Final Report – Speech Detection and Censoring profanity in speech.

Aryaman Dhawan
The University of Adelaide
aryaman.dhawan@student.adelaide.edu.au

Abstract

The problem at hand is of developing a novel machine learning model to detect spoken words and censor curse words in audio data containing human speech. This report summarizes the methodology used, experimental setup and the outcomes of proposed approach.

1. Introduction

Currently with the easy access to unregulated audio and video content over the internet and in various TV shows, the major problem is rise in usage of profanity. There exist no regulations or easy to bypass limitations as to what content is accessible to children [1]. I propose to develop a fast and robust application leveraging the power of current Machine Learning methods to detect and censor profanity in real time audio and video content. The approach used in this report involves extracting spectrograms of spoken bad words and implementing a supervised image classification on these spectrograms to train a ResNET18 model using transfer learning.

2 Discussion of previous research.

There have been quite a few challenges and interesting problems in speech recognition and classification in the recent years and a variety of methods proposed for different tasks.

Deep learning-based speech recognition has lately gained prominence. Speech recognition systems worked by recognizing different varieties and instances of speech of the same words [2]. However, most of the research of profanity detection and censoring is done by extracting transcript of speech data from videos and then training a neural network using Natural Language Processing (NLP) and LSTM's [3]. But the accuracy of these methods is not sound and cannot be guaranteed as context of usage of these curse words might not be clear and may change from one language to another. There has also been a separate approach proposed using Word2Vec [4], GloVe [5], and FastText for word embedding and then using LSTM to train a model on textual data. Which also relies on conversion of audio to textual data rather than directly classifying audio data.[6]

I devised a method to develop a model that processes and learns directly from curse word audio samples without having to utilize Automatic Speech recognition and Natural Language Processing to classify profane words by training the model on linguistics and using Natural Language Processing. This ensures a faster training method with lesser training parameters and is a novel approach.

3. Methodology

Over the years there have been various neural networks architecture and modeled for deep learning and have been refined to tackle a particular task at hand. When modelling the second task (model training), transfer learning is an optimization that allows for faster development or better performance. There have been various researches that highlight the positive impact of using transfer learning for similar tasks [7]. This project involves the following steps.

3.1 Data Generation and labelling.

The biggest challenge for me was finding a dataset containing labelled data for profane and clean audio of only single word utterances. This was not easy as there is no such dataset readily available for this task and the ones discussed in research are proprietary. For solving this, I had to generate my own dataset keeping in mind the requirement to ensure diversity of the data.

I engineered a dataset is by merging two audio datasets to create a Training and Testing datasets namely-

1. Profane words (This data consists utterances of bad words in a variety of English accents).[8]

2. Clean words (This data consists of human speech commands in English).[9]

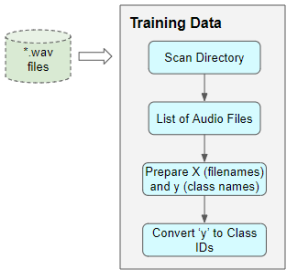

Figure 1 – Dataset generation.

Next, the dataset is split into 2 directories, test and train and simultaneously test_df and train_df datarames are created with the names of the audio files and their class labels (Figure 1.2). We label the data for all filenames from the profanity database, 1 and for all files names from clean words dataset is assigned a label 0. While generating the dataset directories we also rename the files to list which directory they belong to eg “test/abc.wav”. Further two data objects are created -

1. An object that uses audio transforms to pre-process an audio file and prepares one data item at a time.
2. A Data loader object that uses the dataset object to get data items and creates a batch of data.

	file_name	target
0	train/forward.wav	0
1	train/two.wav	0
2	train/off.wav	0
3	train/cat.wav	0
4	train/cat.wav	0

Figure 1.2 Training dataframe

3.2 Feature Extraction

Data transformation is applied on the audio data to help the model learn to generalize to a wider range of inputs by transforming the input data into visual representation of the spectrum of frequencies of the audio signal. Spectrogram represents time, frequency, and amplitude of the audio wave all on one graph. The method adopted in this project uses spectrogram analysis of audio data to train a neural network model using transfer learning for faster analysis and classification of spectrogram images and hence audio samples. Clever approaches include - we first resample the audio to 8KHz and convert all audio to mono from stereo for normalization of data. Then we extract melspectrograms of the audio files by using 512 fourier transforms and 32 filterbanks.

3.2.1 Spectrogram

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time.

3.2.2 MFCC Spectrogram (Mel-frequency cepstral coefficients)

From Wikipedia, MFCC’s are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal spectrum. This frequency warping can allow for better representation of human sound [10].

MFCC is derived as follows –

1. Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows or alternatively, cosine overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

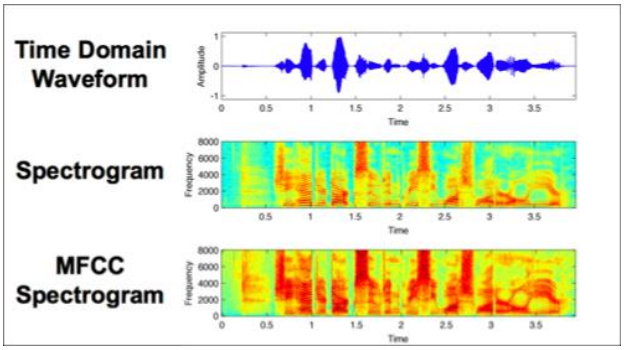

Figure 2. Time Domain waveform of audio signal and corresponding Spectrogram and MFCC Spectrogram

3.2.3 Mel Frequency vs MFCC

MFCC (Mel Frequency Cepstral Coefficients) are used since they provide a compressed representation of the frequency bands from the Mel Spectrogram that correspond to the most common frequencies at which humans speak. Another reason to select MFCC is because for some audio that had shape (128, 134), the MFCC has shape (20, 134) hence lesser data to train on most important features. The MFCC extracts a much smaller set of features from the audio that are the most relevant in capturing the essential quality of the Human sound.

3.2.4 Spectrogram Augmentation

Since normal transforms used on an image don't apply to spectrograms as a horizontal flip or a rotation would substantially alter the spectrogram and the sound that it represents. Using SpecAugment is proposed in which sections of the spectrogram are blocked out.

1. Frequency mask — Randomly mask out a range of consecutive frequencies by adding horizontal bars on the spectrogram.
2. Time mask — Randomly block out ranges of time from the spectrogram by using vertical bars.

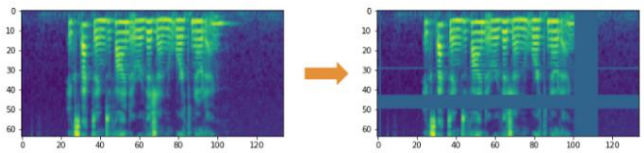

Figure 3. SpecAugment using Time and Frequency Mask

I used 512 fourier transforms and 32 filterbanks to extract the mel spectrograms from the audio data after normalizing all files by respampling them to be 8KHz and converted to mono from stereo as discussed on PyTorch website [19].

3.3 Classification

The data is converted into image form i.e. MFCC Spectrogram and the problem at hand has been reduced to a Supervised Learning Binary Classification task.

For classification of images, transfer learning is used from ResNET18.

3.3.1 Transfer Learning

A pre-trained model is utilized to tackle a problem of the same genre in Transfer Learning. The early and middle layers are used in transfer learning among the layers. And the latter layers are only used to retrain, which helps to save the labelled data of a specific job that the model was trained on [11]. In this research, the reasons for using transfer learning are that it saves training time and improves performance. Furthermore, there is no need to employ a large amount of data in transfer learning [11].

The benefits of using transfer learning are –

1. Saving training time
2. Better performance of model on the training data due to limited samples available for swear words as discussed in the previous reports.

3.3.2 ResNET

A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between. An additional weight matrix may be used to learn the skip weights; these models are known as HighwayNets. Models with several parallel skips are referred to as DenseNets. In the context of residual neural networks, a non-residual network may be described as a plain network. [12]

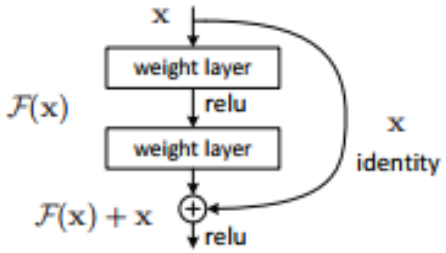

Figure 4. Logical scheme of base building block for ResNet-18

The building block defined as:

$$y = F(x, \{W_i\}) + x. \quad (1)$$

Here x and y are the input and output vectors of the layers considered. The function F(x, {Wi}) represents the residual mapping to be learned. In Figure 4, $F = W_2\sigma(W_1x)$ in which σ denotes ReLU [13] and the biases are omitted for simplifying notations. The operation F + x is performed by a shortcut connection and element-wise addition. We adopt the second nonlinearity after the addition (i.e., σ(y), see Fig. 4). The shortcut connections in Eqn.(1) introduce neither extra parameter nor computation complexity. This is not only attractive in practice but also important in our comparisons between plain and residual networks. We can fairly compare plain/residual networks that simultaneously have the same number of parameters, depth, width, and computational cost (except for the negligible element-wise addition). The dimensions of x and F must be equal in Eqn.(1). If this is not the case (e.g., when changing the

input/output channels), we can perform a linear projection Ws by the shortcut connections to match the dimensions:

$$y = F(x, \{Wi\}) + Wsx. \quad (2)$$

We can also use a square matrix Ws in Eqn.(1). But we will show by experiments that the identity mapping is sufficient for addressing the degradation problem and is economical, and thus Ws is only used when matching dimensions. If F has only a single layer, Eqn.(1) is similar to a linear layer:
$y = W1x + x$,
Although the above notations are about fully-connected layers for simplicity, they are applicable to convolutional layers. The function F(x, {Wi}) can represent multiple convolutional layers. The element-wise addition is performed on two feature maps, channel by channel. [14]

Layer Name	Output Size	ResNet-18
conv1	112 × 112 × 64	7 × 7, 64, stride 2
conv2_x	56 × 56 × 64	3 × 3 max pool, stride 2
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$
conv3_x	28 × 28 × 128	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$
conv4_x	14 × 14 × 256	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$
conv5_x	7 × 7 × 512	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$
average pool	1 × 1 × 512	7 × 7 average pool
fully connected	1000	512 × 1000 fully connections
softmax	1000	

Figure 5. ResNet-18 Architecture.

3.3.3 Model Selection

According to results presented in Table 1 [14]. It is a clear choice to use ResNET-18 for this classification task as it has the lowest error rate.

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Table 1. Top-1 error (%, 10-crop testing) on ImageNetValidation.[14]

3.3.4 Knowledge Transfer

The primary goal of transfer learning (TL) is to swiftly deploy a model. Instead of developing a DNN (dense neural network) from start to handle the current challenge, the model will transfer the features it has learned from different datasets that have completed the same task. Pytorch API calls a pre-trained model of ResNet18 by using models.resnet18(pretrained=True), the function from TorchVision's model library trained on ImageNet data set [15].

4. Experimental Setup

The summarized approach used in this project involves extracting spectrograms of spoken bad words and implementing a supervised image classification on these spectrograms to train a ResNET18 model using transfer learning by performing steps detailed in Section 3.

After training the model on melspectograms of audio data from training data, model's performance is measured on testing data.

Further, a prediction dataframe is created containing actual and prediction columns. Prediction column is then utilized to record model's classification of the testing sample and the performance on unseen mixed testing data containing swear words and clean words is calculated.

Finally, the trained model is fed with the longform audio. This was a challenging task as we need to now figure out how to detect if there is a word and not noise in the sample and if the word is a swear word from trained data. I devised a methodology to split the audio data into multiple small segments extract the mel spectrograms and get prediction for that segment from the trained model. During testing, it was found that the model is able to detect words better if the files are split into 200 millisecond segments instead of 1 second or 2 second long segments. samples containing profanity one file at a time by splitting the long audio into multiple 200 millisecond files saved as .wav files. We then feed these files one by one to the model and get a classification as 1 containing profanity and 0 no profanity for the file.

If the file is labelled 1, we replace the file with a censor audio sound the same file is its labelled 0. Lastly, we retrieve the censored audio by stitching all the files back together. F1_score from scikit-learn was used for training accuracy metric and F1 Score and Accuracy along with confusion matrix for testing accuracy. The model was trained using 5 fold cross validation.

5. Novelty of Method

For the specific task at hand, to classify English speech data as profane and censor it, there have been no such approach with the same methodology as stated in this report implemented. Traditional methods involve using spectrogram analysis of speech to convert speech to text and then use this text to classify if the text contains profanity or not. This is done by using Automatic Speech

Recognition methods [16].

In my approach we directly sample the audio data and train a neural network on which has a collection of 347 different bad words spoken in 13 different accents.

This offers wide range of curse words spoken in various accents to be directly sampled and train the model for classification. Hence reducing the amount of training required in traditional methods most importantly skipping the requirement to train the network to recognize letters in individual words. There have been however some different approaches proposed -

In paper [17] researchers used SVM to classify audio data into rude and proper using audio recordings of 27 people, this achieved accuracy in ranges of 72% to 95%.

In paper [18] researchers used SVM classifier where they used prosodic, dialog, and lexical features to find a clear style achieving 50% to 75% accuracy.

6. Results and Evaluation Metric

An Evaluation must be adopted to verify results and correctness of the model. Since this is a supervised binary classification problem, simple metrics can be used as we already know what the right answers are. So the model either returns the correct or the incorrect label. The model was trained on 80% of data and then tested on 20% of the data

6.1 Metrics Used -

Accuracy - It is the ratio of number of correct predictions to the total number of input samples.

F1 Score – A harmonic mean of precision and recall.

Confusion Matrix - It is matrix that describes the complete performance of the model. For the binary classification problem, the two classes: 1 or 2. The model predicts a class for the input sample and returns one of the following –

True Positives : The cases in which we predicted 1 and the actual output was also 1.
True Negatives : The cases in which we predicted 2 and the actual output was 2.
False Positives : The cases in which we predicted 1 and the actual output was 2.
False Negatives : The cases in which we predicted 2 and the actual output was 1.

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 6. Metrics formula.

6.2 Results –

This section concludes the results with discussion from all experiments performed in Section 4 - Experimental Setup using steps discussed in Section 3 – Methodology.

6.2.1 Training Results –

F1 score is chosen metrics for testing the model performance on testing data due to imbalance between the classes samples 4511 for swear words and 105835 for clean words. Accuracy might give a biased result in this scenario while F1 score will be reliable.

Model performed well with **F1 Score of 1** on training data which was trained for 10 epochs but converged within 3 epochs, benefit of transfer learning.

6.2.2 Performance on Testing Data

F1 score and Accuracy are chosen metrics for testing the model performance on testing data. Model performed well with **Accuracy of 94.78% and F1 Score of 0.78**. This validates that training on Mel Spectrograms of audio data is correct approach.

Confusion matrix of testing data (Figure 7)

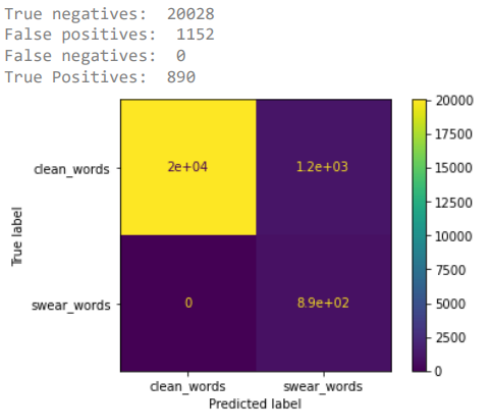

Figure 7. Confusion Matrix on testing data

7. Prediction on long audio clips

I wanted to censor the parts of the audio clip containing utterances of swear words that were listed in the TAPAD dataset. This was a challenging task as I need to now figure out how to first detect if there is a word and not noise in the sample and if the word is a swear word from trained data. Experiment was conducted by recording an audio clip containing profanity from both vocabulary in the training data and unseen words.

7.1 Results of censoring long audio clips

Even though the model performs well on predicting swear words from the TAPAD dataset and SPEECHCOMMANDS dataset. It does not quite work well with unseen audio. It does not completely overlap the censor audio with the curse word and misses out on some curse words. Furthermore, it sometimes mistakes normal words to be curse words as seen in the previous audio. I am not entirely sure why this happens, but I have a few explanations –

1. The audio from the profane and speech commands dataset is limited and exhaustive training could not be done.
2. When converting Mel Spectrograms from the sample audio split files, the model thinks these belong to profane words even though they are not profane words. This can be due to noise in the audio recording and limited vocabulary.

8. Future Scope

In the future I would like to generate new audio samples and long audio clips in a controlled environment and train and test the model on these to further validation. Since most of my time was spent on working with audio data and learning how to handle audio data, I did not have much time to explore latest models from academia. I would like to work with other renowned models as well. I would also like to take another approach in which audio data is converted to text and use Word Embedding and LSTM [6,7] to see how my model performs in comparison.

9. Learning Outcomes

This project was developed incrementally over the duration of the course Applied Machine Learning under guidance of Prof Qi Wu and with help of tutor Mahdi Moghaddam. Coming form a background of software development, this was the first time I have worked hands on Machine Learning. Not only did I learn how to practically train a model but also the core intrinsic of various machine learning methods the under the hood functionality of neural networks and various activation functions that affect the performance of a neural network.

Machine Learning is fun! Working on this project I got a chance to work with number of different libraries such as fastai, torchaudio and ffmpy etc. These are highly optimized libraries for machine learning and while developing the project I learned how to use libraries and extracting the most valuable information from source. Even though some libraries have very nice documentation, some lack and for this I had to read the source code to understand various parameters of different functions and make sense of how to use a particular function.

It is interesting to note how transfer learning can really deliver accurate results with reduced training time which was a completely alien concept to me. Using transfer learning not only reduces training time but also helps save computation cost and hence results in lesser CO_2 emissions.

Reading through various papers and from the lectures , I got a chance to learn about various machine learning alorithms and best practices.

Working with audio data is not easy and in other scenarios the data might not be available or might not be very useful for training. It made me realize the importance of selecting an appropriate dataset and the need of preprocessing the data.

References

[1] Tuttle, K. The Profanity Problem: Swearing in Movies. In Movie Babble 2018. Available online: https://moviebabble.com/2018/02/20/the-profanity-problem-swearing-in-movies/.

[2] Santosh K.Gaikwad, Bharti W.Gawali and Pravin Yannawar. Article:A Review on Speech Recognition Technique. International Journal of Computer Applications 10(3):16–24, November 2010. Published By Foundation of Computer Science.

[3] Bhandary, Unnathi, "Detection of Hate Speech in Videos Using Machine Learning" (2019). Master's Projects. 691

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient Estimation of Word Representations in Vector Space, 2013, http://arxiv.org/abs/1301.3781.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient Estimation of Word Representations in Vector Space, 2013, http://arxiv.org/abs/1301.3781.

[6] MoungHo Yi, MyungJin Lim, Hoon Ko, JuHyun Shin, "Method of Profanity Detection Using Word Embedding and LSTM", Mobile Information Systems, vol. 2021, Article ID 6654029, 9 pages, 2021. https://doi.org/10.1155/2021/6654029

[7] Weiss, Karl & Khoshgoftaar, Taghi & Wang, DingDing. (2016). A survey of transfer learning. Journal of Big Data. 3. 10.1186/s40537-016-0043-6.

[8] https://github.com/theabuseproject/tapad/tree/master/audio

[9] https://www.tensorflow.org/datasets/catalog/speech_commands

[10] Wikipedia contributors. "Mel-frequency cepstrum." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 7 May. 2021. Web. 21 Sep. 2021.

[11] N. Donges, “What is transfer learning? Exploring the populardeep learning approach,” Built In, 16-Jun-2019. [Online]. Avail-able: https://builtin.com/data-science/transfer-learning.

[12] Wikipedia contributors. "Residual neural network." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 10 Aug. 2021. Web. 21 Sep. 2021.

[13] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In ICML, 2010

[14] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2016). "Deep Residual Learning for Image Recognition". Proc. Computer Vision and Pattern Recognition (CVPR), IEEE. Retrieved 2020-04-23.

[15] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255).

[16] https://towardsdatascience.com/audio-deep-learning-made-simple-automatic-speech-recognition-asr-how-it-works-716cfce4c706

[17] S N Endah et al 2017 IOP Conf. Ser.: Mater. Sci. Eng. 190 012039.

[18] R. Ranganath, D. Jurafsky, and D. A. Mcfarland, “Detecting friendly,flirtatious, awkward, and assertive speech in speed-dates,” ComputerSpeech & Language, vol. 27, no. 1, pp. 89–115, 2013.

[19] https://pytorch.org/tutorials/intermediate/speech_command_recognition_with_torchaudio.html#formatting-the-data