Abstract—Automatic speech recognition (ASR) systems are prevalent, particularly in applications for voice navigation and voice control of domestic appliances. The computational core of ASRs are deep neural networks (DNNs) that have been shown to be susceptible to adversarial perturbations; easily misused by attackers to generate malicious outputs. To help test the correctness of ASRS, we propose techniques that automatically generate blackbox (agnostic to the DNN), untargeted adversarial attacks that are portable across ASRs. Much of the existing work on adversarial ASR testing focuses on targeted attacks, i.e generating audio samples given an output text. Targeted techniques are not portable, customised to the structure of DNNs (whitebox) within a specific ASR. In contrast, our method attacks the signal processing stage of the ASR pipeline that is shared across most ASRs. Additionally, we ensure the generated adversarial audio samples have no human audible difference by manipulating the acoustic signal using a psychoacoustic model that maintains the signal below the thresholds of human perception. We evaluate portability and effectiveness of our techniques using three popular ASRs and three input audio datasets using the metrics - Word Error Rate (WER) of output text, Similarity to original audio and attack Success Rate on different ASRs. We found our testing techniques were portable across ASRs, with the adversarial audio samples producing high Success Rates, WERs and Similarities to the original audio.

Index Terms—Untargeted adversarial attack, Automatic Speech Recognition, Blackbox attack, Masking threshold, word error rate, cosine similarity

I. INTRODUCTION

Automatic speech recognition systems are widely used in a variety of applications, such as in-vehicle voice navigation and voice smart home appliances. However, such systems can be susceptible to security vulnerabilities that may be exploited by adversaries [19], [20], [28]. For instance, consider a voice saying “Go to a supermarket” to the car navigation system while driving. An attacker may construct an audio that sounds similar to the original but causes the ASR within the navigation system to translate it erroneously into say, “Go to a school” instead, causing the system to plan a different route. Many of the attacks on ASRs are constructed such that they have no audible difference between the attack and original audio, making them hard to detect.

Adversarial attack generation on ASRs can be classified into (1) targeted and non-targeted based on whether or not attacks are generated for a given text output, and (2) whitebox or blackbox based on whether or not knowledge of the neural network (NN) structure within the ASR is used. Most existing methods for attack generation are targeted and whitebox that set the text to be translated in advance and use the internal NN information to generate attacks. These methods suffer from the following drawbacks (1) Lack of portability to other ASRs owing to their whitebox nature where attacks are specifically designed for a given ASR. Additionally, internals of the NN structure may not always be visible making whitebox attacks impractical for certain ASRs, (2) Targeted text for attack generation needs to be known in advance. It is not practical to anticipate every possible targeted text for attacks, and (3) time taken to generate attacks is considerable. Attack generation can take several hours depending on the technique used. The few existing techniques for blackbox untargeted attack generation rely on using white noise for creating adversarial samples [2], [3], [15]. ASRs like Google are effective at filtering out such noise (seen in Section V) making these attacks ineffective.

To address these limitations, we propose a blackbox untargeted attack generation approach that generates adversarial samples by modifying the audio signal processing stage within the ASR. Most ASRs use the same or largely similar signal processing setup. Thus attacking this stage will result in adversarial samples that are portable across ASRs. The goal in our approach is to produce adversarial samples that will produce an output text markedly different from the original output while ensuring the attack sounds similar to the original. Unfortunately this goal is not easily achievable as maximising difference in output text often contradicts with maximising similarity to original audio.

Our approach uses a psychoacoustics concept called masking threshold that determines how sounds interfere and mask each other. We ensure the perturbations we introduce in the adversarial audio samples are not easily distinguished from the original audio by keeping them below the masking threshold of the original audio. We provide three noise generation approaches based on variations of this idea – Guassian White Noise (WN), Griffin Lim Reconstruction (GL) and Original Phase (OP). We also provide the option of selectively introducing perturbations to a small fraction of audio frames rather than all to increase similarity to the original sound. Our approach provides three frame selection options – Random, Important
and All. Among them, the Important option identifies the frames that cause the most change to output text when set to zero and we then introduce perturbations to just these important frames to maximize change to output text while maintaining similarity to original audio.

We evaluate our approach on three different ASRs – Deepspeech [13], Sphinx [16] and Google [1], using three different input audio datasets – Librispeech [21], Commonvoice [6] and Timit [10]. We assess the effectiveness of our approaches for noise generation and frame selection using the metrics - WER, Similarity and attack Success Rate. We also compare our approach with a targeted whitebox state of the art method [8]. It is worth noting that this is the first time an evaluation of adversarial attacks across different ASRs and different datasets has been conducted. The scale of our evaluation is much bigger than in existing work; we use 1600 audio samples in total as compared to order of tens or hundreds in existing work [2], [3], [8], [22].

We found our approach that uses GL or OP for noise generation combined with Important or All frame selection was effective at attacking all three ASRs using all three audio datasets, although with slightly different levels of effectiveness. On average, with OP+All, we achieved 63%/87% WER/Similarity on Librispeech, 78%/86% on Commonvoice and 71%/88% on Timit datasets. The attacks generated using one dataset were easily portable across several ASRs with Success Rate over 97% for OP+All compared to 33% with existing technique [8]. Finally, our attack generation is fast, up to 222× faster than Carlini et al. [8].

In summary, the contributions in this paper are as follows:

1) A novel portable approach for untargeted blackbox attack on ASRs that uses perturbations within the masking threshold of the input audio.
2) Novel technique that selectively introduces noise to only the important audio frames that are identified based on output impact.
3) Extensive empirical evaluation of the noise generation and frame selection options within our approach on three ASRs and three audio datasets using WER, Similarity and attack Success Rate. We also compare performance to an existing whitebox targeted technique.

The source code for our approach can be found at https://anonymous.4open.science/r/lalalala-54BF

II. BACKGROUND

We present a brief description of an ASR system and the masking threshold concept used in our approach.

A. Automatic Speech Recognition (ASR)

Structure and workflow within a typical ASR is shown in Figure 1. Most current ASRs comprise the following stages when converting an input audio to a translated text output.

1) Preprocessing: This step removes high-frequency noise in the audio. A voice activity algorithm is used to detect human voice parts in a given input audio and then passes it through a low-pass filter to remove high-frequency noise that cannot be perceived by human ears.

2) Signal Processing stage: Output from this stage is audio features that is subsequently used by a deep neural network. In the signal processing stage, the audio signal in the time domain is divided into frames and every frame is converted to the frequency domain using Fast Fourier Transform. The result of this step is a complex matrix, where the real part of the matrix is the amplitude information of the frame, and the imaginary part is the phase information. The phase spectrum is discarded, and only the amplitude spectrum is retained. This amplitude spectrum is the expression of the audio in the frequency domain, which details different frequencies and corresponding intensities in the frame. Subsequent steps in the ASR are completed on the basis of the amplitude spectrum.

To extract audio features, the amplitude spectrum is passed through Mel filters and Discrete Cosine Transform (DCT). The output is Mel Frequency Cepstral Coefficient (MFCC), which is commonly used in ASRs as features of audios. Detailed description of this step can be found in [4].

3) Neural network prediction and output selection stage: The extracted features from the audio are fed into a deep neural network (DNN), such as a Recurrent Neural Network, that then predicts a probability distribution of characters for every time step or audio frame. From the character sequence distributions, an output selection algorithm, such as Beam search, is used to select the most likely translated text as shown in Figure 1. More details on this stage can be found in [4].

It is worth noting that much of the existing work on adversarial attacks against ASRs are aimed at the DNN stage (prediction stage) and typically use gradient-based optimization to minimize the difference between the target and output text [8], [17], [22]. In contrast, our approach for generating adversarial attacks makes changes in the signal processing stage, described in Section III.

B. Frequency Masking and Masking Threshold Computation

Frequency masking is a psychoacoustic phenomenon that occurs when the perception of a sound is affected and masked by another, distracting the ear from being able to clearly perceive the simultaneous sounds [11], [15]. The masking audio signal creates a masking threshold such that other sounds below this threshold cannot be heard. Our main idea for adversarial attacks is to add noises that are below or equal to the masking threshold of the original audio so that they will not be detected by the human ear, while potentially changing the output text from ASRs.

To calculate the masking threshold for a given audio signal, we need to first convert the audio from the expression in the time domain to the frequency domain (using FFT in Section II-A2), then discard the phase information in the spectrum. We then use the amplitude information of the spectrum to calculate the log-magnitude power spectral density (PSD) of
this audio. PSD is better suited to spectral analysis than FFT as they are normalized to the frequency bin width preventing the duration of the data set (and corresponding frequency step) from changing the amplitude of the result. The process of calculating the masking threshold is typically divided into three parts: the identification of individual maskers; calculate their respective masking thresholds using a two-slop spread function which is to imitate the excitation mode of maskers, as seen in [17], [22]; and finally combine the individual masking thresholds into a global masking threshold. Detailed description of how to calculate the masking threshold can be found in [17], [18].

C. Griffin-Lim Algorithm

The Griffin-Lim (GL) algorithm helps reconstruct speech waveforms with a known amplitude spectrum but an unknown phase spectrum [12]. We use this algorithm to produce adversarial audio after perturbing the amplitude spectrum of the original audio. Steps in the algorithm are as follows: (1) Randomly initialize a phase spectrum, (2) Use this phase spectrum and the known amplitude spectrum to synthesize a new waveform through Inverse Short-Time Fourier Transform (3) Use the synthesized speech to get new amplitude spectrum and new phase spectrum through Short-time Fourier Transform, (4) Discard the new amplitude spectrum, (5) Repeat steps 2, 3, 4 for a fixed number of iterations. Output is a waveform with an estimated phase spectrum and the known input amplitude spectrum.

III. METHODOLOGY

In this section, we propose techniques for generating adversarial samples for ASRs. As seen in Figure 2 our methodology has two important stages, 1. Audio Frame Selection and 2. Noise Generation. The general workflow in our approach is as follows: Given an input audio sample, we first select audio frames using one of the three techniques for audio frame selection – Random, Important and All. Independently, we generate noisy audio from the input audio using one of three techniques – White Noise, GL Reconstruction, Original Phase. We then replace the selected frames in the original audio with corresponding noisy frames (from noisy audio generated in Stage 2) while keeping the rest of the audio unchanged. The combination of original and noisy audio forms the adversarial audio output. We describe our techniques for frame selection and noise generation in detail below.

A. Stage 1: Frame Selection

As mentioned in [II-A2] the audio signal input to an ASR is split into frames in the signal processing stage. We explore generation of adversarial audio by adding noise to a subset of frames in the entire audio. We provide three approaches to select audio frames into which noise should be added – Random, Important and All. We will start by describing the technique to select Important frames.

1) Important:: The rationale for selecting important frames into which we later add noise was to produce an adversarial audio that will potentially produce significant difference in the output text while still sounding similar to the original audio. We define importance of frames based on the proportion of WER produced by masking that frame in the original audio. The steps involved in selecting important frames is as follows,

1) For every input audio sample, record output translated text from ASR.
2) Pick one of the input audio samples. For every frame in the processed audio sample, set it to zero (masked) while keeping the remaining frames unchanged. Record translated text using the ASR for the masked audio.
3) Compute WER between the masked and original output. Repeat this for all frames. The frames that result in a non-zero WER are identified as important frames for that audio sample. Magnitude of WER for frame selection can be altered to suit needs.
4) Repeat Steps 2 and 3 for the remaining input audio samples.

At the end of this process, every input audio sample is associated with a list of important frames.

2) Random:: To enable us to compare the effectiveness of only using important frames in frame selection, we also provide a means to select frames randomly. The number of frames selected for a given audio sample is set to be the same as the number of important frames selected for that audio.

3) All:: We simply use all the frames from the noisy audio generated in the adversarial audio output. This was done to evaluate how much WER was achievable and compromised by
Fig. 2. Our approach for generating adversarial audio samples comprises of three stages, 1. Frame Selection, 2. Noise Generation and finally 3. Adversarial audio formed by combining information in the first two stages.

frame selection with Important and Random versus the gain in Similarity with those techniques.

B. Stage 2: Noise Generation

From the original audio sample, we generate noisy perturbed audio using three techniques. All three techniques add noise to the signal processing stage of the ASR rather than the DNN stage, which is unlike existing techniques [8], [22], [23]. We describe each of our noise generation techniques below.

1) Gaussian White Noise (WN): This technique uses the naïve approach of adding Gaussian noise to the original audio, serving as the baseline technique in our evaluation. Gaussian noise added is meant to simulate unknown noise. In a real environment, noise is typically caused by many sources, rarely arising from a single source. Thus, if we consider real noise as the sum of several independent random variables, each with its own probability distribution. Then according to the Central Limit Theorem, their normalized sum approaches a Gaussian distribution as the number of noise sources increase. Based on this assumption, the use of synthesized Gaussian noise is a simple and approximate simulation of real noise when the sources and their noise distributions are unknown. This technique directly adds Gaussian white noise to the original audio.

2) GL Reconstruction (GL): The essence of adding noise to an audio is to change its amplitude information. In the GL technique, we aim to add noise to the audio sample in such a way that it is not perceived. We use the masking threshold concept described in Section II-B to construct noise that will be masked by the original audio. The modified amplitude to reflect noise is then combined with phase information estimated by the Griffin-Lim algorithm to give the output perturbed audio signal. Steps involved are listed below,

1. Masking Threshold Calculation: The masking threshold is calculated in the frequency domain, so we change each frame of the given audio from the time domain to the frequency domain through a Short-Time Fourier Transform. This will result in amplitude and phase information for each audio frame. It is worth noting that the phase information is discarded for the next step and we reconstruct phase in Step 3. We then calculate the log-magnitude power spectral density (PSD) of each frequency bin on each frame. Then we use the method in [17] to calculate the masking threshold of every frame in a given audio.

2. Amplitude from Modified PSD: When we construct the noise, as long as the normalized PSD estimate of the noise is less than or equal to the masking threshold, the noise will be masked by the original audio. As a result, we use the calculated masking threshold (for each frame) as the PSD of the noise to be generated. We then compute the corresponding adversarial amplitude information for the noisy audio based on the modified PSD inversely [18].

3. Estimating Phase: As a final step we compute the phase information for the noisy audio. We do this using the Griffin-Lim algorithm described in Section II-C to estimate the phase information of the noise.

4. Synthesize Noisy Audio: We synthesize the noisy audio signal using inverse FFT over the estimated phase, computed in Step 3, and the masking threshold based amplitude from Step 2.

3) Original Phase (OP): The primary difference between the OP and GL technique is in Step 3 for estimating phase. Estimating phase using the GL algorithm introduces randomness and lack of consistency across multiple runs. To avoid this problem, the OP technique retains phase information from the original audio. We believe using phase information from the
original audio to synthesize the noisy audio will make the noise harder to detect which is the goal with our adversarial audio generation. Steps involved in the OP technique are shown below (differences with GL are italicized).

1. **Masking Threshold Calculation:** Same as Step 1 of GL technique, described in Section [III-B2] except that we save the phase information of the original audio sample as opposed to discarding it.

2. **Modifying PSD:** Same as Step 2 of GL technique.

3. **Synthesize Noisy Audio:** As with GL, we use inverse FFT to synthesize the noisy audio but using phase information from the original audio and the amplitude corresponding to the masking threshold.

### C. Stage 3: Combining Original and Noisy Audio

In this final stage of adversarial audio generation, seen in Figure [2], we take the selected frames in the original audio (identified in Stage 1) and replace them with corresponding frames from the noisy audio (generated in Stage 2). Other frames from the original audio are kept unchanged in the adversarial audio. This combination forms the adversarial audio output.

The source code for our technique, with the three noise generation and three frame selection methods, to generate adversarial audio can be found at https://anonymous.4open.science/r/lalalala-54BF.

### IV. Experiments

We evaluate the effectiveness of the techniques in Section [III] for generating adversarial audio samples using three different datasets and three different ASRs. The three audio datasets include, (1) The first 1000 audio samples of Librispeech [21], (2) The first 300 audio samples of Commonvoice [6], and finally (3) 300 audio samples randomly selected from Timit [10]. The ASRs used are Deepspeech [13], Sphinx [16], and Google [1].

Our choice of datasets and ASRs were inspired by their use in related work for adversarial ASR attack generation [2] [3] [8] [22] [29]. We discuss the evaluation metrics to measure effectiveness of our techniques and the research questions in our experiments in the rest of this Section.

### A. Evaluation Metrics

We use three metrics to measure the effectiveness of our techniques for generating adversarial audio samples – Word Error Rate (WER), Similarity and Success Rate. We are interested in generating adversarial audio samples that sound the same as the original audio (high Similarity) but produce an output text different from the original (high WER). Adversarial audio samples that meet this criterion help point to potential erroneous behaviour in ASRs as similar sounding audio is expected to produce similar text outputs. Success Rate is used to measure portability of generated adversarial audio samples across several ASRs.

- **a) WER:** is a common metric to evaluate the difference in output translated text generated by the ASR for original versus adversarial audio [9] [14]. WER is computed using Equation (1).
  
  \[
  \text{WER} = \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Total Words in Correct Transcript}} \tag{1}
  \]

- **b) Similarity metric:** is used to estimate the extent to which two audios sound the same to the human ear. The MFCC feature extraction process, briefly described in Section [II-A2], imitates the audio feature extraction process of the human ear. We use cosine similarity between MFCCs of the two audios to estimate audio similarity, shown in (2) and also previously used by [5].

  \[
  \text{Similarity} = \text{cosine_similarity(MFCC}_A, \text{MFCC}_B) \tag{2}
  \]

  \[
  \text{MFCC}_A \text{ represents the MFCC features of the audio } A, \text{ and MFCC}_B \text{ represents the MFCC features of audio } B.
  \]

  \[
  \text{Similarity} = \text{cosine_similarity(MFCC}_A, \text{MFCC}_B) \tag{2}
  \]

- **c) Success Rate:** shown in Equation (3), refers to the ratio of adversarial samples that can successfully attack a given ASR. A successful attack happens when the adversarial sample results in a non-zero WER with respect to the original output text. This definition of successful attack was used by Abdullah et al [2].

  \[
  \text{Success Rate} = \frac{\text{Number of successful attacks}}{\text{Total number of adversarial samples}} \tag{3}
  \]

### B. Research Questions

We aim to answer the following research questions (RQs) in our experiments,

- **RQ1:** Which frame selection method among Random, Important, All performs best?

  We compare the WER and Similarity achieved by the different frame selection techniques for adversarial audio generation across three different ASRs, with each ASR using three different input audio datasets. Answering this research question will help us assess the value of selecting a subset of frames versus just changing the whole audio.

- **RQ2:** Which noise generation technique among WN, GL, OP performs best?

  We compare the WER and Similarity achieved by the different noise generation techniques across three different ASRs and three different input datasets. Answering this question will help us measure the benefits of using GL or OP that uses the masking threshold over the naïve WN method that introduces Gaussian noise.

- **RQ3:** Are the adversarial samples portable across ASRs?

  One of the primary selling points of our techniques for generating adversarial audio is that they are blackbox and untargeted, and therefore agnostic to the structure and workings within ASRs. We validate this by evaluating the Success Rate of the generated adversarial samples using different noise generation methods across three different ASRs.

- **RQ4:** Does our blackbox untargeted technique perform better than a whitebox targeted technique recently proposed by Carlini et al. [8]?
Owing to the absence of available blackbox untargeted techniques that we can directly compare against, we instead use a highly cited whitebox targeted technique by Carlini et al. [8] that has not been fully surpassed by any other whitebox technique thus far. Additionally, Carlini et al. [8] have been compared against in similar articles [22] [17] [23]. Carlini’s whitebox targeted method for generating adversarial audio is based on the Deepspeech ASR and uses Commonvoice as the input dataset. To enable comparison, we use results from our technique for the same ASR and input dataset. Owing to the targeted nature of their technique, they require the text output to be used as target to be specified in advance. We, therefore, use the text output from our adversarial samples as the targets in their technique. We then compare our technique with Carlini et al. with respect to time taken to generate adversarial audio samples, Similarity to original audio, and Success Rate on other ASRs, Google and Sphinx. Since the text outputs in both techniques are the same, it does not make sense to compare WER.

We use Google Colab Pro with two NVIDIA Tesla T4 GPUs (16GB RAM, 2560 cores) to run our experiments.

V. RESULTS AND ANALYSIS

We present and discuss the results from our experiments in the context of the research questions presented earlier. To answer RQ1 and RQ2, we show the WER and Similarity achieved by three different noise generation technologies and three different frame selection technologies across the three ASRs in Figures 3, 4, and 5 for input datasets – Liberispeech, Commonvoice and Timit, respectively.

A. RQ1: Frame Selection Comparison

The best performing frame selection technique is one that achieves high WER and high Similarity. However, these two metrics are often conflicting. We discuss and compare WER and Similarity achieved by the three frame selection techniques in our approach below.

a) All frames: We find in Figures 3, 4, 5 and 6 that the All frame selection achieves the highest WER and lowest Similarity compared to Important and Random across ASRs, input datasets and noise generation methods. This is in line with our expectations as the other two frame selection techniques select a small part of the audio to introduce noise into achieving higher Similarity to original audio and smaller change in output text.

b) Statistical Analysis: We confirmed the statistical significance (at 5% significance level) of the observed differences between the frame selection techniques using one-way Anova and Tukey’s Honest Significant Difference (HSD) test. Tables I and III list the P-values for pairwise comparisons of WERs.
and Similarities, respectively, between frame selection techniques while fixing the noise generation technique to OP. P-values for pairwise comparisons using other noise generation techniques can be found in a supplementary document\footnote{https://anonymous.4open.science/r/lalalala-54BF/SupplementaryResults.pdf} owing to space constraints. Table III does not show different ASRs as the adversarial audio samples are agnostic to the ASR used. For all combinations, we find the All frames selection technology is significantly better than Important and Random for WER while being significantly worse in terms of Similarity.

c) Important versus Random: For most combinations of ASR, dataset and noise generation, we find Random frame selection produces the lowest WER and the highest Similarity, while Important frame selection results in a WER and Similarity between Random and All. We find that selecting important frames is an effective approach for generating adversarial audio that produces a significant difference in WER, as opposed to Random, while still producing audio very similar to the original. It is worth noting that we found that on the Timit dataset, there is no significant difference between Random and Important frames selection for both WER and Similarity as seen in Tables I and III. This is because for audio samples in Timit, the number of frames selected by Important and, therefore, Random is much smaller, an average of 7% of the total frames in each audio. As a result, the proportion of noise introduced and its effect are very small, making it difficult to differentiate between the two frame selection techniques. This is also confirmed by the very high Similarity values observed for both techniques in Figure 5.

With Google ASR and Commonvoice dataset, no significant difference in WER is observed between Important and Random. We believe this is because the important frames selected is based on the text output of Deepspeech rather than Google ASR. We did this to reduce the overhead in our experiments. Owing to this mismatch, the selected frames replaced with noise may have been given lower attention causing a smaller than expected difference in WER. We plan to select important frames based on the evaluated ASR, rather than just Deepspeech, in our experiments in the future.

d) Summary: In terms of WER, we find All frames performs best. However, Important and Random frames perform better in terms of Similarity. Our approach offers a choice of frame selections and the choice is left to the user based on the acceptable tradeoff between WER and Similarity. We find Important is a good choice that achieves reasonable performance in both WER and Similarity.

B. RQ2: Noise Generation Comparison

We compare WER achieved by WN, GL, OP using different ASRs and different input audio datasets in Figures 3, 4 and 5. We also compare Similarity of the generated adversarial audio samples to the original audio using WN, GL and OP across input audio datasets in Figure 6. Best performing noise generation technique is one that results in a high WER and high Similarity to original audio.

a) Performance of WN: We find the WN noise generation technique performs the worst in terms of WER as seen in Figures 3, 4 and 5. It also achieves the worst Similarity for the Librispeech dataset. WN is our baseline technique that
simply adds Gaussian noise and the added noise does not reliably change the text output while maintaining closeness to original audio. WER on Google ASR is consistently low with WN across datasets. This is because Google is effective at filtering out Gaussian white noise causing little change in output text.

b) Performance of OP, GL: These two techniques have a significant improvement in WER when compared to WN, confirmed with pairwise comparison using one-way Anova followed by Tukey’s HSD test as seen in Table IV. WER achieved over Google is generally lower than the other two ASRs. We believe this is because Google is more effective over other ASRs at filtering out noise.

Similarity to original audio is superior with GL, OP when compared to WN using the All frame selection for all three audio datasets. P-values for pairwise comparison of similarity using one-way Anova followed by Tukey’s HSD test is shown in Table IV. For the Timit dataset, with Important and Random frame selections OP and GL have no significant difference in similarity compared to WN. As mentioned in RQ1, the number of frames selected in Timit with Important and Random is very few, only amounting to 7%, so the effect of noise added is not significant. Overall we find, both GL and OP produce adversarial audio that is similar sounding to the original sample whilst producing WERs for the output text that is higher than WN.

c) OP versus GL: We find OP produces better WER than GL on the Librispeech dataset but not on the other datasets. The techniques are not significantly different in their similarity for all three datasets. The benefit with using OP lies in its use of reliable and stable phase information within an adversarial sample. On the other hand, GL will produce slightly different phase information for the same audio sample in different runs.

d) Summary: OP and GL are superior techniques for adversarial audio generation when compared to WN, as they produce noisy audio very similar to the original (average similarities are 0.88 and 0.87, respectively) and the WERs of OP and GL are 0.74 and 0.71, respectively, for all frame selection across ASRs and datasets.

C. RQ3: Portability across ASRs

We evaluate portability of the adversarial audio samples generated by OP, GL, WN across the three ASRs using the Success Rate metric, described in Section IV-A. Table V presents Success Rates achieved by these three techniques across different ASRs with different input datasets.

Compared to WN baseline (average ranging from 43% to 86%), the Success Rate of adversarial samples generated by GL (average of 93% to 98%) and OP (average of 97% to 99%) techniques is significantly higher (confirmed with permutation test). Average Success Rates for each technique across ASRs within each dataset can be found on the last row of Table V. With Google, Success Rates for WN are low as it is effective at filtering out Gaussian noise. In contrast, OP and GL produce high Success Rates with the Google ASR, demonstrating their robustness.

Across datasets and ASRs, OP and GL produce fairly high average Success Rates across datasets (97.9% and 95.7%, respectively). Between them, OP does better than GL, although the magnitude of difference is small. Based on these results, we can confidently state that our approach using GL and OP produce adversarial samples that are portable across ASRs with high Success Rates.

D. RQ4: Comparison to Existing Technique

As mentioned in Section IV-B we compare performance of our approach against a whitebox targeted technique proposed by Carlini et al. [8]. We fix the ASR to DeepSpeech and input dataset to CommonVoice to match with Carlini et al. [8]. For comparison, we use OP for noise generation with Important and All frame selections as the two best performing techniques in our approach for Similarity and WER, respectively. We show results for all techniques measuring generation time for adversarial samples, Similarity to original audio and Success Rates of the samples in attacking two other ASRs, Google and Sphinx, in Table VI. We do not compare WERs as the target text for Carlini et al. [8] is the translated text output from our adversarial samples, so there will be no difference.
We find adversarial generation using Carlini’s technique is extremely slow and takes 5× and 222× longer than OP+Important and OP+All, respectively. We achieve a slightly higher Similarity of 95% with OP+Important and a slightly lower Similarity of 86.6% with OP+All compared to 91% by Carlini et al.

To evaluate portability of adversarial samples onto other ASRs, we transcribe their adversarial samples on Google and Sphinx. We find that with Google ASR, adversarial samples generated by Carlini et al. has a 33% Success Rate while our techniques achieve a 70% Success Rate for OP+Important and a 91% Success Rate for OP+All. For Sphinx, compared to 77% attack Success Rate with Carlini et al., we reach a 89% Success Rate for OP+Important and a 99.6% Success Rate for OP+All. This is because the attack method in Carlini et al. is specifically designed for the neural network inside Deepspeech, so their adversarial samples are not as effective when used on other ASRs. This is a drawback also seen in other whitebox attacks. However, since our method is not designed for any specific ASR, we can successfully port our adversarial samples to other ASRs.

Across all three evaluation metrics, we find our techniques using OP+Important and OP+All are effective. OP+Important is superior to Carlini et al. [8] across all three metrics. OP+All shows significant gains in generation time and Success Rate but at the cost of Similarity which is slightly lower than Carlini et al. If maximum Similarity is not a requirement, then OP+All may be a better choice than OP+Important for reasons of generation speed and Success Rate.

### VI. Related Work

|                  | untargeted | targeted | white-box | black-box |
|------------------|------------|----------|-----------|-----------|
| Vaidya [26]      | Yes        | Yes      |           |           |
| Carlini (2016) [7]| Yes        | Yes      |           |           |
| Carlini (2018) [8]| Yes        | Yes      |           |           |
| Qin [22]         | Yes        | Yes      |           |           |
| Zhang [29]       | Yes        | Yes      |           |           |
| Yuan [28]        | Yes        | Yes      |           |           |
| Yakura [27]      | Yes        | Yes      |           |           |
| Schönherr [24]   | Yes        | Yes      |           |           |
| Szurley [25]     | Yes        | Yes      |           |           |
| Abdullah [2]     | Yes        |           | Yes       |           |
| Abdullah [3]     | Yes        |           | Yes       |           |
| Khare [15]       | Yes        |           | Yes       |           |

|                  |            |           |           |           |

**TABLE VII**

**EXISTING WORK ON ADVERSARIAL ATTACK GENERATIONS FOR ASRs**

Adversarial attack generation on the ASR can be classified along two dimensions: 1. Targeted for a given text output or untargeted, and 2. Whitebox, with respect to the NN within the ASR or Blackbox. In targeted attacks, the attacker sets the text to be translated from the adversarial sample in advance, while untargeted means the attacker does not use any knowledge of output text for the attack. When the attacker has knowledge of the internal structure and parameters of the DNN stage of the ASR system, this attack is called a whitebox attack, versus a blackbox attack when no DNN knowledge is used. The techniques for generating adversarial samples is listed in Table VII along with their classification along the two dimensions. As can be seen, much of existing work uses targeted whitebox attacks, different from our untargeted blackbox approach.

#### A. Targeted Attacks

Vaidya et al. [26] pioneered the first method for attacking ASR in 2015. Given the translated text that needs to be obtained, they gradually approach the target text by continuously fine-tuning the parameters of the extracted MFCC features. Once the goal is reached, they use the obtained adversarial MFCC features to reconstruct the speech waveform. On the basis of Vaidya’s work and in an effort to improve the efficiency of their approach, Carlini et al. [7] proposed Hidden Voice Command in 2016, adding noise that is often encountered in real life. However, neither of these two types of attacks can conceal the existence of noise, and such adversarial samples can be easily recognized as random noise rather than effective commands. ASRs like Google are effective at filtering out random noise as we saw with the WN technique in our experiments.

Zhang et al. [29] in 2017 leveraged the human ear’s insensitivity to ultrasound to create an ultrasonic attack method called DolphinAttack that cannot be perceived by the human ear. Their method modulates the voice on the ultrasonic carrier to make it into a command that humans cannot hear and then inserting such a command into the original audio. Such commands are set in advance, such as “Hi, Siri”, “Open the window”, making this method a targeted attack. However, this method is not easy to reproduce as it uses hardware characteristics of the microphone to complete the attack.

Yuan et al. [28] proposed a method for embedding commands into songs so that when these songs are played, the commands will be translated by an ASR, but it is difficult to detect changes in audio. Additionally, they improve the robustness of adversarial samples by introducing noise generated by hardware devices. This approach, however, is restricted to songs as the carrier of commands, and is, therefore, limited in application scenarios.

Carlini et al. [8] in 2018 used gradient descent to modify the original audio so that the difference between the translated text and the target text is smaller. Their method is targeted and whitebox. Their experimental results show their attack Success Rates reached 100% on Deepspeech ASR and Commonvoice data set. However, their approach is faced with some limitations: First, it can take up to several hours to generate attacks; second, the gradient descent method requires the attacker to have a good understanding of all the internal parameters and structures of the attacked system before it can be used; the third is that the samples generated will be invalid when other ASRs are used; and finally the fourth is that prior information on targeted text is required.

Yakura et al. [27] proposed some improvements to [8] to maintain attack performance under over-the-air conditions.
(mixed with sound of the surrounding environment). They generate adversarial examples accounting for noise caused by echo and recording in real life, so as to obtain more robust adversarial examples. However, other shortcomings in Carlini et al. \cite{8} (such as long generation time and weak transferability) have not been addressed.

In 2018, Schönherr et al. \cite{24} applied the knowledge of masking threshold to generate adversarial samples against ASRs. They proposed to limit the generated noise below the masking threshold of the original audio to ensure that the obtained perturbation is not audible to the human ear. In more recent work \cite{23}, they introduced room impulse response (RIR) simulator to improve the robustness of samples that produces different types of noise for different environment configurations. Their work is targeted and whitebox, constructing different loss functions based on the internal network structure of the ASR for a target text selected in advance.

Inspired by Schönherr et al., Qin and Carlini et al. \cite{22} developed a whitebox method and optimized perturbations to make it lower than the masking threshold of the original audio. This method achieved a 100% attack Success Rate on the Lingvo system. Like other whitebox targeted approaches, their work lacks portability to other ASRs and is time consuming for attack generation.

Around the same time, Szurley et al. \cite{23, 25} proposed a whitebox method similar to Schönherr et al. \cite{23, 24} and Carlini et al. \cite{8, 22} that constructed an optimization based on masking threshold and combined it with room reverberation. Their method reached a 100% Success Rate on Deepspeech but still suffers from limitations of lack of portability and time consuming attack generation.

### B. Untargeted Attacks

In 2018, Khare et al. \cite{15} proposed a blackbox method that can be used for both targeted and untargeted attacks based on multi-objective optimization. Specifically, they used genetic algorithms to implement blackbox attacks, and tried to find the mutated individuals that best meet the two goals of increasing transcribed text differences and reducing audio differences as adversarial samples. Mutated individuals were generated by adding white noise. As seen from our experiments, white noise based approaches are not effective on ASRs like Google that filter them out effectively. Their evaluation is not extensive, only using 100 audio samples from Commonvoice. We plan to use genetic algorithms to select adversarial samples that can optimise both WER and Similarity in the future. It is worth noting that their code was not available to facilitate comparison and the authors were unresponsive to our request.\footnote{They have made 20 adversarial samples available but this was not enough for our comparison.}

In 2019, Abdullah et al. \cite{2} proposed another untargeted blackbox attack method in 2019. They constructed an adversarial sample by decomposing and reconstructing the original audio. Specifically, they removed the low-intensity components in the audio and retained the relatively high-intensity components. Their method reduced the time to generate adversarial samples to a few seconds. However, they did not evaluate on different input datasets, and the extent of difference between the translated text of adversarial samples and the translated text of the original audio is unclear. For example, if the original translation of an audio is “I love apples” and the translation of an adversarial sample is “I lovee apples”, the difference between the two translated texts is small. However, in their experiments, they record it as a successful attack without measuring WER that gives a measure of text output difference. Compared with their experiments, we explore the performance of our method on different audio datasets, and we were able to generate adversarial samples with an average WER of 74% (with OP+All). Owing to lack of available code and adversarial samples for Abdullah et al. \cite{2, 3}, despite reaching out to them, we have not been able to compare with their methods.

### VII. Conclusion

We proposed a blackbox untargeted adversarial attack generation technique for ASRs that introduces perturbation in the signal processing stage such that the adversarial audio is similar to the original while producing a change in the output text. Our approach provides three noise generation approaches — WN, GL and OP. We also provide the option of selectively introducing perturbations to a small fraction of audio frames using three frame selection options — Random, Important and All. Evaluation of our techniques over three ASRs and three input datasets showed that our techniques can be effective at achieving high WERs (average of 71% with OP+All) while also achieving high Similarity (average of 95% with OP+Important). The choice in noise generation and frame selection helps achieve a good balance between these two metrics. We also confirmed that our techniques were portable across ASRs with high Success Rates (average over 98% with OP+ALL) and superior to existing whitebox targeted technique \cite{8} in terms of Similarity, Success Rate and attack generation time. In the future, we will explore defense methods for adversarial attacks, inspecting both the signal processing and the DNN stage within ASRs.
