Adversarial Black-Box Attacks for Automatic Speech Recognition Systems Using Multi-Objective Genetic Optimization

Shreya Khare,† Rahul Aralikatte,‡ Senthil Mani,†
†IBM Research, ‡University of Copenhagen
shkhare34@in.ibm.com, rahul@di.ku.dk, sentmani@in.ibm.com

Abstract
Fooling deep neural networks with adversarial input have exposed a significant vulnerability in current state-of-the-art systems in multiple domains. Both black-box and white-box approaches have been used to either replicate the model itself or to craft examples which cause the model to fail. In this work, we use a multi-objective genetic algorithm based approach to perform both targeted and un-targeted black-box attacks on automatic speech recognition (ASR) systems. The main contribution of this research is the proposal of a generic framework which can be used to attack any ASR system, even if its internal working is hidden. During the un-targeted attacks, the Word Error Rates (WER) of the ASR degrades from 0.5 to 5.4, indicating the potency of our approach. In targeted attacks, our solution reaches a WER of 2.14. In both attacks, the adversarial samples maintain a high acoustic similarity of 0.98 and 0.97.

Introduction
Advancements in deep learning have improved the state-of-the-art systems, in domains like Computer Vision, Natural Language Processing and Speech Recognition. Recent studies (Biggio et al. 2017; Carlini and Wagner 2016; Szegedy et al. 2013) have shown that these systems can be easily fooled with carefully crafted inputs. For example, (Goodfellow, Shlens, and Szegedy 2014) showed that an image classifier could be fooled to classify an image to any class, by introducing small perturbations to the input image which were imperceptible to humans. As deep learning is being adopted in many safety-critical systems like autonomous driving (Geiger, Lenz, and Urtasun 2012) and biometric identification (Jain, Hong, and Pankanti 2000), it is imperative that they are robust and should not be fooled by adversarial attacks.

Automatic Speech Recognition (ASR) systems are becoming ubiquitous with the pervasiveness of smart devices. With the success of digital assistants like Google Assistant1 Apple’s Siri2 Microsoft’s Cortana3 and Amazon Alexa4 speech is replacing text as the main mode of communication with smart devices. Along with automating mundane human activities, these digital assistants can now accomplish complex tasks which may require sensitive user data. For example, Alexa can now access a user’s bank details and perform actions like checking the balance and making credit-card payments5. Therefore, it is essential that ASR systems are not susceptible to adversarial attacks.

Attacks on ASR systems can be classified based on two attributes: i) model transparency, and ii) intent of the attack.

Attacks based on model transparency: Attacks in which the adversary has access to the internals of an ASR system (network structure, weights, etc.) are known as ‘White Box’ attacks (Carlini and Wagner 2016). Attacks of this kind are rare as commercial ASR systems seldom expose their internal working to users. ‘Black Box’ attacks on the other hand, where the attacker only has access to the input-output pairs of the model, is more probable. This requires the attacker to somehow reverse engineer the model parameters of the neural network, which is computationally expensive and remains intractable in most real-world scenarios.

Attacks based on intent: The intention of an adversarial attack on an ASR can be either to cause failure or manipulate the system towards an end-goal. The former is called ‘Un-targeted’ attack where the intent is to make the ASR system output a wrong text for a given audio sample. In the latter, known as ‘Targeted’ attack, the aim is to perturb the sample in such a way that the ASR outputs some desired text.

---

1https://assistant.google.com/intl/en_in/  
2https://www.apple.com/in/ios/siri/  
3https://www.microsoft.com/en-in/windows/cortana  
4https://developer.amazon.com/alexa  
5https://goo.gl/TFHdkG
While targeted attacks can be dangerous as they can be used to manipulate actions taken by digital assistants, un-targeted attacks can degrade the ASR performance.

In recent literature, gradient estimation techniques have been used to generate adversarial examples in the image domain (Yuan et al. 2017). This technique, apart from being very expensive computationally, needs access to the gradient flow within the system which limits its scope to only white-box attacks. Since genetic algorithms (Last, Eyal, and Kandel 2006) are used extensively to solve black-box optimization (Audet and Kokkolaras 2016; Conn, Scheinberg, and Vicente 2009), problems, they have been used for carrying out black-box attacks on ASR systems with some success (Alzantot, Balaji, and Srivastava 2018; Taori et al. 2018).

A genetic algorithm works by slowly evolving better solutions over time, similar to biological evolution. Initially, a population of randomly initialized individual solutions are created, characterized by their ‘genes’, which can be considered as model parameters. Next, a fitness function is defined which measures the fitness score of an individual. Higher the fitness score, better the individual solution. The fittest individuals are then allowed to reproduce new individuals, and the unfit population is discarded. This process is repeated until a terminal condition is achieved.

In this work, we propose a multi-objective genetic algorithm based framework to generate targeted and un-targeted adversarial examples to fool ASR systems as shown in Fig. 1 as opposed to single objective genetic algorithms (Alzantot, Balaji, and Srivastava 2018; Taori et al. 2018). It takes an audio clip and a target text as input and outputs two adversarial samples: i) an audio clip which sounds similar to the original, but whose ASR generated text is as different as possible (un-targeted), and ii) an audio clip which remains close to the original audio, but moves the ASR generated text towards the desired text (targeted). Since the ASR system is a pluggable component in this framework, it can be used to attack any ASR system. Though it may seem that performing un-targeted attacks do not serve any purpose, they can be instrumental in generating high-quality adversarial inputs which can be used as training data to improve the robustness of the ASR systems.

Usually, complex problems with large search spaces cannot be effectively solved by genetic algorithms due to their random search strategy and slow convergence rate. However, we find that with a carefully crafted initialization technique and fitness function, the genetic algorithm can produce adversarial audio samples which are very similar to the original clips.

To summarize, the main contributions of the paper are: i) A novel multi-objective optimization approach based on text and acoustic similarity for generating adversarial samples, and ii) A quantitative and qualitative evaluation of the adversarial samples on state-of-the-art DeepSpeech ASR system.

Existing Literature

Adversarial Generation in Vision

(Goodfellow, Shlens, and Szegedy 2014) demonstrated that neural networks are vulnerable to very small changes in the input. These changes, which are indistinguishable to the human eye, cause the classifier to mis-classify the input. They developed the Fast Gradient Sign Method (FGSM) for generating adversarial examples to showcase that the state-of-the-art neural models are not robust to minute adversarial perturbations introduced in their inputs. This work triggered a huge interest in the community and various approaches have been proposed for increasing the robustness of networks for tasks like image classification, face detection, image segmentation, etc. (Papernot et al. 2016; Szegedy et al. 2013; Tabacof and Valle 2016; Kos, Fischer, and Song 2017; Arnaab, Miksik, and Torr 2017).

Adversarial Generation in Audio

In contrast to images, audio presents additional challenges for attackers to deal with. Majority of image-based deep learning models directly operate at the pixel level. However, ASR attacks require the input to be pre-processed to some intermediate representation. For example, an audio waveform is usually transformed by: i) obtaining the Mel-frequency Cepstrum Coefficients (MFCC) as features, or ii) transformed in the frequency domain to obtain a spectrogram, which is then fed into ASR systems. (Cisse et al. 2017) proposed a framework which can be used to attack a variety of deep learning models, both for image and audio inputs. But the framework was only able to generate the adversarial spectogram for audio samples and not the actual audio samples themselves. Rebuilding time domain signals from a spectogram is difficult due to overlapping windows which are used for analysis. This makes adjacent components in a spectogram dependent on each other.

In (Iter, Huang, and Jermann 2017), the proposed approach generated adversarial speech samples for ASR systems by adding perturbations to the MFCC features and performing lossy reconstruction of the speech signal in the time domain. This lossy reconstruction made the perturbations perceptible to the human ear. (Carlini and Wagner 2018) developed a method which enabled the propagation of gradients to the MFCC reconstruction layer. They performed white box targeted attacks on Mozilla’s DeepSpeech6 model and obtained adversarial samples which were 99.9% similar to the desired target text.

Furthermore, works like (Roy, Hassenieh, and Roy Choudhury 2017; Zhang et al. 2017) have utilized non-linearities of microphones in smart devices to generate commands (adversarial examples with frequency 40KHz.) audible only to speech assistants and not to humans. Another kind of attack proposed in (Carlini et al. 2016) intends to generate sounds that are interpreted as voice commands by devices but are unrecognizable to human listeners. These attacks, although powerful, are usually easy to detect or prevent.

https://github.com/mozilla/DeepSpeech
Adversarial Generation in Audio with Genetic Algorithms

(Alzantot, Balaji, and Srivastava 2018) used a genetic algorithm to perform a black-box attack on audio samples containing short speech commands. Each waveform belonged to a pre-defined class and the attack changed the way they were classified. This was done by iteratively applying noise to the audio samples.

(Taori et al. 2018) extended this approach to work on longer phrases and sentences. Their approach is limited to ASR systems which give access to the last layer (logits) of the model and hence is limited to systems like Mozilla Deep-speech. Additionally, their approach can be considered as Grey-box at best, as the attacks are generated based on prior knowledge of the model’s loss function and may not scale to other ASRs. These works perform targeted attacks using a genetic algorithm by minimizing the some distance metric between the actual text and the text generated by the ASR with the adversarial audio as input and do not use any features to capture acoustic similarity between the adversarial sample and original sample.

In our work, we consider adversarial audio generation as a multi-objective optimization problem which should achieve a balance between conflicting objectives. As far as we know, we are the first to propose, implement and evaluate such a framework for adversarial audio generation.

Proposed Approach

An overview of the proposed framework for producing adversarial audio samples for a given input is shown in Algorithm 1. As with any genetic algorithm based approach, the optimization process can be divided into four main stages: i) population initialization, ii) fitness-based selection, iii) reproduction, and iv) mutation. The remainder of this section relates these stages with the major system components depicted in Fig. 2 and explains each component in detail.

Population In genetic systems, each potential solution is termed as an Individual. Each individual’s characteristics are determined by it’s Genes. Genes can be thought of as analogous to model parameters. The group of individual solutions in a genetic system is called its Population. The population iteratively gets fitter over time resulting in more optimal solutions. The population in each iteration is known as a Generation. At the beginning of the first generation, genes of the population are either randomly initialized or initialized according to some heuristics which helps the process to converge quickly.

Fitness Computation In order to gauge the optimality of each individual solution, a Fitness Function is defined for the problem at hand. The genes of each individual are plugged into the fitness function to get a Fitness Score.

In literature, adversarial attacks usually use a fitness function with a single objective. Their approaches select elite individuals from the population based on a single criterion.

Algorithm 1 Framework for producing adversarial audio samples, given an audio input

1: if Targeted then
2: Input: audio signal: a, text: \( t = \text{ASR}(a) \),
3: target text: \( t' \), maxiters
4: else
5: Input: audio signal: a, text: \( t = \text{ASR}(a) \), maxiters
6: Output: adversarial audio signal: \( a' \)

1: procedure Initialization(\( x \))
2: \( P_0 \leftarrow [x + w \times \mathcal{N}(\mu, \sigma^2)] \times \text{popsize} \)
3: return \( P_0 \)
4: end procedure
5: procedure Fitness Computation(\( P \))
6: for \( p \in P \) do
7: \( t' \leftarrow \text{ASR}(p) \)
8: if Targeted then
9: \( f_1 \leftarrow \text{EditDistance}(t', t'') \) \( \triangleright \) minimized
else
11: \( f_1 \leftarrow \text{EditDistance}(t', t) \) \( \triangleright \) maximized
12: \( f_2 \leftarrow \text{Similarity}((\text{MFCC}(p), \text{MFCC}(a))) \)
13: return \( f_1, f_2 \)
14: end if
end for
end procedure
6: procedure Mating Pool Identification(\( P, F \))
16: \( \text{pool}_1 \leftarrow \text{SameRankSelect}(P, F) \)
17: \( \text{pool}_2 \leftarrow \text{InvertedRankSelect}(P, F) \)
18: \( \text{pool}_3 \leftarrow \text{RouletteWheelSelect}(P, F) \)
19: return \( \text{pool}_1 \cup \text{pool}_2 \cup \text{pool}_3 \) \( \triangleright \) list of parent pairs
20: end procedure
21: procedure Crossover(\( \text{ParentPairs} \))
22: for \( p_1, p_2 \) in \( \text{ParentPairs} \) do
23: \( \text{children}_1 \leftarrow \frac{p_1 + p_2}{2} \)
24: \( \text{children}_2 \leftarrow \frac{2p_1 + p_2}{3} \)
25: \( \text{children}_3 \leftarrow \frac{p_1 + 2p_2}{4} \)
26: return \( \text{children}_1 \cup \text{children}_2 \cup \text{children}_3 \)
27: end for
end procedure
28: procedure Mutation(\( c \))
29: for \( c \in \text{children} \) do
30: \( \text{children}_m \leftarrow [c + \text{prob} \times \mathcal{N}(\mu, \sigma^2)] \)
31: return \( \text{children}_m \)
32: end for
end procedure
33: procedure GetDominations(\( P \))
34: \( P_{\text{new}} \leftarrow \text{ComputeDominance}(P) \)
35: \( P_{\text{new}} \leftarrow \text{ReRank}(P_{\text{new}}) \)
36: return \( P_{\text{new}} \)
37: end procedure
38: repeat
39: \( P_i \leftarrow \text{Initialization}(a) \)
40: \( F_i \leftarrow \text{FitnessComputation}(P_i) \)
41: \( \text{parents}_i \leftarrow \text{MatingPoolIdentification}(P_i, F_i) \)
42: \( \text{children}_i \leftarrow \text{Mutation}(\text{Crossover(\text{parents}_i)}) \)
43: \( P_{i+1} \leftarrow \text{GetDominations(\text{parents}_i \cup \text{children}_i)} \)
44: until \( P_i \equiv P_{i+1} \) or \( i > \text{maxiters} \)
45: \( P_{\text{final}} = P_i \)
46: \( a' \leftarrow P_{\text{final}}[0] \) \( \triangleright \) highest ranked individual
(Alzantot, Balaji, and Srivastava 2018) use the score of the model and (Taori et al. 2018) use text similarity. These works have ignored acoustic similarity, which is a strong indicator of the goodness of an individual. An audio sample is considered to be a good adversary, if it sounds very similar to the original sample but generates an entirely different text. Hence the sample generated should have: i) maximum acoustic similarity and minimum text similarity (for un-targeted attacks), or ii) maximum acoustic similarity and maximum text similarity with a target text (for targeted attacks). We achieve this by introducing a fitness function with multiple objectives which tries to strike a balance between acoustic and text similarities.

A multi-objective optimization problem can be formulated as shown in Eqn.\[1\] where \(N\) is the number of objective criteria:

\[
\forall i \min \max f_i(x) \quad (1)
\]

Our fitness function uses two objectives to calculate each individual’s fitness score: i) the Euclidean distance between the Mel-frequency cepstral coefficients (MFC) (Rabiner, Gold, and Yuen 1978) of the original and the generated audio samples, and ii) the edit distance between the texts generated by the ASR system when the original and generated sample are provided as inputs.

**Mating Pool Identification** Once the fitness score of each individual in a population is computed, we create pairs of individuals which act as parents to the next generation. Usually, this is done by picking the top-\(k\) individuals with the best fitness scores and pairing them randomly. However, this does not work well when the fitness function has more than one objective to optimize. Therefore, we use an ensemble of techniques to create a diverse mating pool which is more likely to produce fitter individuals:

- **Same Rank Selection:** In this selection scheme, we first create two lists \(l_1\) and \(l_2\) by ranking the individuals based on each of the two objectives. Let \(l_x \leftarrow i_{xx}\) where \(i\) is an individual in list \(x\) with rank \(r\). We pair two ‘\(i\)’s which have the same \(r\) in the two lists.

- **Inverse Rank Selection:** This scheme is the opposite of the previous one. One list is ranked from best to worst and the other vice-versa. This makes sure that diversity is maintained in the population and we do not constrain the search space.

- **Roulette Wheel Selection** (Eshelman and Schaffer 1992): In this scheme, the fitness score is used to compute a probability of selection for each individual. If \(f_i\) is an individual \(i\)’s fitness score, then its probability of being selected is:

\[
p_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \quad (2)
\]

where \(N\) is the number of individuals in the population. While unlikely, individuals with high fitness scores may still be eliminated ensuring there is no premature convergence to local optima.

**Crossover** Once the mating pairs are identified, three crossover functions are applied on them to form new offspring whose genes are a combination of its parents. We use arithmetic recombination operators to perform a single-point crossover (Eiben and Smith 2003). If \((p_1, p_2)\) is a mating pair, then the first child is created to have the characteristics of the parents in equal proportion:

\[
p_1 = \frac{p_1 + p_2}{2} \quad (3)
\]

The second child is created such that \(p_1\)’s characteristics is dominant in its genes.
Figure 3: A Pareto Chart showing the dominance of individuals in various generations for a random audio sample during an un-targeted attack.

$\frac{2 \times p_1 + p_2}{3}$

and the third child’s genes are dominated by $p_2$’s characteristics.

$\frac{p_1 + 2 \times p_2}{3}$

**Probabilistic Mutation** Once the offsprings are created, we mutate parts of their genes so that the new population becomes diverse and can hence be scattered over a larger area in the search space by adding Gaussian noise to the genes with a probability $prob_m$.

**Dominance evaluation** In multi-objective optimization, a candidate solution $x_i$ is said to dominate another solution $x_j$, if $x_i$ is no worse than $x_j$ in all objectives and $x_i$ is strictly better than $x_j$ in at least one objective. Multi-objective schemes with conflicting objective functions often give rise to a set of two or more optimal solutions because none of them can be considered as better than the other(s) concerning all objective functions. These optimal solutions are known as Pareto-optimal solutions (Fonseca and Fleming 1995).

In our system, after a set of offspring is created, their fitness scores are evaluated in the same way as it was done for their parents. The combined set of parents and children are ranked based on their dominance. The most dominating individuals are carried forward to the next generation, and the others are discarded. Fig. 3 shows the edit distance and MFCC distance of individuals over generations. The blue points are the individuals of the final generation. We can observe the formation of a knee curve, and the individual closest to the knee point is considered to be the best solution.

**Implementation Details**

A set of 100 individuals are created to form the initial population. They are initialized by adding random uniform noise to the original audio signal with a sampling rate of 16,000. Therefore, a two-second audio clip will result in individuals having 32,000 genes. Intuitively, this helps in faster convergence when compared to random initialization.

After computing the fitness scores based on acoustic (MFCC) and textual (edit distance) similarities, the population is converted to a list of mating pairs. The extraction of MFCC features is performed using python librosa library\footnote{https://github.com/librosa/librosa} with window size of 25ms and a stride of 10ms. It is to be noted that an individual may appear in more than one mating pair. The three crossover operations described previously are applied to each pair to produce three children. Probabilistic mutation is applied to every gene of every individual with $prob_m = 0.005$.

Next, the parents and children are combined and re-ranked based on their dominance and the top 30 Pareto-optimal individuals are propagated to the next generation. This process is repeated for a maxiters = 50, or until we get the same set of Pareto-optimal solutions over two successive generations which indicates convergence. The most dominant individual from the last generation is the adversarial counterpart for our input audio signal. After termination, there may be multiple, equally good dominant solutions to pick from (with equal dominance). In this case, we choose one at random as the best solution. We run all the experiments on a 16 core machine with 64GB of RAM. A typical two-second clip will converge in $\sim$ 60 minutes showcasing the speed of our approach.

**Experimental Results and Analysis**

In this section, we describe the dataset, the experiments carried out, and the metrics used for evaluating the results.

We use the standard Mozilla Common Voice data-set (CVS\footnote{https://voice.mozilla.org/en}) to test the effectiveness of our approach. It is to be noted that since our approach does not ‘learn’, it does not require any training data. Therefore, we only take the 100 test instances of CVS and use them to perform both un-targeted and targeted attacks.
### Table 1: Texts generated by Deepspeech on randomly selected adversarial samples during un-targeted and targeted attacks.

| Attack Type       | Original text                                                                 | Deepspeech Text                  | Targeted Text | Adversarial Text |
|-------------------|--------------------------------------------------------------------------------|----------------------------------|---------------|------------------|
| Un-targeted Attack| the one you are blocking                                                       | the one you are blocking         | N.A.          | the money of locking |
| Un-targeted Attack| follow the instruction here                                                    | follow the instruction here       | N.A.          | all of these shapes |
| Un-targeted Attack| what's your name he asked                                                      | what's yo name he asked          | N.A.          | one of late ask   |
| Targeted Attack   | I've got to go to him                                                          | I've got to go to him             | a cat         | a cat             |
| Targeted Attack   | don't do it for me                                                             | don't do it for me                | I am good     | I am good without it |
| Targeted Attack   | do you know what                                                               | did you know that                 | that I love you| I love            |

Table 1 shows original text (ground truth), text generated by Deepspeech on original speech, targeted text (text desired to be generated) and the Deepspeech generated text on our adversarial samples.

We have evaluated our approach in two ways: i) quantitative - where we empirically prove the goodness of the samples by measuring their word error rates and correlation coefficient, and ii) qualitative - where we show that the generated audio samples are good by conducting a survey. We perform this evaluation for both un-targeted and targeted attack scenarios. Also, for all the experiments conducted, Deepspeech is used as the underlying ASR system.

### Experimental Setup
Let the input audio sample be denoted by \( a_i \) and its corresponding text be \( t_i \). Let the generated adversarial output be represented by the symbol \( a_o \) and let \( t_o \) be its corresponding text generated by the ASR.

For un-targeted attacks, the setup is straightforward. Given inputs \( a_i \) and \( t_i \), the goal of the framework is to produce an \( a_o \) such that \( t_o \) is as different from \( t_i \) as possible while maintaining acoustic similarity with \( a_i \).

For targeted attacks, we receive an additional input \( t_d \) which is the desired text to be generated. In this case, we want \( t_o \) to be ideally same as \( t_d \), or somewhat close to it, while maintaining acoustic similarity with \( a_i \).

### Metrics
The goal of generating adversarial samples is to degrade the performance of an ASR system. In literature, the performance of ASR systems is commonly measured using Word Error Rate (WER). WER compares a reference \((t_i, t_o)\) to a hypothesis \((t_o)\) and is defined as:

\[
W = \frac{S + D + I}{N}
\]

where, \( S, D \) and \( I \) are number of substitutions, deletions and insertions. \( N \) is the number of words in the reference text.

In order to measure the similarity of \( a_o \) with \( a_i \), we use correlation coefficient which is defined as:

\[
\rho(a_o, a_i) = \frac{\sum_{j=1}^{n}(a_{oj} - \overline{a_o})(a_{ij} - \overline{a_i})}{\sqrt{\sum_{j=1}^{n}(a_{oj} - \overline{a_o})^2 \sum_{j=1}^{n}(a_{ij} - \overline{a_i})^2}}
\]

Higher values of \( \rho(a_o, a_i) \) corresponds to higher similarity between \( a_i \) and \( a_o \).

### Quantitative Evaluation

**Word Error Rate:** The first row of Table 2 shows the WER of the text generated by Deepspeech for the original and adversarial audio samples. The 100 test samples from the CVS dataset are first fed into Deepspeech. The average length of these samples is \( \sim 6 \) words. As expected, the WER is low at 0.5. Next, an un-targeted attack is carried out on the test set using the generated adversarial samples and a WER of 5.4 is obtained for the text generated by Deepspeech. This shows that, on an average 5 or 6 insert, substitute or delete operations are required to convert the generated text to the original text. In other words, most of the words are recognized wrongly by the ASR, hence degrading the performance of an ASR which shows the goodness of our approach for generating adversarial samples.

For targeted attacks, the target text snippets are generated as follows: i) a vocabulary of words present in the 100 original audio samples is created ii) if the word length of the original sample text is \( n \), then \( n \pm 1 \) words are randomly sampled from the vocabulary to form the target text. In this situation, the WER between the original and targeted text will be in the range \((n - 1, n + 1) \approx (4, 6)\). When a targeted attack is performed, the WER drops to 2.14 after 50 iterations. We observe that this value keeps decreasing as we increase the number of iterations.

**Correlation Coefficient:** The average correlation between the 100 test samples and the generated adversarial samples is 0.98 for un-targeted attacks and 0.97 for targeted attacks, indicating the generated samples are extremely similar to the original audio. Fig. 4 shows one sample of original audio clip plotted in the time domain, overlaid on the adversarial sample’s plot for an un-targeted attack. As evident from the figure, the two are almost equivalent wave forms, emphasizing the high correlation between them.

### Qualitative Evaluation

While quantitative evaluation measured dissimilarity between the original text and the text generated by Deepspeech from the adversarial samples using WER and CC, it is necessary to measure the quality of generated audio itself. For this purpose, a qualitative evaluation has been carried out by
conducting subjective listening tests with 15 participants.

Each participant was presented with 15 random adversarial samples (both targeted and untargeted) and were required to transcribe the text and rate the listening effort required to comprehend the text, which is an indirect measure of the noise/aberrations present in the samples. WER is then calculated for the transcribed text and compared with the ASR generated text, with the original text being the ground truth. Fig. 5 shows a box plot of the WER for Deepspeech and human transcribed text. Overall, we had 225 data points to infer our insights. The mean WER for the human transcribed text is close to 0 which clearly suggests that humans were not able to find any significant difference between the adversarial and the original samples. Deepspeech, on the other hand, generated incorrect transcriptions for the same samples.

Fig. 6 depicts the listening effort required by the participants to comprehend the audio. More than 80% of the participants required no effort to comprehend and transcribe the text, indicating that there was either no or minimal noise in the generated adversarial samples.

**Conclusion and Future Work**

In this work, we introduce a framework for fooling ASR systems using a multi-objective genetic algorithm based approach. The framework can be extend for any ASR system and can be used for both un-targeted and targeted attacks. Our fitness function uses Euclidean distance of MFCC features to get the acoustic similarity between audio samples and edit distance to measure the generated text dissimilarity. To the best of our knowledge, this is the first time a multi-objective fitness function is used to optimize auditory and textual features jointly.

We evaluate our framework both qualitatively and quantitatively on Mozilla Deepspeech ASR by reporting various metrics like WER, CC, and listening effort. For un-targeted attacks, we see that Deepspeech’s WER increases by a factor of 10 from 0.5 to 5.4 while maintaining a CC of 0.98 which indicates very high audio similarity. For targeted attacks, where the generated text should move closer to the desired phrase, we report a WER of 2.14 with a CC of 0.97. This means that on an average, the generated text is two edit operations away from the desired text. We also conduct a survey of 15 users to evaluate the generated audio.

Our findings prove that, generation of adversarial samples through black-box attacks i.e., having access only to inputs and outputs of ASR systems, is a viable approach. It will be interesting to explore the following questions in the future.

**Is this approach generalizable to other domains?** It will be interesting to examine if the proposed generic approach of generating adversarial samples works well in other domains like image, text by extending the fitness function.

**Are other ASR systems robust?** The findings presented in this research is based on the evaluation on Deepspeech ASR system. Since our approach considers ASR systems to be a black-box, and only requires the output text of ASR system, it can be extended for evaluating other ASR systems like Kaldi ASR, Wave2letter etc. Additionally, building defence mechanisms against such attacks will be an interesting future direction.

**Are the attacks transferable?** Currently, the optimization framework needs to be run for every audio input. If the ASR system changes, then the experiment has to be restarted. An interesting research direction would be to generate adversarial examples which work well on multiple ASRs, thus avoiding the need for re-running the optimization framework every time.

**Can other speech based features/techniques be useful?** This work reveals the importance of acoustic similarity for generating high quality adversarial samples. We use MFCC features to capture important speech signal characteristics, which are retained by the offspring during optimization. It will be interesting to see the effects of using other time and spectral features of the speech as a similarity metric.
References

[Alzantot, Balaji, and Srivastava 2018] Alzantot, M.; Balaji, B.; and Srivastava, M. B. 2018. Did you hear that? adversarial examples against automatic speech recognition. CoRR abs/1801.00554.

[Arnab, Miksik, and Torr 2017] Arnab, A.; Miksik, O.; and Torr, P. H. S. 2017. On the Robustness of Semantic Segmentation Models to Adversarial Attacks. ArXiv e-prints.

[Audet and Kokkolaras 2016] Audet, C., and Kokkolaras, M. 2016. Blackbox and derivative-free optimization: theory, algorithms and applications. Optimization and Engineering 17(1):1–2.

[Biggio et al. 2017] Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Smidt, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2017. Evasion attacks against machine learning at test time. CoRR abs/1708.06131.

[Carlini and Wagner 2016] Carlini, N., and Wagner, D. A. 2016. Towards evaluating the robustness of neural networks. CoRR abs/1608.04644.

[Carlini and Wagner 2018] Carlini, N., and Wagner, D. A. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. CoRR abs/1801.01944.

[Conn, Scheinberg, and Vicente 2009] Conn, A. R.; Scheinberg, K.; and Vicente, L. N. 2009. Introduction to Derivative-Free Optimization. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

[Eiben and Smith 2003] Eiben, A., and Smith, J. 2003. Introduction To Evolutionary Computing, volume 45.

[Geiger, Lenz, and Urtasun 2012] Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, 3354–3361.

[Goodfellow, Shlens, and Szegedy 2014] Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and Harnessing Adversarial Examples. ArXiv e-prints.

[Iter, Huang, and Jermann 2017] Iter, D.; Huang, J.; and Jermann, M. 2017. Generating adversarial examples for speech recognition.

[Jain, Hong, and Pankanti 2000] Jain, A.; Hong, L.; and Pankanti, S. 2000. Biometric identification. Commun. ACM 43(2):90–98.

[Kos, Fischer, and Song 2017] Kos, J.; Fischer, I.; and Song, D. 2017. Adversarial examples for generative models. ArXiv e-prints.

[Last, Eyal, and Kandel 2006] Last, M.; Eyal, S.; and Kandel, A. 2006. Effective black-box testing with genetic algorithms. In Proceedings of the First Haifa International Conference on Hardware and Software Verification and Testing, HVC’05, 134–148. Berlin, Heidelberg: Springer-Verlag.

[Papernot et al. 2016] Papernot, N.; McDaniel, P. D.; Goodfellow, I. J.; Jha, S.; Celik, Z. B.; and Swami, A. 2016. Practical black-box attacks against deep learning systems using adversarial examples. CoRR abs/1602.02697.

[Rabiner, Gold, and Yuen 1978] Rabiner, L. R.; Gold, B.; and Yuen, C. K. 1978. Theory and application of digital signal processing. IEEE Transactions on Systems, Man, and Cybernetics 8(2):146–146.

[Roy, Hassanieh, and Choudhury 2017] Roy, N.; Hassanieh, H.; and Roy Choudhury, R. 2017. Backdoor: Making microphones hear inaudible sounds. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys ’17, 2–14. New York, NY, USA: ACM.

[Szegedy et al. 2013] Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2013. Intriguing properties of neural networks. CoRR abs/1312.6199.

[Tabacof and Vallee 2016] Tabacof, P., and Vallee, E. 2016. Exploring the space of adversarial images. In 2016 International Joint Conference on Neural Networks (IJCNN), 426–433.

[Taori et al. 2018] Taori, R.; Kamsetty, A.; Chu, B.; and Vemuri, N. 2018. Targeted adversarial examples for black box audio systems.

[Yuan et al. 2017] Yuan, X.; He, P.; Zhu, Q.; Bhat, R. R.; and Li, X. 2017. Adversarial examples: Attacks and defenses for deep learning. CoRR abs/1712.07107.

[Zhang et al. 2017] Zhang, G.; Yan, C.; Ji, X.; Zhang, T.; and Xu, W. 2017. DolphinAttack: Inaudible Voice Commands. ArXiv e-prints.