Alexander Khosrowshahi, William Stone, Ross Williams, and Saswata Majumder

Team YELL!

Black-box Adversarial Learning against Automated Audio Speech Recognition (ASR) Models

01 Introduction

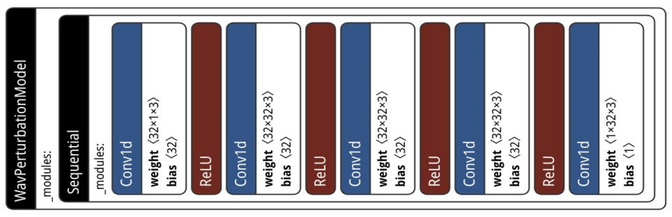
YELL is an adversarially trained convolutional neural network designed to disrupt audio speech recognition (ASR) models. Though ASR models serve many important functions in technology today (automatic closed captioning, Siri, etc.), they can be used for nefarious purposes—whether eavesdropping or mass-scraping online audio content without creator consent.

In an effort to give users agency over the usability of their audio/speech data, we created YELL. YELL takes an audio sample and generates a noise filter to be reapplied to the sample, designed to render speech samples uninterpretable to ASR models. Secondarily, the noise filter attempts to be as unobtrusive to human listeners as possible, though our results varied.

02.1 Methodology - Architecture

YELL employs an extremely simple perturbation model architecture, utilizing a series of identical 1D CNN-ReLU layers. Our working hypothesis is that a more complex architecture would hurt more than it would help.

It is worth noting that our network only learns perturbations, so any layers you might expect to see in an ordinary CNN (max pooling, batch normalization) are excluded.



02.2 Methodology - Data

We trained using the LibriSpeech ASR Corpus, a widely used dataset of approximately 1000 hours of 16kHz read English speech (V. Panayotov, G. Chen, D. Povey and S. Khudanpur), a commonly used training set for ASR tasks.

Our training involved running unperturbed LibriSpeech samples through whisper, adding the perturbation noise filter, and comparing the outputs to determine adversarial reward.

02.3 Methodology - Training

Evolutionary Strategies

Evolutionary Strategies (ES) is a popular optimization technique and one particularly attractive for black-box adversarial learning. Unlike reinforcement learning, it assumes nothing about the world state, actions, or even model. It only needs a reward function.

In evolutionary strategies, we start with some parameters, jitter them around with random noise, and then take the weighted sum of the results of our parameter jitter, weighting those that received a higher reward more. These then become our new parameters, and we do it again. It is incredibly simple and works well for black-box optimization where the only feedback is our designed reward.

Our Method

We employ the ordinary steps of ES optimization, but include a number of novel additions based on the goals of 1) high ASR error and 2) retained interpretability. For our attack, we targeted OpenAI's Whisper, a leading open-source ASR Model (though we treated it as a black box).

Reward Design

Whisper's API forward call allows for the exposure of logits to the user. As such, we decided to use the logit entropy of whisper as our reward function, given a higher entropy will generally lead to a more "confused" model.

This gave us a basic reward function of:

$$H_i = -\sum_t p_t(x_i)\ \log p_t(x_i)$$

That said, this reward function alone will incentivize the model to perturb audio beyond human interpretability. As such, we add in penalty and reward factors.

First, we penalize large jumps in loudness by penalizing the mean squared change in loudness between our clean and perturbed waveforms.

$$D_i = \frac{1}{T}\sum_{t=1}^{T}\big(x_i(t) - x(t)\big)^2$$

Further, we sculpt a reward based on the highest and lowest frequencies of our waveforms, encouraging perturbations in the very high areas of the audio spectrum (which are less vital to human intelligibility) and detracting those in the lower half (which often correspond to more overdriven/distorted perturbations). This gives the model an additional incentive based on insights into human audio processing.

$$\Phi_i = \frac{\sum_{f>f_{\text{vh}}} \left|\text{STFT}(x_i)_f\right|}{\sum_{f\le\frac{F}{2}} \left|\text{STFT}(x_i)_f\right|}, \qquad f_{\text{vh}} = \lceil 0.95\,F \rceil$$

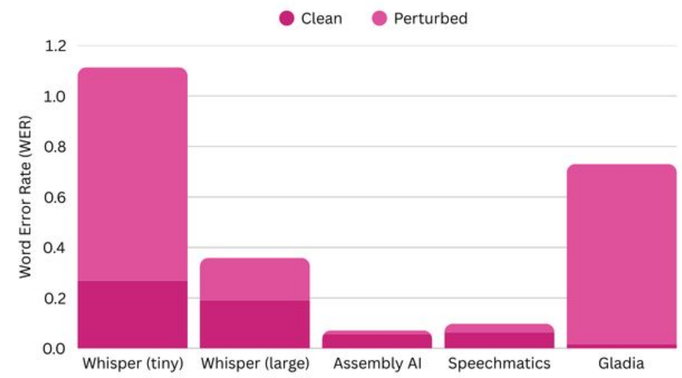
Our final reward is given by:

$$R_i = \alpha\, H_i \ - \ \beta\, D_i \ + \ \gamma\, \Phi_i$$

From which we pick the top k performers (user defined) and update our weight parameters after normalization. This all yields a training cycle in which perturbations become more suited toward exploiting Whisper but are disincentivized from perturbing audio in ways that would make it unintelligible.

03 Results

Word Error Rate Across Several ASR Models



Unsurprisingly, our adversarial model performs best on the model we targeted in training (Whisper Tiny), but it is interesting to note the outsized impact it had on Gladia, especially noting Gladia performed best on the original, clean audio. Unfortunately, though, the attack did not transfer over well to Assembly AI or Speechmatics, presenting an interesting juxtaposition to Gladia and highly mixed results with respect the transferability of our attack.

Comparison to Ordinary Filters

While our attack architecture serves to create difficulties for Speech-to-Text models, simple noise filters come to mind as a much easier alternative than training an adversarial model. We implemented a testing architecture to add Gaussian noise to our audio samples, and compared the **WER** delta between a simple noise filter and our perturbation model. Generally, **our perturbation model had a higher WER delta**, meaning that the models we ran our tests on performed less accurately under our perturbation filter as compared to random noise. This indicates that our perturbation model was in fact learning ways to attack the outputs of speech-to-text models, potentially attacking specific features of an utterance rather than obscuring the audio as a whole.

Human interpretability is ultimately subjective. However, our audiences agreed that audio perturbed by our model was more pleasant to listen to than the noisy audio. Our perturbation model produces audio akin to "someone with a bad mic," while noise filters are much more harsh and obvious in their presence. Therefore, while we acknowledge perceptibility of our perturbations, we believe them to be more amenable to human perception than a simple filter, while also being more effective.

04 Discussion

Highly Adversarial // Limited Interpretability

By far our main difficulty with YELL was interpretability. It seems that, though ASR models like Whisper do not attach to the *exact* same features of speech that humans do for interpreting speech, they both have some correlation to distortions in similar bands. Therefore, if we incentivized model performance degradation, that usually coincided with a direct decrease in human interpretability.

Future Work

With more time, there are several modifications to our approach would have liked to have explored. Firstly, it would have been interesting to see whether changing our model size or architecture would affect interpretability or increase word error rate. Secondly, we would have liked to play more with the many hyperparameters in our learning process; we still feel as though the optimal combination eludes us. Lastly, it would have been interesting to see our model improve with more training (black box optimization requires a lot!).