AIPet: A Constant Personalized Companion

Li-Cheng Chien, Pei-Ju Chien, Chia-Yun Hsu, Yi-Jen Ju, Chia-Yuan Kuo

***Abstract*—Our framework aims to create a machine-learning-based virtual pet capable of recognizing the user's emotion and responding appropriately. We also want to design such a virtual pet to tune to the user's specific needs. This paper proposes a method that uses weights to distinguish the quality of virtual pet behavior to enhance the suitability of an AI pet to a specific user. The emotion recognition model we use is Convolution Neural Network (CNN)-based, which is suitable for building a model to recognize human emotions from voice. Our model relies solely on human voice data and features relevant to emotion recognition, such as pitch and waveform, without semantic analysis. The model is trained with human voice speaking English, and therefore performs best when the user uses English. The performance of the model is evaluated by accuracy.**

***Keywords*—deep learning, emotion recognition, CNN**

Introduction

Both anecdotal and qualitative research evidence can confirm that social interaction greatly improves a person's emotions and mental health. We often find ourselves in a more positive emotional state whenever we are accompanied by friends, family, or even just pets. In an increasingly digitized and urbanized world, however, physical interaction with our closest friends, family, or pets may not always be readily available. As a result, when we inevitably encounter setbacks that engender negative emotions, we can struggle to find ways to bring ourselves to a more positive emotional state.

One solution to the lack of physical interaction with people or pets in the modern world is virtual social interaction. Social media and many other platforms enable interactions with people not in our immediate vicinity. However, interaction with people on social media can be limited by their availability; our friends and family who also lead busy lives might not always be able to console us out of our negative emotions. Virtual pets, on the other hand, can be available to us whenever we are in need.

We propose a new framework, AIPet, that provides personalized pet images best suited for the user in different emotional states. We believe that individuals would respond differently to the same stimuli under the same emotional state, and in turn would require different pet images to bring them to a more positive emotional state. Our framework is designed to tune to a user's specific preferences toward certain images against others based on historic usage, thereby providing the most effective image to bring a specific user to a more positive emotional state.

Related Works

DATABASE

The paper for Emov-DB in 2018 ***The Emotional Voices Database: Towards Controlling the Emotion Dimension in Voice Generation Systems*** provides an open-source database of emotional speech intended to be used for synthesis and generation purposes.

The database contains data for 2 male and 2 female actors in English and a male actor in French respectively. It covers 5 emotion classes, including Amused, Angry, Disgusted, Neutral, and Sleepy, with which we chose to train our model. [1]

MODEL

We refer to the model offered by Renovamen as our model's first layer. Renovamen's model uses CNN、LSTM 、SVM and MPL to achieve speech emotion recognition [2]. The tool it required is Opensmile, which can extract sounds feature and save them as .csv file. It uses python modules including scikit-learn、Keras、TensorFlow、librosa、SciPy、pandas、Matplotlib and numpy. They used scikit-learn to build SVM & MLP model and split data for training and testing. After Opensmile extracts sounds feature, librosa can address them and draw the oscillogram.

The second layer of our model was built by ourselves. We import package 'time' [3] in order to monitor whether the data is updated. The user interactive part includes voice recording and screen showing. Voice is recorded and addressed by 'sounddevice', a package of Python, which can deal with numerous kinds of devices. We show our AIPet by screen provided module 'Tkinter' in Python [4]. With the screen, we can let users interact with our AIPet.

Methods

Framework Architecture

Our proposed framework consists of two models, one takes users' voice as input data and determines its associated emotion, and the other selects an appropriate image in graphics interchange format, or GIF, and updates the user's preferences toward certain images against others as the user continues to use our interface.

The first model is a 17-layer CNN model, which includes convolution, dropout, and max-pooling layers. The model outputs which of the 8 states of emotion the user is in.

The second model takes as input the output of the first model and selects a GIF to display based on the emotion of the user. Since there are 8 possible emotional states in which the user might be, there are 8 possible outputs, one for each emotion, in each iteration of the model's usage. The model then collects the user's response in voice to the displayed GIF and feeds it back to the first model to determine how the GIF has changed the user's emotional state. This information is then used to update this second model with respect to how the model selects the appropriate GIF for each emotion.

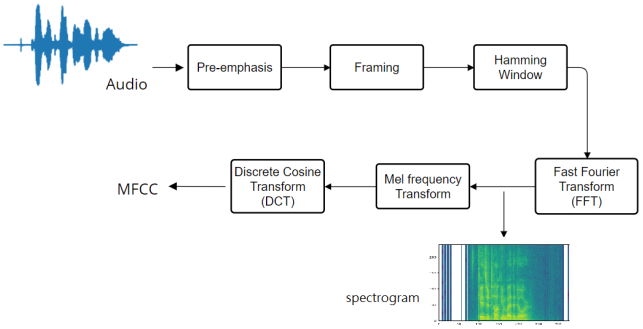
Preprocessing Audio Data

1. *Mel-Frequency Cepstral Coefficients (MFCC)*

In sound processing, we often transform the time series data into power spectrum, which describes the distribution of power of a time series as its frequency components, because it can reveal certain aspects of the signal not apparent in the time domain. For example, the power spectrum can have a distinct peak indicating a sine wave component, or peaks corresponding to harmonic series.

Specifically, mel-frequency cepstral coefficients, or MFCC, are coefficients that collectively define a mel-frequency cepstrum. The mel-frequency cepstrum is a representation of the short-term power spectrum of a sound. Cepstrum is the inverse discrete Fourier transform, or IDFT, of a spectrum; it is the rate of change in spectral bands derived by applying Fourier transform on the time series data, then taking the log of the magnitude of this Fourier spectrum, then again taking the spectrum by cosine transformation. Such preprocessing reveals features most relevant to analysis of human voice. The frequency bands are equally spaced on the mel scale, which relates the perceived frequency of a tone to the actual measured frequency; it scales the frequency in order to match more closely to what human ears can hear than linearly-spaced frequency bands.

We extracted the MFCC to represent the audio clip. We kept approximately 40 MFCC's from each audio clip to use as training features.



2. *Mel Spectrogram*

Spectrogram is a visual representation of signal strength over time at various frequencies. In particular, the mel spectrogram visualizes sound on the mel scale. Transferring audio to the mel spectrogram by mel-scale filter bank decreases the complexity of the features, thereby reducing the audio feature to an appropriate size more convenient for analysis.

3. *Chromagram*

Chromagram is the mapping of spectral audio information into one octave. Chromagram consists of chroma filters, which project energy of the recorded sound into 12 bins that represent pitch classes. By taking the dot product of the Fourier-based spectrogram (in our case, the mel-frequency spectrogram) we can map audio onto a set of pitches. Because some characteristics of sound are partcularly relevant to pitches, such as the biological sex of the speaker and some emotions, such preprocessing is especially useful for our analysis.

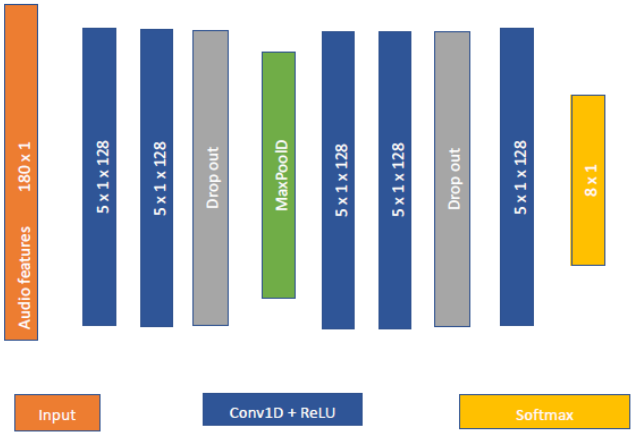
The aforementioned preprocessing is all completed with the librosa library: it extracts the Short-Time Fourier transform (STFT) from the audio data, and computes the chromagram from the waveform of the audio.

Neural Network Model Architecture

We built a Convolutional Neural Network as the model to detect the emotion from audio. There are 17 layers in the model, including 1D convolution layer with 5x5 kernel size and activation using Rectified Linear Unit (ReLU).

After two convolution layers, we added a dropout layer and pooled the convolution layer to decrease training time. Finally, the model will output the probability of each of the 8 types of emotions. We then used softmax to determine the mostly likely emotional state in which the user is.

We used Stochastic Gradient descent (SGD) as the optimizer, setting the learning rate and momentum as 0.001 and 0.09 respectively. Besides, the loss function is Sparse Categorical Cross Entropy, which is used in multi-class classification. Each epoch, when the model gets the highest accuracy ever from the validation set, we will store the checkpoint. As long as we need to use the model to predict one's emotion, we can load the model and checkpoint much faster.



Choosing a GIF

We chose 40 GIFs to be the output of AIPet, and they initially all have the same weight 1/40. In each usage of AIPet, the GIF with the highest weight is chosen to display; since all the GIFs have the same weight initially, the first displayed GIF is chosen as random. As the user continues to use AIPet, we compare the emotions predicted by the model before and after the chosen GIF was displayed, and update the weights on GIFs according to how they change the user's emotion. The emotions are ranked by their degree of positivity, specifically Happy > Surprised > Calm > Neutral > Angry > Disgusted > Sad > Fearful. With each usage of AIPet, the weight of the displayed GIF is updated by 1.5 times the difference between the ranks of emotion of the user before and after seeing the GIF. As a result, the GIFs most suited to improve a particular user's emotion would iteratively obtain higher weights; conversely, GIFs that do

not improve or worsen the user's emotions would obtain lower weights.

Interface

We used the tkinter module in python as the front-end module; tools such as frames to divide multiple blocks, text strips, buttons for recording audio files were particularly useful. We also used this model to display GIFs.

1. *Recording User's Voice*

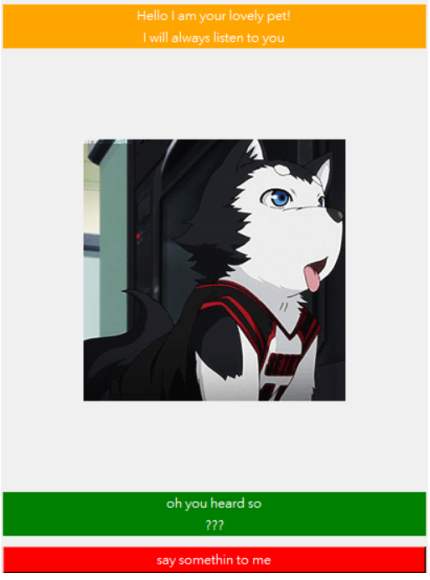
Create a button in a block, when clicked, call the module for recording. We used a sound device to record for 3 seconds, and then save the data as a .wav file. It should be noted that the file is stereo, but the training model uses mono sound. So we needed to delete a channel in the file. Since the two channels' signals are the same, it doesn't matter which channel is taken out.

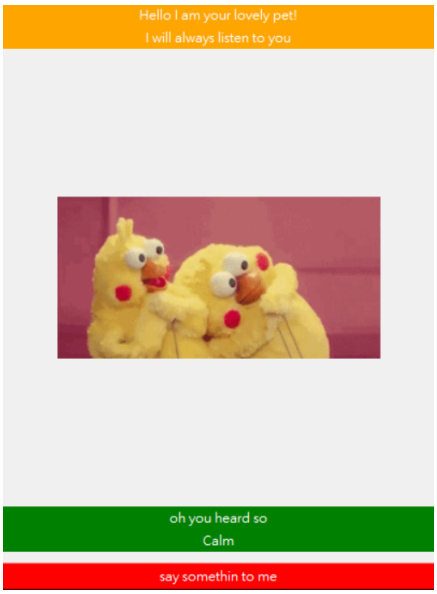
2. *Displaying GIFs*

In this module, pictures are displayed statically. To display the GIF, we need to use the iterator to take out the frames one by one and to refresh the window. Then set a delay of 0.1 seconds in the middle to achieve the animation effect. After each run, the model checks whether the displayed GIF needs to be updated. The second model would then determine the best GIF to display based on the user's feedback. If this GIF is different from the one currently being displayed, then the displayed GIF would be replaced by the most appropriate GIF.

3. *Show Current Emotion*

The text in this module is printed on the window with a label, but the text cannot be changed after setting. To solve this problem, we used the StringVar () to set the variable of the text. It causes that after setting the text we want to display, we can rewrite the variable at any time. So, when the emotion of the user is changed, we can easily change the text to the returned emotion predicted from the model to show in the window.





Results

Our emotion detection model achieved a performance of about 70% in accuracy. We tested two optimizers to see whether it affects the overall performance. Figure 2 shows that, when using SGD as optimizer, the model overfitted after around 600 epochs. Figure 4 shows that the model overfitted after 800 epochs with Adam as the optimizer. However, the overall performance of both models converges to approximately 70%. We decided to use the model with less overfitting during training, with parameters at minimal loss, as our final model.

Optimizer	Accuracy	F1 score
Adam	68.85%	0.64
SGD	70.14%	0.65

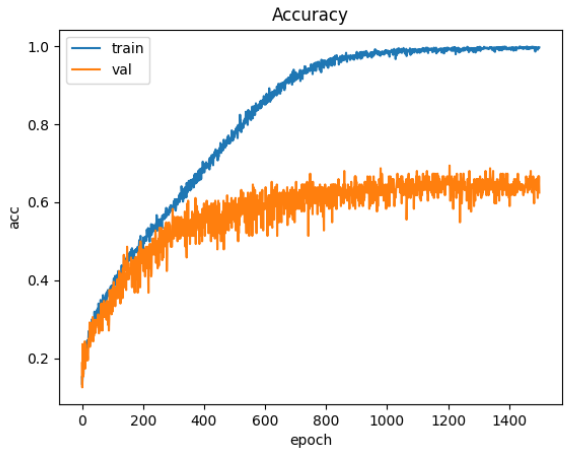


Figure 1: Accuracy with SGD

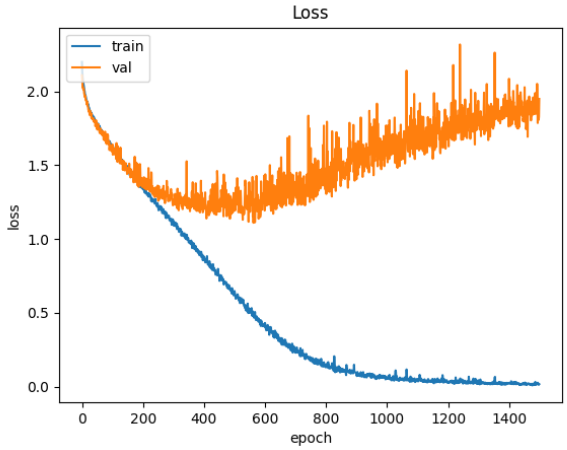


Figure 2: Loss with SGD

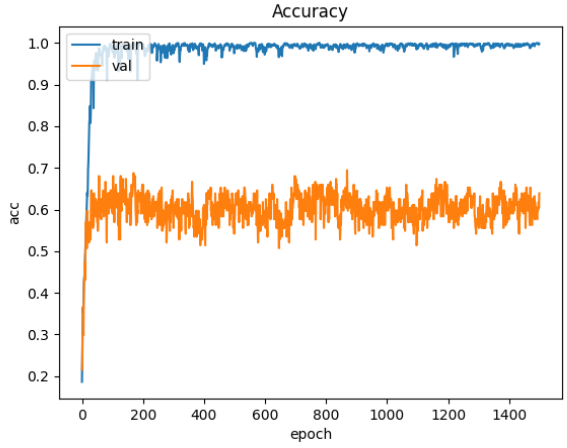


Figure 3: Accuracy with Adam

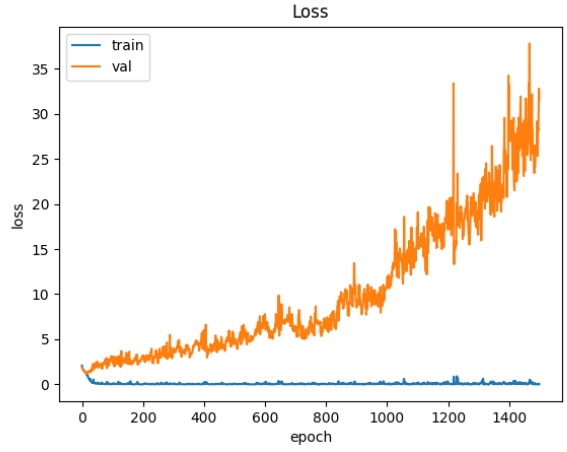


Figure 4: Loss with Adam

Discussion

The first problem we encountered was the interactive screen. Originally we wanted to use HTML, JavaScript, CSS as the front-end and python as the back-end, but we have always encountered many bugs in the transmission of wav files. The traditional web pages are matched with PHP and database, so we gave up this method. Then we used python's existing Tkinter module to implement functions such as interactive, recording and buttons, which can achieve the effect of display and interaction more smoothly and efficiently.

The second problem was building a general model and integrating a front-end and back-end model. It's hard for us to build a complete and general model. We must always pay attention to every detail of the program. And we also expected to execute screen and model in parallel, but it took too much time and was hard to implement.

The last problem was finding data sets. Many websites require users to register a new account, it brought us a lot of restrictions and inconveniences.

Conclusion & Future Work

We have created an AIPet that can interact with the user. The AIPet will recognize the emotions expressed by the user and give an appropriate response. However, the accuracy of the AIPet's feedback is only about 70 percent. In the future we may be able to increase the number of datasets or adjust the parameters and hyperparameters of the AI model, or even apply another model that is more suitable for distinguishing emotions.

In terms of interactive screen, we output AIPets locally. In the future, we hope that AIPets can be presented on the Internet through web pages. That way, anyone who wants to interact with an AIPet is always welcome.

And in the animation of AIPets, we use many different GIF files to convey the reaction of AIPets. We think that we can add hand-painted creation to replace the GIF files to make the animation cuter and more delicate.

In addition, originally we limit users to recording their voice for three seconds, but we can extend the input time, which will allow users to have more time to express their inner thoughts. However this must add more longer data sets when training the model.

References

[1] A. Adigwe, N. Tits, K. E. Haddad, S. Ostadabbas, T. Dutoit, "The Emotional Voices Database: Towards Controlling the Emotion Dimension in Voice Generation Systems" 2018 SLSP

[2] Renovamen.. *Renovamen/speech-emotion-recognition: Speech emotion recognition implemented in Keras (LSTM, CNN, SVM, MLP): 语音情感识别*. GitHub. Retrieved January 20, 2022, from https://github.com/Renovamen/Speech-Emotion-Recognition

[3] *Python 監測檔案是否更新*. Python 監測檔案是否更新 - IT閱讀. (n.d.). Retrieved January 20, 2022, from https://www.itread01.com/content/1546905268.html

[4] 為應用程式設計圖形化介面，使用Python tkinter 模組. (n.d.). Retrieved January 20, 2022, from https://www.rs-online.com/designspark/python-tkinter-cn

Author Contribution Statements

L.-C. C. (107080006) (20%) first model implementation, Model integration, Performance examination, writing

P.-J. C. (108070025) (20%) voice database, model searching and testing, framework design, writing

C.-Y. H. (107070038) (20%) front-end design and programming, framework design, writing

Y.-J. J. (107080072) (20%) framework design, second model implementation, writing

C.-Y. K. (108062303) (20%) picture database, voice database, model searching and testing, writing