Hiding Emotions in plain sight: Adversarial approach to ensure privacy

Lahiru Wijayasingha (LNW8PX)

1 Motivation

With the prevalence of intelligent personal digital assistance devices like amazon Alexa, Siri and Google home, Privacy has become a greater concern. This is because these devices listen to the user (and other people around them) and process these vocal data and make inferences. For example these devices may learn your identity and emotional state. Some researchers are calling for a ban for emotion recognition technologies because they can be biased (e.g racially) [5]. Also it may listen to confidential and private conversations you are having. The threat becomes greater since these devices usually send the data (voice recordings) to cloud for further analysis. Refer Figure 1 for further a diagram of a simple example digital assistant. Some malicious agent may get hold of these data. Therefore it is important to investigate methods to mask certain attributes out from voice. For example, if we can hide the emotional content from voice, and if we can still use other voice functions (such as speech recognition), it would be beneficial.

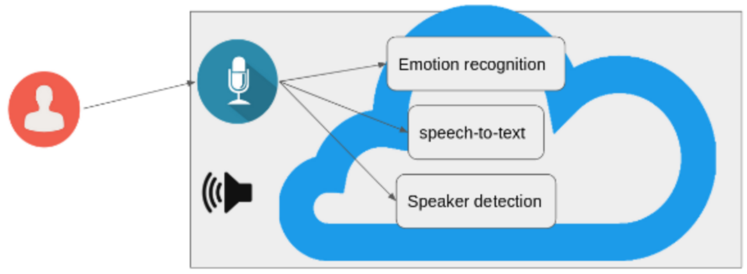


Figure 1: Basic operation of a smart digital assistant

2 Background

2.1 How modern emotion recognition work

Convolutional Neural Networks (CNN) which is a type of Deep Neural Networks (DNN) have become the state-of-the-art of emotion recognition from voice [2]. CNNs are initially designed for image classification tasks. Researchers have adapted them for vocal emotion recognition. This is done by obtaining the voice spectrogram using Fast Fourier Transformation (FFT) [2][6]. Since spectrogram can be represented as an image, we can use the same methods of using CNN for image classification to classify vocal emotions. Refer Figure 2 for an overview of the inference process of a CNN for voice data classification.

2.2 Adversarial samples

Adversarial samples in the context of CNN is adding small (ideally unnoticeable to humans) values to the input of a CNN so that the CNN will make a mistake. If we can leverage this method to modify

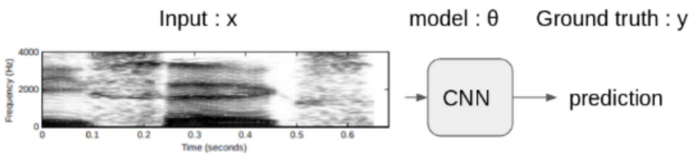


Figure 2: Basic operation of a smart digital assistant

our voice we may be able to fool an emotion recognition model. Several methods of performing these are mentioned in [3]. The main idea is to modify the input image so that the prediction loss of that image is increased. Please refer Figure 2 for related notation. The method to modify the input x is called Fast Gradient Sign Method (FGSM) and is as shown in equation below where L is the loss function.

$$x + \epsilon sign(\nabla_x L(\theta, x, y)) \tag{1}$$

We can also apply this equation repeatedly on the same input. This is represented as

$$x^{t+1} = x^t + \alpha sign(\nabla_x L(\theta, x, y)) \tag{2}$$

In this study the samples are generated according to equation (2). The number of iterations performed is a hyperparameter.
In this study, the spectrograms are the immediate inputs to the CNN. Therefore, spectrograms are the data which is modified according to equation 2. But modifying spectrograms is not very practical as can be seen from the next section.

3 Methods and Results

3.1 Modifying vocal data with FGSM

In modern personal assistants like Alexa, the voice is directly sent to the cloud to make inferences. Therefore we will not have access to the spectrograms generated. So we will have to add perturbations directly to the raw voice signal directly after the microphone. But we have to perform additional transformations to adapt equation (1) to perform this. The method to perform this transformation is shown in appendix. This is one of the main contributions to the research. But since other functions (e.g. speech-to-text) depend on the same voice signal, these functions may get affected by modifying voice directly. Ideally, after adding perturbations to the voice signal, only the emotion recognition model should be affected and other systems should be affected as less as possible. These issues will be analyzed in this study. When we create adversarial samples in the raw voice data domain, we can use the equation

$$r + \epsilon sign(\nabla_r L(\theta, x, y)) \tag{3}$$

Where r is the raw voice signal.

3.2 Which target to use when calculating loss

When calculating adversarial samples by using equation (1),(2) or (3) we can see that we have to use the true label (y) of the sample. This is not always practical. For example, if we try to hide our emotion from a model, we have to always know what our emotion is and input this to the FGSM algorithm. This does not work because we do not have labeled data of our own voice. But instead we can use the predicted emotion (from the same model) as a proxy for this. Effects of doing this is analyzed in this study.

3.3 Assumptions

In this study we assume that we have a copy of the model that is used to classify emotions. This can be a realistic assumption because various personal digital assistant service providers might publish the emotional model they use.

3.4 Experiment Methods

In this study, various steps are performed to evaluate the effectiveness of adversarial examples generated by multi step FGSM on emotion recognition model. The steps performed are,

1. Train a CNN based emotion recognition model.
2. Evaluate performance of this model on adversarial spectrograms (using ground truth labels)
3. Evaluate performance on adversarial spectrograms (using predictions from the copy of the model we have as a proxy to ground truth labels)
4. Transform FGSM so that we can generate adversarial examples in the raw voice data domain
5. Evaluate the performance of the model on adversarial examples generated by the transformed method
6. Evaluate the performance of speech-to-text system on the adversarial data.

3.5 Training the CNN

Dataset used : Berlin Emo DB
Emotions present : anger,boredom,disgust,anxiety/fear,happiness,sadness
CNN model chosen : Modified model from [1]
Results of training :
Train accuracy: 0.79 train loss : 0.7048
Validation accuracy : 0.60 validation loss: 1.4531
See the appendix for details about training hyperpapramaters and model architecture.

3.6 Evaluate performance of the trained CNN on adversarial spectrograms (using ground truth labels)

In the first two experiments I assumed that we can intercept the voice signal just before going into the emotion recognition model (see Figure 3). As the first experiment, I created the adversarial spectrograms with FGSM. By looking at equation (1) we can see that we need ground truth labels to calculate the perturbations. I use the labels from the dataset for this purpose. The performance of the model for different number of FGSM steps are plotted in Figure 4.

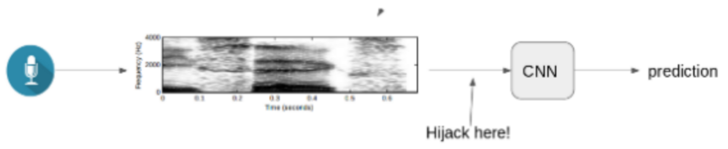


Figure 3: Intercept signal just before the model

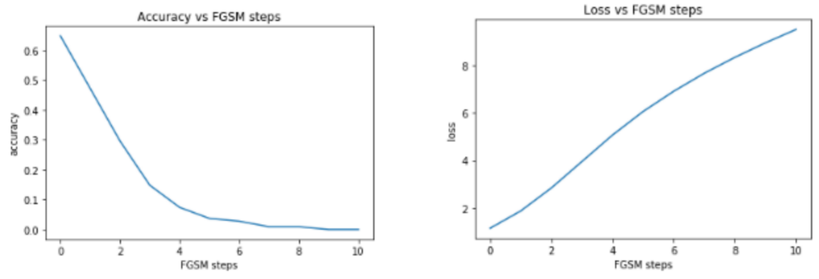


Figure 4: Accuracy and loss on test set vs number of FGSM steps

3.7 Evaluate performance on adversarial spectrograms (using predictions from the copy of the model we have as a proxy to ground truth labels)

The only difference between this section and the previous one is now we use the predictions from our copy of the model as a proxy for groud truth labels when calculating the FGSM adversarial samples. This is more realistic since we cannot always have ground truth labels. Assume we are talking to Alexa and we need to hide out emotional content. To create adversarial examples from this data, we need to perform FGSM on these data. But to do so, we need the true labels. But we do not have them in the real world scenario. As a workaround we can first sent unmodified samples through the model and obtain it's predictions. Then we can use these as a proxy for the ground truth labels. The results in Figure 5 show the performance of the CNN model after performing this step. Note that this is a more realistic scenario than the previous section.

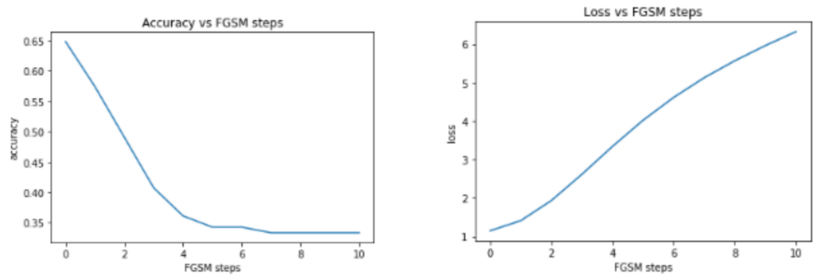


Figure 5: Accuracy and loss on test set vs number of FGSM steps

Note that in figure 4, the accuracy reached zero. But on the figure 5, it stops decreasing at around 0.33. This is because the situation in Figure 5 assumes more realistic assumptions (we do not have ground truth labels).

3.8 Transform FGSM so that we can generate adversarial examples in the raw voice data domain

In previous 2 sections we assumed that we can intercept the voice signal as shown in figure 3. But this is unrealistic because most of the digital assistants like Alexa will send the raw audio signal to cloud to make inferences. It will never modify it locally. Therefor we will not have access to spectrograms/audio features locally. Therefore it is more realistic to modify the signal in the raw voice data domain as shown in Figure 6. This section analyzes that situation. But we have to transform the equation (1) so that it can modify the samples in the raw voice data domain. The proof of this and the methods are shown in appendix. After performing this transformation of equation (1) we can use this to directly generate adversarial samples in the raw voice domain. We can use the voice signal just after the microphone as shown in Figure 6.

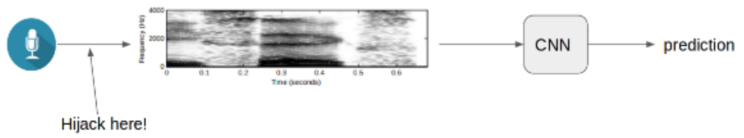
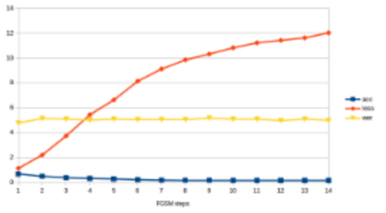


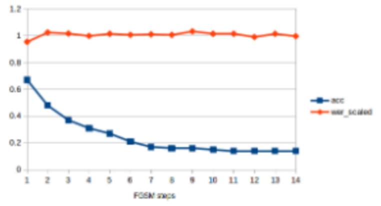
Figure 6: Intercept signal just before the model

3.9 Evaluate the performance of the model on adversarial examples generated by the transformed method and on speech-to-text system

After modifying voice data to create adversarial examples we have to analyze how the performance of model changes according to the number of FGSM steps. But by looking at Figure 1, we can see that if we modify the signal just after the microphone, it will affect all the systems which uses this voice; not just the emotion recognition system. If we want to hide our emotional state from the system but



(a) accuracy,loss of emotion recognition model and WER of speech recognition model



(b) accuracy of emotion recognition model and scaled (for clarity) WER of speech recognition system

Figure 7: Accuracy and loss of emotion recognition model and accuracy of speech recognition system (WER) vs the number of FGSM steps

still be able to use other functionalities like speech-to-text, evaluating only the performance of the emotion recognition system is not enough. We should also evaluate the performance of other systems (e.g. speech-to-text). So in this section we analyze the performance of emotion recognition model and a speech recognition model on adversarial data. I chose the speechRecognition library on python and used google API which is included in it to convert speech to text. Refer Figure for the results.

3.9.1 Evaluating the performance of speech-to-text systems

Since the emotional voice cips are in German language, I had to use the German version of google speech recognition API. After obtaining the text from a certain voice clip (with speech-to-text), we have to evaluate how accurate that is. To do this Word Error Rate (WER) is used. This basically calculates how much error is included in the result text when compared to the original ground truth text. The method of this calculation is performed as shown in [4]. Ground truth can be obtained by the dataset website.

From Figure 7, it can be seen that WER does not go down as the performance of the emotion recognition model degrades. This means we can generate samples which can reduce the performance of emotion recognition model (hide emotions) at the same time not affecting speech-to-text system.

4 Contributions and Conclusions

In this research, we have shown that adversarial samples generated by multi step FGSM can degrade the performance of state-of-the-art vocal emotion recognition model. But with existing methods, we can only create adversarial samples in the immediate output to the CNN (spectrograms in this case). I extended this so we can directly create adversarial samples in the voice data domain. This is necessary because in modern smart digital assistants we will not have access to intermediate audio features (e.g. spectrograms) since they send voice clips directly to cloud to make inferences. We have shown that creating adversarial samples in the raw voice data domain can degrade the performance of CNN emotion recognition model. But it is important for these modified data to be useful for other functions like speech-to-text. I evaluated the performance of speech-to-text (with Google API) to evaluate this. It can be seen that even when the performance of emotion recognition model degrades, the performance of speech-to-text (which uses the same modified data) does not change. Therefore it can be concluded that this new method of creating adversarial voice data can use used to hide our vocal emotions. But we will still be able to operate other functions like speech-to-text.

5 References

[1] Haytham M. Fayek, Margaret Lech, and Lawrence Cavedon. “Evaluating deep learning architectures for Speech Emotion Recognition”. In: *Neural Networks* 92 (2017), pp. 60–68. ISSN: 18792782. DOI: 10.1016/j.neunet.2017.02.013.

[2] Zhengwei Huang et al. “Speech emotion recognition using CNN”. In: *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*. ACM, 2014, pp. 801–804. ISBN: 9781450330633. DOI: 10.1145/2647868.2654984.

[3] Aleksander Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).

[4] Martin Thoma. *Word Error Rate Calculation.* 2013. URL: https://martin-thoma.com/word-error-rate-calculation/ (visited on 12/01/2019).

[5] Whitney KImball. *Emotion Recognition Is Creepy As Hell.* 2019. URL: https://gizmodo.com/emotion-recognition-is-creepy-as-hell-1840415094 (visited on 12/01/2019).

[6] Jianfeng Zhao, Xia Mao, and Lijiang Chen. “Speech emotion recognition using deep 1D & 2D CNN LSTM networks”. In: *Biomedical Signal Processing and Control* 47 (2019), pp. 312–323. ISSN: 17468108. DOI: 10.1016/j.bspc.2018.08.035.

6 Appendix

6.1 Generating adversarial samples in raw voice data domain

Let r be an array of time-seris voice samples. Assume we take k number of samples for a time window. This window is considered for classification. r can be represented as
$r = [v(t_0), v(t_1), ..., v(t_{k-1})]$
Here x represents the amplitude of voice and t is time
Let X be the spectrogram obtained by performing Fast Fourier Transformation (FFT) on r. Imagine, that depending on the number of windows and the stride of the FFT, we end up with m frequency bands and n time windows. So the matrix X is of sie $m * n$. This procedure can be represented as below
$r_{1*k} --(window) --> r'_{1*n} --> (fft) --> X_{m*n} --> (cnn) --> predictions$
Earlier with equation (1) we obtained
$\nabla_x L(\theta, x, y)$
Here we take the derivative of L with respective to X because X is the input. But now if we want to modify the vocal signal itself, we have to take the derivative with respect to r and then take
$r + \epsilon sign(\nabla_r L(\theta, r, y))$
$= r + \epsilon sign(\nabla_r L(\theta, x, y))$

So let’s find $\nabla_r L(\theta, x, y)$

From the chain rule,

$$\frac{\partial L}{\partial r'} = \frac{\partial L}{\partial X} * \frac{\partial X}{\partial t} * \frac{\partial t}{\partial r'} \tag{4}$$

$$\frac{\partial L}{\partial r} = scale(\frac{\partial L}{\partial r'}) \tag{5}$$

where t is time.
Here $\frac{\partial L}{\partial X}$ can easily be taken by the deep learning platform (I used keras).
$\frac{\partial t}{\partial r'} = \frac{1}{\frac{\partial t}{\partial r'}}$ can be found by simply taking the differences of the scaled data array r’ which has a size of [1*n] (after scaling down from the initial array r which had size of [1*k])
The tricky p art is finding $\frac{\partial X}{\partial t}$. This can be found analytically by considering the properties of Fourier transformation.
The Fourier transformation of a function $r(t)$ is defined as

$$X_j(w_l) = \int_{window_j} r(t) e^{-2\pi i w_l t} dt \tag{6}$$

Where w_l is the l^{th} frequency component. And $window_j$ is the set of samples that belong to the j^{th} FFT window. Since we are working on the discrete domain, this can also be written as a summation Now,

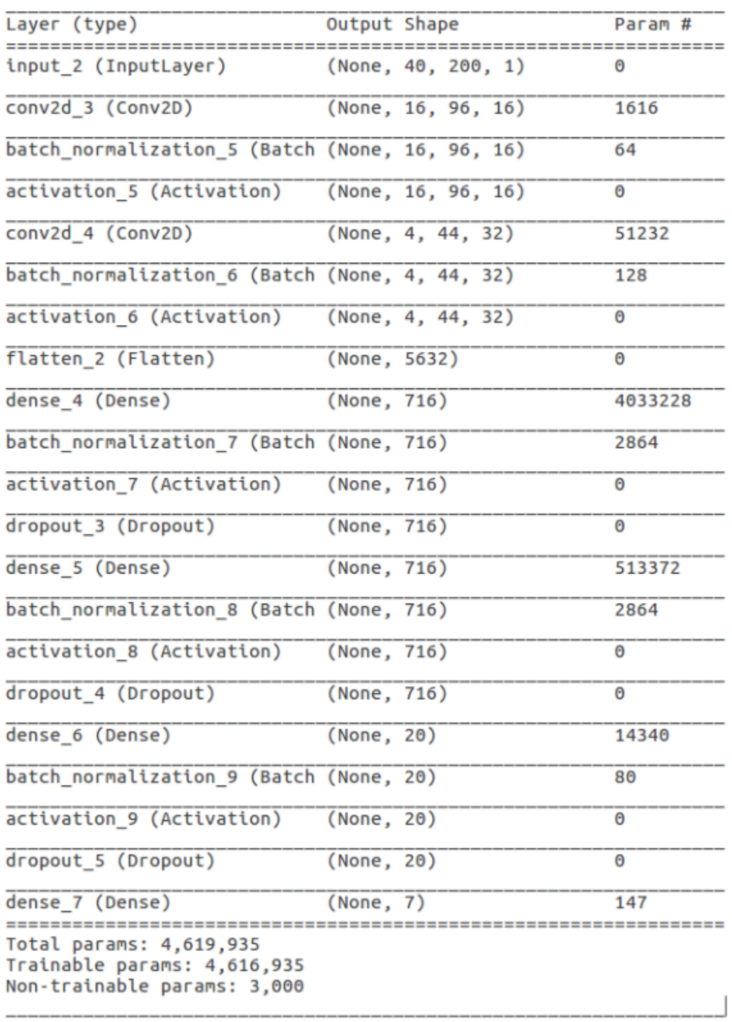
$$\frac{\partial X_j(w_l)}{\partial t} = r(t) e^{-2\pi i w_l t} \tag{7}$$

Likewise we can define the matrix $\frac{\partial X(w_l)}{\partial t}$ Where the element (l,j) of the matrix $\frac{\partial X(w_l)}{\partial t}$ can be represented as
$r'_j(t) * e^{-2\pi i w_l t}$
Here r' is used instead of r because we have to scale r to match the windowing in FFT. r'_j is taken as the mean value of the j^{th} window over the original voice data array r.
Now we can find $\frac{\partial L}{\partial r}$ and use this to find $r + \epsilon sign(\nabla_r L(\theta, x, y))$ in order to create adversarial examples in the audio signal domain.

6.2 CNN model train details

Deep learning library : keras
optimizer : RMSprop with initial leaning rate=0.001, rho=0.9
reduce learning rate when validation loss plateaus by a factor of 0.1. Patience=50
loss = categorical crossentropy
batch size = 128
train data = 0.6 of the whole dataset
validation and test data : each 0.2 of the whole dataset

6.3 CNN model architecture



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         (None, 40, 200, 1)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 96, 16)        1616      
_________________________________________________________________
batch_normalization_5 (Batch (None, 16, 96, 16)        64        
_________________________________________________________________
activation_5 (Activation)    (None, 16, 96, 16)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 44, 32)         51232     
_________________________________________________________________
batch_normalization_6 (Batch (None, 4, 44, 32)         128       
_________________________________________________________________
activation_6 (Activation)    (None, 4, 44, 32)         0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 5632)              0         
_________________________________________________________________
dense_4 (Dense)              (None, 716)               4033228   
_________________________________________________________________
batch_normalization_7 (Batch (None, 716)               2864      
_________________________________________________________________
activation_7 (Activation)    (None, 716)               0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 716)               0         
_________________________________________________________________
dense_5 (Dense)              (None, 716)               513372    
_________________________________________________________________
batch_normalization_8 (Batch (None, 716)               2864      
_________________________________________________________________
activation_8 (Activation)    (None, 716)               0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 716)               0         
_________________________________________________________________
dense_6 (Dense)              (None, 20)                14340     
_________________________________________________________________
batch_normalization_9 (Batch (None, 20)                80        
_________________________________________________________________
activation_9 (Activation)    (None, 20)                0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 20)                0         
_________________________________________________________________
dense_7 (Dense)              (None, 7)                 147       
=================================================================
Total params: 4,619,935
Trainable params: 4,616,935
Non-trainable params: 3,000
_________________________________________________________________
```