YELL!

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

Alex Khosrowshahi
alexander_khosrowshahi@brown.edu
akhosrow

William Stone
william_stone@brown.edu
wlstone

Ross Williams
ross_williams@brown.edu
rjwillia

Saswata Majumder
saswata_majumder@brown.edu
smajum14

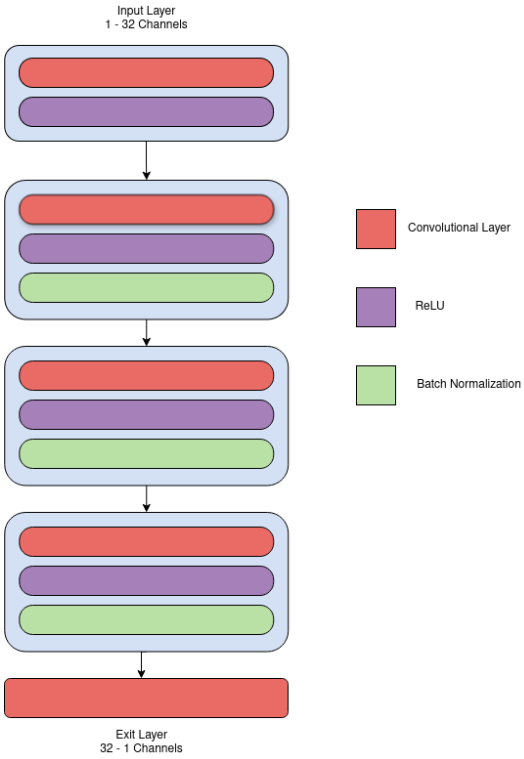
1 Introduction

Audio speech recognition (ASR) models serve many important functions in technology today: automatic close captioning, voice commands, etc. However, they can also easily be used for nefarious purposes, from scraping online audio content without creator consent to large-scale automated surveillance. In an effort to enforce data privacy and give people agency over their audio data, we created YELL - an adversarially-trained convolutional neural network designed to disrupt ASR models. YELL takes an audio sample and generates a noise filter to be applied to the sample, designed to render the hardened sample uninterpretable to ASR models while attempting to be as unobtrusive to human listeners as possible. Though our results varied across models, we were able to achieve noticeable degradation of model performance in most cases without completely losing interpretability (though audio was significantly degraded) with higher performance than analogous ordinary noise filters.

2 Methodology

2.1 The Model

YELL employs an extremely simple perturbation model architecture, using a series of identical 1D CNN-ReLU layers followed by batch normalization for training stability. Part of our philosophy around model simplicity was that we intended to learn noise filters rather than extracting particularly meaningful features from our data.



2.2 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.

In 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).

2.3 Reward Design

2.3.1 Logit Entropy

Whisper's API forward call allows for the exposure of logits to the user. In information theory, entropy is a measure of disorder or randomness in a closed system. We decided that by maximizing the entropy of the logits outputted by Whisper on a forward pass, we could easily "confuse" the model into mistranslating audio sequences to text.

Entropy is given by the equation

$$H = -\sum_{t} p_t(x_i) \log p_t(x_i)$$

Where p_t is the logit entropy for a given audio tensor x_i at t

The issue with such a reward metric is that it incentivizes large and confusing perturbations, which often results in extremely blown-out noise. To compensate for this, we add two additional parameters to our reward function.

2.3.2 Distortion Penalty

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

$$D_i = \frac{1}{T} \sum_{i=0}^{T} (x(t)^c - x(t)^p)^2$$

Where $x(t)^c$ is the clean waveform and $x(t)^p$ is the perturbed waveform. This incentivizes overall perturbations to be smaller, hopefully incentivizing the model to learn more nuanced disruptions than random noise.

2.3.3 KL Divergence

We integrate a KL divergence penalty to incentivize analogous distributions between our clean and perturbed audio. Training with this term has revealed large increases in Short Term Object Intelligibility (STOI) which seems to suggest its effectiveness, though entropy does not seem to markedly increase or decrease with its introduction.

$$\mathrm{KL}(x^p \| x^c) = \frac{1}{B} \sum_{b=1}^{B} \sum_{f} \tilde{P}_{bf} \left(\log \tilde{P}_{bf} - \log P_{bf} \right)$$

2.3.4 Deprecated Strategies: High Frequency Incentive

We previously tried to integrate insights from audio processing, incentivizing high frequency perturbations less likely to be heard by humans. That said, this strategy seemed less effective with further testing as it seems that Whisper and other ASR models are most affected by human speech bands on which we were trying to reduce perturbations. Instead, we focused on having the model learn subtler features.

2.3.5 Wrapping it up

This overall gives us a reward function

$$\text{Reward} = \lambda_{\text{adv}} \cdot H(\text{logits}) - \lambda_{\delta} \cdot D_i - \lambda_{\text{KL}} \cdot \mathrm{KL}(x^p \| x^c)$$

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.

3 Results

To measure our results, we used the metric of Word Error Rate (WER). WER measures the differences between strings by counting the number of substitutions, deletions, and insertions performed to transform one string to another. The ultimate result is a proportion:

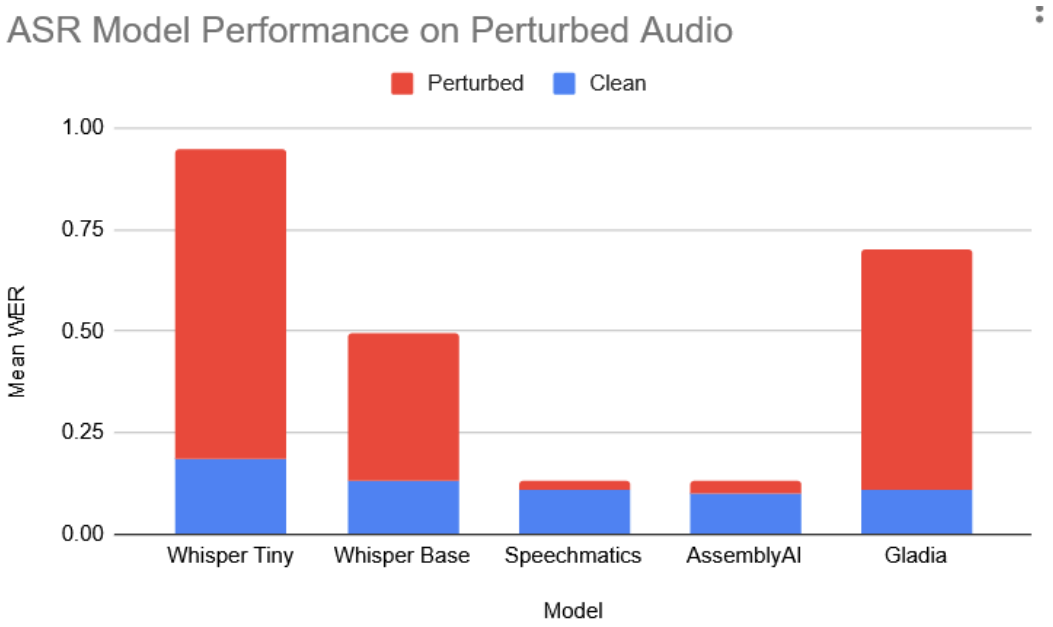
$$WER = \frac{S + D + I}{S + D + C}$$

Where S is the number of substitutions, D is deletions, I is insertions, and C is correct words from the reference (thus, $S + D + C$ is the total number of words in the reference string). A value of 0.5, for example, means there is a 50% error rate between our string and the reference string.

To test our models, we performed the following steps:

1. Grab a batch of audio files and transcripts from our test data set.
2. Run the clean audio files through an ASR model, and evaluate its WER when compared to the test transcripts, averaged over the batch.
3. Run the clean audio files through our perturbation model, acquiring perturbed audio files.
4. Run the perturbed audio files through the same ASR model, and evaluate its WER when compared to the test transcripts, averaged over the batch.
5. Calculate the delta between these two mean WERs: the higher the delta, the better our model performed its goal of worsening the ASR target model's results.

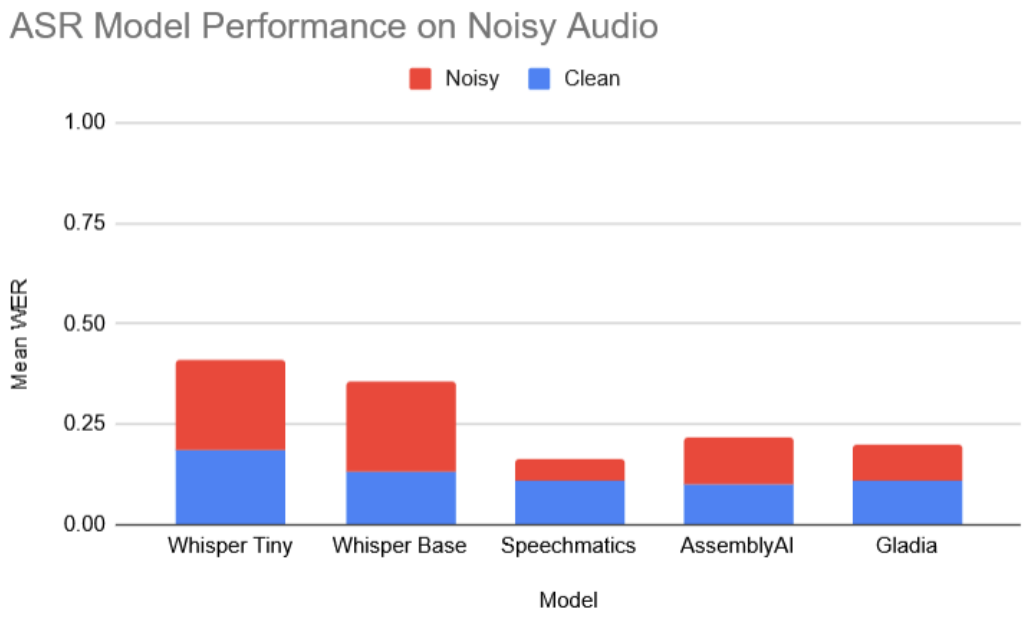
The results are as follows:



Our adversarial model unsurprisingly performed best on the model we targeted in training—Whisper's tiny model. It is quite interesting, however, to note the outsized impact our model had on Gladia. Due to Gladia performing the best on our clean audio, we were surprised to see how well our perturbations performed. We hypothesize this could be due to Gladia itself overfitting to our dataset, LibriSpeech, and thus being very good at predicting LibriSpeech but extremely weak to perturbations.

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 to the transferability of our attack.

Another facet of our analysis was ensuring that our model was actually performing a non-trivial task: 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 thus decided to test our implementation against a simple random Gaussian noise filter, to compare both audio quality and impact on ASR models. Our testing architecture was precisely the same as [insert section number talking abt process above], but step 3 was replaced with a noise filter rather than our perturbation model. We ran these tests on our models, with results as follows:



Generally, our perturbation model had a higher WER delta than straight Gaussian noise, meaning that the models we ran our tests on performed less accurately under our perturbation filter, which are results we wanted to see. 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. Interestingly, Speechmatics and AssemblyAI performed worse with Gaussian noise, highlighting a potential shortfall of our model with these specific ASR architectures.

Finally, we wished to maintain human interpretability in our perturbed audio, or in other words minimize perceptibility of our perturbations themselves. This metric is ultimately subjective; however, the audiences we presented our audio to agreed that audio perturbed by our model was more pleasant to listen to than audio with a noise filter applied. Our perturbation model produces audio akin to "someone with a bad microphone," whereas noise filters were said to be much more harsh and obvious in their presence. Therefore, while we acknowledge the perceptibility of our perturbations, we believe them to be more amenable to human perception than a simple filter, while also being more effective.

4 Challenges

The journey to get to this final product was not a smooth one. When we first conceived of the idea of working on this project, we perhaps didn't fully appreciate just how much work would be involved in getting it off the ground in the first place. Beyond the somewhat standard tasks of determining a proper scope for the project, acquiring data, deciding on a model architecture, figuring out how to run jobs of Oscar, etc., we also had to, among other things, familiarize ourselves with PyTorch, thoroughly research various black box learning techniques, educate ourselves on audio data and best ways to represent it for our purposes, and pore over the code in Whisper's GitHub repository (documentation for all but the most basic features of its API was nonexistent).

And it didn't magically turn into a cakewalk once we finally felt we were in a good enough position to begin coding. Since we were not working from a research paper to begin with and didn't have any super similar research papers to reference, there was a lot of two steps forward, one step backwards moments throughout as we sought to figure out the best direction to head in. One example of this was early on when we were constructing our preprocessing infrastructure. The leading research on audio preprocessing points to turning audio into log mel spectrograms, which are basically images of the amplitudes at different frequencies of the audio over time, causing us to build our preprocessing code around turning our audio into tensors of spectrograms. We later realized, though, that because our model would just be applying small perturbations on top of our audio, it would be simpler to implement, more compute friendly, and possibly lead to better results if we stuck with audio in its more standard time series form, causing us to have to go back and rework our preprocessing pipeline.

The preprocessing change-up was a relatively mild setback, as most of the original code was still usable with some modest changes, but not all of our two-steps-forward-one-step-back moments were. A more extreme example has to do with the selection of our attack strategy. Originally, we had intended on pursuing a variety of attack strategies and comparing their efficacy, and as recent as last Friday, we were working on a Projected Gradient Ascent (PGA) white box attack and a Covariance Matrix Adaptation Evolution Strategy (CMAES) attack in addition to the standard ES attack that now holds the central role in our project. However, challenges with both other approaches (an odd bug with PyTorch's cross entropy function and Whisper's tokenizer for PGA and apparent non-learning possibly due to reliance upon an external library for CMAES) as we approached the deadline forced us to abandon these in favor of dedicating all of our energy to our ES attack.

Finally, once we settled upon our ES attack and were over the hump in terms of implementation uncertainty and project scope, we faced challenges with what quickly became the key tension of our project: optimizing jointly for both high *transcription failure* and respectable *interpretability*.

The difficulty with this was the design of our reward function. As put above, we wanted to take into consideration both preserving interpretability to humans and confusing whisper as much as possible, and we explored many avenues to do so. Our main issue was that we never seemed to stumble upon a method for which both goals were satisfied. We tried incentivizing certain bands outside of human speech primary ranges, but this seemed to have the model converge to barely any entropy change at all. We tried penalizing distortion (which we still do), but this is still relatively coarse grained, and it seems that finer-grained metrics exhibit similar behavior of converging to very small entropy changes. That all said, we were able to confirm our model was learning adversarial perturbations of some sense, as it performs better than naive noise masking strategies.

5 Reflection and Next Steps

How do you feel your project ultimately turned out? How did you do relative to your base/target/stretch goals?

Ultimately, we feel pretty good about what we were able to accomplish with this project. While we did only end up accomplishing the "base goal" we set for ourselves in the second check-in, we feel that we may have somewhat underestimated the difficulty of what we had originally set out to do and don't feel as though we came up short with what we were able to accomplish.

Did your model work out the way you expected it to?

As of now, our model does not live up to our highest of possible hopes. While it does produce audio with a very high word error rate on our target model (Whisper Tiny), the perturbations it produces

are still large enough that they are very clearly perceptible to humans and while it is not particularly difficult to hear what it is exactly that is being said, it does take some concentration. That being said, with incremental changes in approach here and tweaks in our hyperparameters there, we have certainly been able to step up our interpretability bit by bit, which is heartening to see. Additionally, as noted in Results, we were able to yield better WER and better interpretability than a random noise filter, demonstrating our model's ability to at least learn *something*, another heartening result.

How did your approach change over time? What kind of pivots did you make, if any? Would you have done differently if you could do your project over again?

Our approach changed a lot over time. Like A LOT (see Challenges). Some of the notable changes we can think of are as follows:

1. Preprocessing - *Spectrograms to time series data*
2. Attack Strategy - *PGA, ES, and CMAES to just ES*
3. Transcription Loss Modeling in Training - *WER to logit entropy*
4. Interpretability Strategy - *Reliance on "max δ" scaling to incorporation into our reward function*
5. Several Small Architecture & Reward Tweaks - *Adding BatchNorm to our model, culling non-top performers in evolution selection, etc.*

If we could do our project all over again, we would likely have decided to work just with basic ES from the beginning and devote all of our time and energy into thinking about way we could change our mode and reward function to increase intepretability without nerfing our model's ability to confuse the target model. I think we would also have tried to more explicitly split up the project's workload and stick to a project schedule because there was a fair amount of last minute scrambling when things took a little longer than we had hoped they would towards the project's end.

What do you think you can further improve on if you had more time?

We have a lot of ideas as to both ways we could improve our project or ways we could extend it. Firstly, the complexity of our reward function means that there were a lot of hyperparameters we had to play with, and right combination was (and is) not at all obvious to us. How much weight should we give the distortion minimizing term relative to the logit entropy maximizing one? How many members of our model population should we take into account when updating our model? Should these values change with time or scale with respect to other metrics? There are many unanswered questions with respect to hyperparamaters that we could explore and answer with time.

Beyond these hyperparameters, there's also a lot of aspects of our model and training setup that we think could be altered to improve performance/interpretability. Even after our poster presentation, small changes have been made that have led to better results; changes to using KL divergence instead of entropy and incorporating BatchNorm into our model's architecture (which we originally thought wouldn't be helpful to the relatively simple task of adding perturbations) have yielded interpretability gains. With more time, we would likely have still more ideas for how to adjust things here and there to bring about better results.

Additionally, more time to train would've been helpful. We never quite got OSCAR to work with us because it seemed to be canceling our jobs midway through the time we had allotted for them, so our best model only trained for 50 epochs, which is not a lot for an ES model.

Besides model improvements, there are many interesting directions that this project could be taken in to extend its findings. For instance, one student at DL day asked about the extensibility of our model to other languages/dialects of english. We could not provide much of an answer here because we restricted our training to "clean" english audio data in an effort to keep to a reasonable project scope, but this is an interesting idea; would our current model with the same set of parameters work equally well with American english audio as it might with other languages or dialects of english? If so, why? If not, how different do the model's parameters need to be and in what layers/areas? This question and others could be explored with more time.

What are your biggest takeaways from this project/what did you learn?

- **Alex |** My biggest takeaway from this project is that too much inertia is a killer. It's easy to get stuck on one idea and treat your model as a big messy data machine that you hope

throwing enough into will yield something, but oftentimes your core assumptions about how your model *should* work are not actually the path to success. Most of our marked improvements in our model performance came from 4:00 AM "what ifs?" that were just desperate enough to, if not get us moving in the right direction, get us *moving* at all.

- **Ross |** My biggest takeaway from this project is that while deep learning seems from the outside like a magical box that you just sort of fiddle with until it works, in reality intuition and educated guesses can light the way forward much more than you might have expected at first glance. At the end of the day, model tuning and achieving best results can still feel like a magical box at points, but when you really understand what you're doing, you don't feel *quite* as lost.
- **Sas |** My biggest takeaway was, similarly, the fact that intuition plays a huge role in the lifetime of a model. While it certainly did help to try different ideas in parallel to avoid getting stuck with a suboptimal model (kind of like evolutionary strategies!), our forays were usually informed to a good degree by intuition - of our own, and of the researchers who've come before us. It made me realize the creative aspect of deep learning - the art of thinking of a cool improvement, seeing that it works, and then realizing it worked for a completely different reason.
- **William |** My biggest takeaway from this project is that in order to experiment with deep learning, you don't necessarily need to be an expert in information theory or other specialized fields of higher mathematics; while that all certainly helps, even just having an intuition or simple idea about some small change that could be made to a learning process can dramatically improve the results you see. Because of this project, learning in general feels a little bit more approachable and not just something where I can either rely solely upon prebuilt pieces and frameworks or get two math degrees to meaningfully interact with.