Audio Processing and Indexing
Machine learning workshop

Wei Chen
s3156486

1 Assignment 1

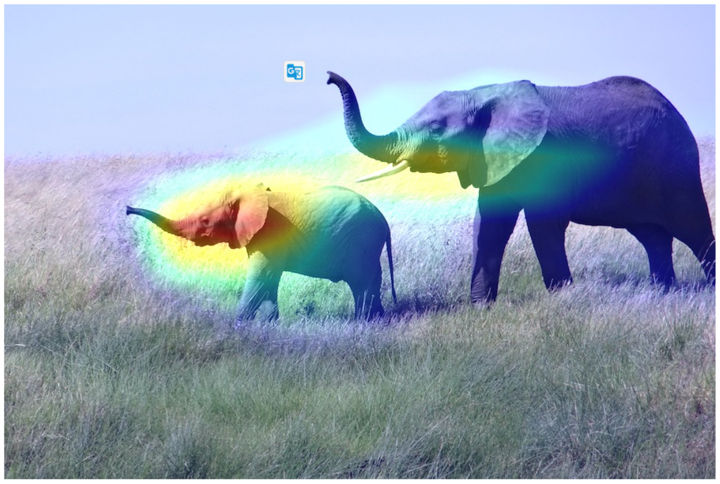

Figure 1: Image that superimposes the original image

(a) Test accuracy is **0.9921000003814697**

(b) For test dataset, **loss**: 0.4467 **- accuracy**: 0.8000

(c) For the training of dense classifier network, **loss**: 0.2581 **- accuracy**: 0.8870; For the training of fine tune model, **loss**: 0.3698 **- accuracy**: 0.9330. Further, to avoid graph errors occurring, batch size for some experiments is reduced to 8.

(d) The resulting image that superimposes the original image with the heatmap is shown as figure 1.

2 Assignment 2

(a) Table 1 below describes the prediction results of speech recognition of digits raging from 0 to 9. It can bee seen that 6 out of 10 numbers were successfully predicted. This spectrogram-based classifier proved to be useful. It is, however, far from performing well.

(b) Function *graph_spectrogram* takes as input the audio files (.wav) and outputs corresponding images. Essentially, it converts audio to spectrogram, which visualizes sound by representing its signal over time. Personally, I borrow my ideas from this as well. Specifically, sound signal is a one-dimensional signal, and intuitively only characteristics of frequency domain could be easily observed, not that of time domain (tend to be invisible). For this reason, FT, an acronym to Fourier transform is applied to transform the representation of audio from time domain into frequency domain. Some excellent libraries, e.g., librosa, can be used to do so. Next, for images generation to have better performance than the function given by this Jupyter notebook, Mel Frequency Cepstral Coefficients (MFCCs) spectrograms is calculated to obtain more features relative to classification problem. Once MFCCs are computed, they are brought to CNNs model for training. A potentially effective method that converts an audio file into an image is shown as follows, in pseudo-code form (See algorithm b).

Confidence	Tested Digit									
Prediction	0	1	2	3	4	5	6	7	8	9
Class 0	20	0	2	0	0	0	0	0	0	0
Class 1	5	69	0	0	12	17	0	0	0	29
Class 2	1	2	73	0	0	0	40	40	1	0
Class 3	4	0	15	27	0	0	3	0	0	0
Class 4	0	0	0	0	83	0	0	0	0	0
Class 5	0	2	0	0	0	71	5	0	0	0
Class 6	13	0	9	51	1	0	8	3	98	0
Class 7	22	1	0	0	2	0	42	56	0	0
Class 8	0	0	0	16	1	0	0	0	0	0
Class 9	35	25	0	6	0	11	2	0	0	71

Table 1: Predicted results for audio classification of numbers 0-9, where each cell stands for confidence

Algorithm b Convert an audio file into an image

Input: *AUDIO*: An audio file

Output: *IMAGE*: An image generated from an array, which is computed according to MFCC

```
1: function CONVERT
2:     signal, sample_rate := loadAudioFile(AUDIO);          ▷ signal represents the floating point time series
3:     mfcc_features := calculateMFCC(signal, sample_rate);
4:     mfcc_features_scaled := mean(mfcc_features);          ▷ mean is an aggregate function
5:     IMAGE := arrayToImage(mfcc_features_scaled);
6:     return IMAGE;
7: end function
```