Bark Buddies – README

Structure of Code

Our code submission is structured as follows:

- **videos** - a folder containing short animation clips of the virtual dog that are queued throughout the game
- **audio.py** - a python script containing all audio / speech related functionality
- **barkBuddy.py** - the main python script that runs the game, this script also contains the implementation of our late fusion model
- **eval.py** - a short python script that calculates the performance of our model
- **face.py** - a python script containing all video / facial expression related functionality
- **game.py** - a python script containing all game logic functionality
- **my_model.h5** - a pre-trained model used for facial recognition, citations for this model are provided in the face.py script and report.
- **Dataset.zip** - a zipped folder containing the videos and an excel sheet of tabularized data

Structure of Dataset

Our dataset can be found in Dataset.zip, the structure of this folder is outlined below:

- **Dataset.xlsx** - an excel sheet of data for our data set
- **video clips** - a folder containing the video clips of our data set

Self-Evaluation of Project

The proposal we wrote specified that we would create an interactive game that could process real-time social signals across multiple modalities and generate immediate user feedback in the form of the virtual dog's behaviour. While we did accomplish creating a system that can process multiple modalities, it is far from the responsiveness you would expect from a real-time system. In our proposal, we acknowledged that processing multiple modalities quickly would likely be a challenge for our system and this can be seen by the delay in the system between the initial user interaction and the reaction of the virtual dog. The delay in our system is due to a couple of factors. Firstly, we decided to have our system collect audio and video data for processing in 5-second bursts, after which the collected data would be processed and passed into our pre-trained model for classification. Even though we utilized threads in our application to parallelize various operations, we found that the continuous collection of data was a lot for our system to handle and we chose to simplify the implementation in this regard. Secondly, we had issues running our Python code from within Unity, so the animations of our dog are saved as videos and queued for display rather than interfacing with Unity. We had initially planned to integrate our system within Unity to speed up the queueing of animations but time constraints prevented us from doing so. Our project still functions as an interactive game but is a much slower one in Python but powered by Unity 3D animations as videos.

Our proposal suggested a relatively large scope for our project as it involved working with multiple modalities (facial expressions, tone, and speech sentiment) and attempting to make it real-time, especially for a group of three people. The suggestion to use pre-trained models to process the information for each of our modalities was a big help, this decision was included in our proposal and ultimately implemented in our final project. We did however change the logic of our late fusion system. In our proposal, we specified that we would use a perceptron to infer the labels and confidence based on the labels and confidence predicted for each modality input. In our final system, we simplified this part of the system to instead calculate the average confidence for each of the arousal and valence scores, and select the output of the system to be the arousal and valence scores with the highest average confidence. The confidence from our late fusion was then the average of the highest average confidences for valence and arousal.

In terms of our data and dataset creation, not much was changed. The dataset is self-annotated and records the dog's reaction to different audio and visual cues. The data points are discrete and cataloged in an Excel file and associated video clips showing dog's reactions are also attached. Instead of having emotion labels, we changed our model to output valence and arousal levels only for classification. The valence and arousal values are modified versions of the PAD and Parrot Model.
For evaluation, we assigned ground-truth labels to our dataset and then compared our models' performance against that. A 55% accuracy and 55.6% precision for valence and an 85% accuracy and 84.7% precision for arousal was reported by our model. The confusion matrices of both are shown below:

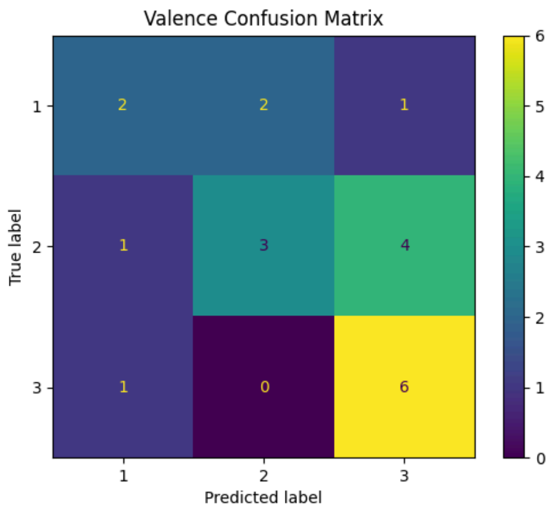

Fig. X. Valence Confusion Matrix

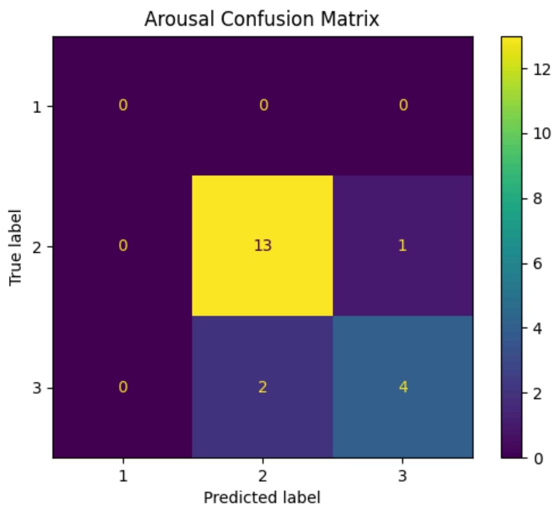

Fig. Y. Arousal Confusion Matrix

The distribution among team members also changed from the initial description provided in the proposal. Raj primarily worked on creating the animations in Unity and the logic to queue them within our Python script. Aru worked on receiving audio input, converting speech to text, performing semantic analysis, and tone inference. Kaleigh worked on receiving video input, facial expression recognition, and the late fusion system.

Dependencies

The pip commands for each dependency are provided. For specific OS, the corresponding links can be used for a successful installation.

Facial Analysis Dependencies

- OpenCV
    - `pip install opencv-python`
    - opencv-python · PyPI
- Numpy
    - `pip install numpy`
    - Installing NumPy
- TensorFlow
    - `pip install tensorflow`
    - Install TensorFlow 2
- PIL
    - `pip install pillow`
    - pillow · PyPI

Audio Analysis Dependencies

- Whisper by OpenAI
    - pip install -U openai-whisper
    - GitHub - openai/whisper: Robust Speech Recognition via Large-Scale Weak Supervision
    - ffmpeg might be needed for whisper to work
        - sudo apt update && sudo apt install ffmpeg
        - Download FFmpeg
- Transformers
    - pip install transformers
    - transformers · PyPI
- Librosa
    - python -m pip install librosa
    - GitHub - librosa/librosa: Python library for audio and music analysis
- Torch

        - pip3 install torch torchvision torchaudio
        - Start Locally | PyTorch
    - Pyaudio
        - pip install PyAudio
        - PyAudio · PyPI
    - Wave
        - pip install Wave
        - Wave · PyPI

Game Dependencies

- Unity Version: 2021.1.6+ (Local Version 2021.3.25)
- Installation guide:
    - Manual: Downloading and installing Unity