval in collaborative online settings.
   - **GPU used for training:** It's noteworthy that the project benefited from the GPU (Tesla K80) available in the Google Colab environment, enhancing the efficiency of computations.

In this project, the synergistic use of scikit-learn, TensorFlow, and Google Colab formed a robust platform for machine learning and deep learning. These libraries, each with its unique set of tools, enabled critical tasks including data preprocessing, model creation, and model performance evaluation.

Framework The code utilizes several Python libraries and frameworks:

1. **Pandas:** For data manipulation and working with DataFrames.
2. **NumPy:** For numerical computations and array manipulation.
3. **Librosa:** For audio processing tasks like trimming and feature extraction.
4. **scikit-learn:** For various machine learning utilities such as train-test splitting and label encoding.
5. **TensorFlow and Keras:** For building, training, and evaluating deep learning models, including dense neural networks, convolutional neural networks (CNNs), and long short-term memory networks (LSTMs).
6. **Matplotlib:** For generating plots and visualizations.

The code follows a structured approach, making use of functions for modularity and clarity. It encompasses data processing, feature extraction, model creation, compilation, training, evaluation, and visualization. The choice of models (DNN, CNN, LSTM) reflects an exploration of different neural network architectures for the given task.

1.9 Activity Time Chart

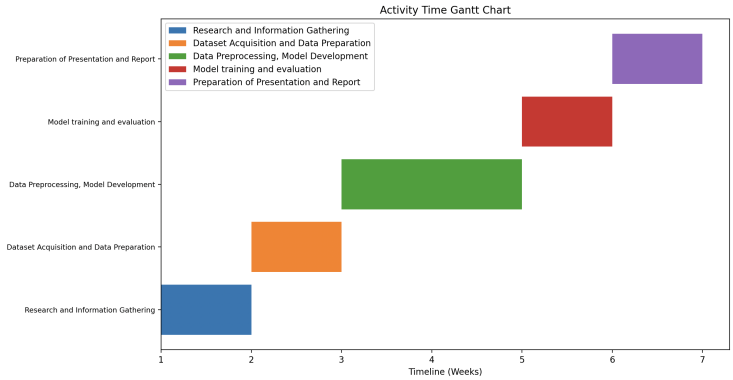

Fig. 1. Activity Time Gantt Chart

1.10 Working

1. **Data Preparation**
   (a) Define Main Folder
   `main_folder is defined as "mini_project".`

   (b) File Paths and Label Names
   `# Initialize empty lists file_paths and label_names`

   (c) Loop for File Paths and Labels
   ```
   # Loop over file_names and labels
   # Create file_path and label_name
   # Append to file_paths and label_names
   ```

   (d) Create DataFrame
   `# Create DataFrame data_df using file_paths and label_names`

   (e) Display Data
   `# Display data_df`

   (f) Label Distribution
   `# Count of unique labels in data_df`

2. **Audio Preprocessing Functions**
   (a) Function: trimm

```
Define the trimm function.
```

(b) Function: pitch

```
Define the pitch function.
```

3. **Feature Exactraction Functions**
   (a) Function extract_features

```
Define the extract_feature function.
```

4. **Data Loading Functions**
   (a) Function: load Data

```
Define the load data function.
```

   (b) Loading DATA

```
Use loadData function to load data_df into X_train, X_test, y_train, y_test
```

5. **DataFrame Creation for Features**
   (a) X_train DataFrame

```
# Convert X_train to DataFrame X_train_df with column names feat_0, feat_1, etc.
```

6. **DataFrame Creation for Labels**
   (a) y_train DataFrame

```
# Convert y_train to DataFrame y_train_df with column name Label
```

7. **Label Encoding**
   (a) LabelEncoder

```
# Use LabelEncoder to encode labels in y_train and y_test
# storing them in new columns 'Labels'
```

8. **Conversion to NumPy Arrays**
   (a) Label Arrays

```
# Convert label arrays yTrainDNN and yTestDNN to NumPy arrays
```

   (b) Shape of Label Arrays

```
# Print the shape of yTrainDNN
```

9. **List of Observed Emotions**

```
observedEmotions is defined as a list containing 'Distressed' and 'Stressed'.
```

10. **Deep Learning Model with Dense Layers (modelDNN)**
    (a) Model Definition

```
# Define the modelDNN with Dense layers
```

    (b) Model Compilation

```
# Compile the model with binary cross-entropy loss and Adam optimizer
```

(c) Summary of the Model
```
# Display summary of modelDNN
```

(d) Model Training
```
# Train the modelDNN, storing training history in historyDNN
```

(e) Plotting Training History
```
# Plot accuracy and loss over epochs for modelDNN
```

(f) Confusion Matrix
```
# Compute and plot confusion matrix for modelDNN
```

(g) Classification Report
```
# Generate classification report for modelDNN
```

11. **Deep Learning Model with Convolutional Layers (modelCNN)**
(a) Model Definition
```
#Define the modelCNN with Conv1D and MaxPooling1D layers
```

(b) Model Compilation and summary
```
# Compile the modelCNN and display its summary
```

(c) Model Training
```
# Train the modelCNN, storing training history in historyCNN
```

(d) Plotting Training History
```
# Plot accuracy and loss over epochs for modelCNN
```

(e) Confusion Matrix
```
# Compute and plot confusion matrix for modelCNN
```

12. **Preparation for LSTM**
(a) Data Truncation
```
# Truncate data to have 39 features per sample
```

(b) Data Reshaping for LSTM
```
# Reshape and expand data for LSTM compatibility
```

(c) One-Hot Encoding for LSTM Labels
```
# One-hot encode labels for LSTM
```

13. **Model with LSTM Layers (modelLSTM)**
(a) Model Definition

```
# Define the modelLSTM with LSTM, Dense, and Dropout layers
```

(b) Model Compilation and summary

```
# Compile modelLSTM with binary cross-entropy loss and Adam optimizer
# Display summary of modelLSTM
```

(c) Model Training

```
# Train modelLSTM, storing training history in historyLSTM
```

(d) Plotting Training History

```
# Plot accuracy and loss over epochs for modelLSTM
```

(e) Confusion Matrix

```
# Compute and plot confusion matrix for modelLSTM
```