Emotion Recognition

Arashdeep Mehroke Mohamed Hasan

Dec 5, 2024

1 Problem Statement

Recognizing human emotions is an essential aspect of understanding and improving human-computer interaction, providing insights into behavioral analysis, mental health monitoring, and customer experience optimization. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) offers a rich collection of labeled audio data that can be leveraged to develop robust machine learning models for emotion classification.

Despite the availability of advanced feature extraction techniques such as YAMNet embeddings and Librosa-based handcrafted features, achieving high accuracy in emotion recognition remains a challenge due to:

High Dimensionality of Features: Audio features like YAMNet embeddings are inherently high-dimensional, complicating the training process for traditional models. Imbalanced Datasets: Emotional classes often exhibit uneven distributions, leading to biased model performance. Complexity of Audio Signals: Variations in pitch, tone, and intensity across speakers make it difficult to accurately classify emotions. Computational Overheads: Extracting, visualizing, and analyzing audio features for large datasets requires efficient preprocessing and dimensionality reduction. This project aims to address these challenges by:

Employing state-of-the-art feature extraction techniques (e.g., YAMNet, Librosa). Balancing the dataset using synthetic techniques like SMOTE. Experimenting with various machine learning and deep learning models to identify optimal solutions. Visualizing high-dimensional feature spaces with dimensionality reduction tools like t-SNE and UMAP for better interpretability. The goal is to build a robust, scalable system capable of accurately classifying human emotions across multiple categories while ensuring computational efficiency and practical applicability.

2 Dataset

The dataset consists of 3117 satellite images and corresponding masks of size 1280×720, with each mask delineating the satellite from the background. The images are of varying resolutions and quality, reflecting the diversity of satellite imagery encountered in real-world scenarios. The dataset is divided into training and validation sets, with the training set containing 2517 images and the validation set comprising 600 images. The masks are categorized into fine and coarse masks, with 403 fine masks and 2114 coarse masks in the training set, the validation set contains all fine masks. The dataset presents a challenging segmentation task due to the presence of complex textures, varying lighting conditions, and the need to accurately delineate the satellite from the background.

2.1 Data Preprocessing

The images are resized to a fixed resolution of 256×256 pixels and the masks are converted to binary format, with pixel values of 0 and 1 representing the background and satellite, respectively, to reduce computational complexity, since my computer has limited memory. The preprocessed dataset has the same split between training and validation sets as the original dataset.

The file `data_loader_unit.py` in the `utils` folder contains the function `load_and_process_files()` to load the dataset and preprocess the images and masks. After processing, the data will be saved as numpy arrays for easy access during training. The saved files are named as follows:

- **prepped_data/trainimages.npy**: Contains the preprocessed training images.
- **prepped_data/trainmasks.npy**: Contains the preprocessed training masks.
- **prepped_data/valimages.npy**: Contains the preprocessed validation images.
- **prepped_data/valmasks.npy**: Contains the preprocessed validation masks.

The function to check if the prepped files exist is `check_prepped_data()` in the `utils` folder. It is used in the scripts `train.py`, `predict.py`, and `main.py`. If the prepped files do not exist, the function is called to create them. Now, let's look at the processed dataset.