Adversarial Examples for Keyword Spotting Systems using GANs

Sushil Khyalia* (160050035), Kartik Khandelwal* (160070025) and Syamantak Kumar* (16D070025)

***Abstract*— In this project we implement a Generative Adversarial Network (GAN) which generates adversarial noise for a given Keyword Spotting (KWS) system and then retrain the KWS system on the adversarial examples to make it more robust.**

I. INTRODUCTION

In recent years, with the increasing use of deep learning based systems in various fields, the robustness of these systems have become a major point of concern, especially in safety-critical fields like autonomous vehicles. Deep neural networks have been recently found vulnerable to well-designed input samples, called adversarial examples. **Adversarial examples** are defined as slightly skewed examples which are mis-classified by the model. They are imperceptible to human but can easily fool deep neural networks in the testing/deploying stage.[1].

Keyword Spotting refers to the problem of identifying keywords in speech. It can potentially be used to develop fully hands-free interface. Keyword spotting is essentially a classification problem in which the model needs to classify the audio into two classes - either the keyword is present or the keyword is absent. In this project, we aim at generating adversarial examples for keyword spotting systems and retraining the system to enhance robustness.

II. PRIOR WORK

Adversarial examples have been an important area of focus in recent years. [1] proposes **Fast Gradient Sign Method(FGSM)**[1], an efficient technique for generating adversarial examples and for increasing robustness of deep learning models by modifying the training routine. They essentially add a small perturbation to the input and select the perturbation such that there is a large variation in the output. [2] used FGSM to generate adversarial examples for an attention-based KWS model. They extract 40 dimensional mel-filterbank features from audio signals and add noise to the features to generate adversarial examples. Then they augment the data with the original model and retrain the KWS model, observing an an overall decrease in the False Reject Rate. [3] uses a genetic algorithm which adds small noise to input audio, without any knowledge of the victim model architecture or parameters, to generate adversarial examples. More recently, **GAN** based solution, namely **AdvGAN (Adversarial GAN)**[4], have also been proposed for the purpose of generating adversarial examples for image classification. We plan to use GANs in a similar manner as [4] for the purpose of generating adversarial examples for keyword spotting systems.

*Equal Contribution

[1]A good summary of recent findings in on adversarial examples on deep learning is presented in [8]

III. PROPOSED APPROACH

In this work, we plan to generate adversarial examples for keyword spotting using AdvGANs [4] and retrain the keyword-spotting model after augmenting the data with the adversarial examples generated. Our project can be divided into the following components :

- Dataset for training KWS model and AdvGAN
- Choosing a Keyword Spotting (KWS) model to perform adversarial attack
- Input/Output Feature Selection
- Selecting an architecture for Generator and Discriminator

A. Dataset for training KWS model and AdvGAN

Google Speech Commands v2.0 containing over 105,000 1-second long .wav files of 30 different utterances such as up, down, left, right etc. We select 7 classes - yes, no, marvin, left, right, silence, unknown for our experiments and use a 80:10:10 split of the data into training, validation & test set.

B. Choosing a Keyword Spotting (KWS) model to perform adversarial attack

For the purpose of a KWS model, we are using Honk which is a PyTorch re-implementation Google's TensorFlow convolutional neural networks for keyword spotting based on [6] as shown in figure 1. It takes as input the MFCC features from the audio signals and classifies it into one of the predefined keyword classes.

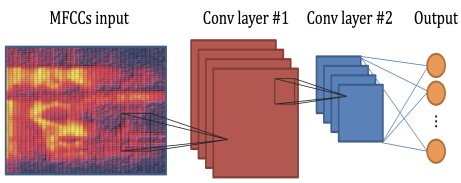

Fig. 1. Convolutional Neural Network Architecture for Keyword Spotting

C. Input/Output Feature Selection

Since we are generating adversarial samples, it is necessary for them to be imperceptible to the human ear. For this purpose, we require such features from which reconstruction of audio signals is possible and efficient. The KWS model we have have selected is constrained to be trained on MFCC features. We therefore, select Mel spectrogram features for

the purpose of training the GAN. It is possible to convert the Mel spectrogram to MFCC features through operations that allow backward flow of gradients. The transformation involves taking natural logarithm of the mel spectrogram and multiplying by the discrete cosine transform filter matrix.

D. Selecting an architecture for Generator and Discriminator

The GAN architecture is similar to the one proposed in **CycleGAN-VC** [7] which used it for the purpose of voice conversion. The generator and discriminator architecture are shown in *Figure 2*. We implemented the generator and discriminator network for the purpose of our task with a similar architecture as [7], using MFCC features instead of MCEP features.

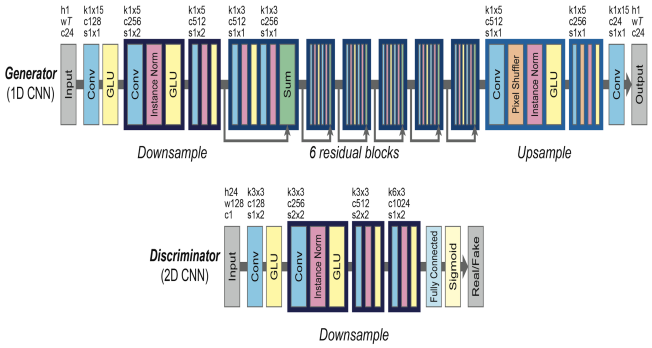

Fig. 2. GAN architecture

IV. Training pipeline and optimisation criterion

Figure 3 shows an overview of the AdvGAN used for the purpose of generating Adversarial examples. The generator network G takes x (a mel spectrogram) as input and generates a perturbation $G(x)$. Then $x + G(x)$ will be sent to the discriminator D, which is used to distinguish the generated data and the original instance x. The goal of D is to encourage that the generated instance is indistinguishable with the data from its original class. In tandem, to generate Adversarial examples, we input x to the target KWS model, whose output is used to calculate the adversarial loss L_{adv} whose goal is to encourage the perturbed input $(x + G(x))$ to be classified incorrectly.

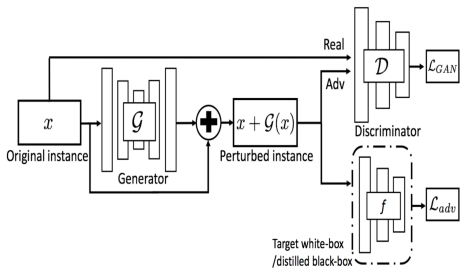

Fig. 3. AdvGAN setup

The AdvGAN has 4 different types of loss functions - Adversarial Loss, Generator Loss, Discriminator Loss, Hinge Loss.

- **Adversarial Loss** L_{adv} : To make the KWS network mis-classify the example L_{adv} = *Mean over batch of probability of correct class*
- **Generator Loss** L_{gen} : Training the generator network. $L_{gen} = -log(D(x + G(x)))$
- **Discriminator Loss** L_{disc} : Training the discriminator network. $L_{disc} = -[log(D(x)) + log(D(x + G(x)))]$
- **Hinge Loss** L_{hinge} :To constrain the noise added by the Generator $L_{hinge} = max(||G(x)||_2 - c, 0)$

The generator is trained on $L_{gen} + \alpha * L_{hinge} + \beta * L_{adv}$ and the discriminator is trained on L_{disc}.

V. Experimental Setup

The KWS Model was first trained on the Google speech commands dataset with 7 classes corresponding to *silence, unknown, yes, no, marvin, left, right* using SGD with momentum. The model converged to a test accuracy of 94.1%. This trained model was then used to train the AdvGAN. The hyper parameters used during training are : learning rate set to 0.001, SGD with momentum(0.9) and L2 regularisation($1e^{-5}$) for optimizer, $\alpha = 0.1$ and $\beta = 0.04$ as loss coefficient,. $c = 0$ for L_{hinge}. The Mel Spectrograms were mean and variance normalized before feeding into the GAN.

VI. Results and Observations

The original KWS model was trained on a pre-processed input in which background-noise is added to the audio signals along with a random time shift randomly selected from 0 to 100 ms at the start or end, before MFCC features were extracted from them. After the AdvGAN model was trained, the KWS model is retrained on sample augmented with adversarial examples obtained from AdvGAN. We evaluate the performance of the re-trained model on both clean and adversarial test data and compare it's performance with the original model. We also test against training the entire model using **SpecAugment** [5], a very recent data augmentation technique for speech recognition. Table VI summarises our results.

The following observation can be made from the results obtained:

- The accuracy on the Adversarial test set (71%) is significantly smaller than the original test set (94.1%), thus proving that the Generator has leaned to fool the kws model.
- Retraining on test data augmented with the Adversarial train samples increases the test accuracy on both the clean and Adversarial Test set showing that training on the Adversarial samples helps to make model more generalised and robust
- One interesting observation is that the kws model trained with SpecAugment suffered a smaller drop in accuracy as compared to original model indicating that

SpecAugment indeed makes the model more robust to noise

TABLE I
COMPARISION OF ORIGINAL MODEL, MODEL RETRAINED ON ADVERSARIAL EXAMPLES AND MODEL TRAINED USING SPECAUGMENT

KWS Model	Clean Test Set	Adversarial Test Set
Original KWS Model	94.1%	71.6%
Retrained KWS Model	**96.07%**	**91.78%**
KWS Model trained using SpecAugment	95.25%	81.56%

Figure VI represents the original spectrogram, the noise added by the generator and the noisy mel spectrogram obtained after adding the noise. These are computed for a specific utterance, "No". We observe that the confidence of the model on the noisy spectrogram is substantially low as compared to the original spectrogram. Although it seems that the mel-spectrogram is very different from the original, the sound generated after converting the spectrogram to audio is very similar to the original audio, and the noise almost imperceptible. The numbers show the confidence reported by the original KWS model on the spectrogram. Some sample audios and their adversarial counterparts are available here.

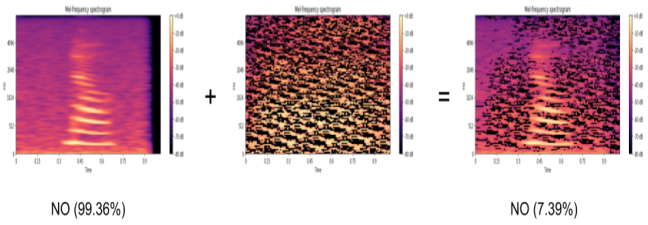

Fig. 4. Adding noise using AdvGAN to an utterance of "No"

VII. IMPLEMENTATION DETAILS

All implementations were done using PyTorch. Our code is available at the following Google Colab links :

- Honk Keyword Spotting Model
- AdvGAN for Keyword Spotting
- KWS Training using SpecAugment

VIII. CHALLENGES & FUTURE WORK

- Initially adding noise directly to MFCC features; No effective method to perform regeneration of audio from MFCC due to loss of temporal information
- Training of GAN quite unstable and extensive hyper parameter tuning was required to generate good examples
- One direction of exploration may be to use an alternative definitions of Adversarial loss than simply just averaging class probabilities
- The technique used could also be extended to generate adversarial examples for more complex tasks such as ASR

IX. CONCLUSION

In this project, we explored the use of adversarial examples generated using an AdvGAN for improving the performance of a Keyword Spotting System. We first verified that adversarial queries with almost imperceptible noise could be created using the proposed GAN architecture. Then we augmented the training data with these generated adversarial examples to retrain our KWS model. we observed that it helped the model to achieve improved performance on the test set thus providing a way to improve robustness of the model.

REFERENCES

[1] Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy. *Explaining and Harnessing Adversarial Examples*

[2] Xiong Wang, Sining Sun, Changhao Shan, Jingyong Hou, Lei Xie, Shen Li, Xin Lei, 2019. *Adversarial Examples For Improving End-To-End Attention-Based Small-Footpring Keyword Spotting*

[3] Moustafa Alzantot, Bharathan Balaji, Mani Srivastava *Did you hear that? Adversarial Examples Against Automatic Speech Recognition*

[4] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu and Dawn Song, 2019. *Generating Adversarial Examples with Adversarial Networks*

[5] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu1, Barret Zoph, Ekin D. Cubuk, Quoc V. Le1 *SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition*

[6] Tara N. Sainath, Carolina Parada. *Convolutional Neural Networks for Small-footprint Keyword Spotting*

[7] Takuhiro Kaneko and Hirokazu Kameoka, 2018. *CycleGAN-VC: Non-parallel Voice Conversion Using Cycle-Consistent Adversarial Networks*

[8] Xiaoyong Yuan, Pan He, Qile Zhu, Xiaolin Li. *Adversarial Examples: Attacks and Defenses for Deep Learning*