Extending Whisper, OpenAI's Speech-to-Text Model

Group 21: Abhishek Divekar (ard3443), Yosub Jung (yj6579), and Roshni Tayal (rt26476)

Department of Computer Science
The University of Texas at Austin

ABSTRACT

We study the performance of OpenAI's Whisper model, the state-of-the-art Speech-to-text model, in noisy urban environments. To do so, we create a dataset consisting of 134 minutes of us reading out loud in both quiet and noisy urban environments (subway, street and cafe) and manually annotating the recordings at 30 second intervals. Using a powerful multi-GPU AWS cluster and distributed computing framework Ray, we find that Whisper performs significantly worse on speeches recorded in noisy environments than on those recorded in quiet environments, in contrast to assertions made by Whisper authors. This performance gap is particularly severe for small Whisper models. This finding is concerning since the small models, due to its low inference-time, are most likely to be deployed on handheld devices (like smartphones), and thus more likely to be exposed to outside noise that can degrade speech-to-text performance. To improve performance, we fine-tune the HuggingFace Whisper implementation on a split of our collected data. We find that fine-tuning on single-speaker noisy speech improves average Word Error Rate (WER) by 2.81 (from 28.76 to 25.95) and fine-tuning on multi-speaker noisy speech improves average WER by 2.61 (from 28.76 to 26.15). Thus we are able to successfully adapt OpenAI Whisper to function reliably in noisy urban environments.

1. INTRODUCTION

Reliable Automatic Speech Recognition (ASR) technology has a huge demand in countries around the world. For example, in the United States alone, 11.5 million adults report limited hearing abilities (United States Census Bureau, 2022). In addition, a huge portion of online speech content is in English, and ASR paired with machine-translation can again benefit users, e.g. through automatic generation of closed-captions for non-native speakers and those with limited hearing abilities.

Deep learning is energizing speech-to-text technology since Baidu's AI team led by Andrew Ng built a deep learning model in 2014 (Hannun et al, 2014). These models perform an

order of magnitude better than existing speech-to-text technology, suggesting the possibility of a reliable Speech-to-text via Deep learning. This research area was further excited by Open AI's recent ASR Deep Learning model, Whisper (Radford et al, 2022). According to Whisper authors, this model performs at-par with human experts on transcribing various English speeches to English texts, and also works for multilingual speech transcription and machine-translation.

Given the large latent demand for reliable ASR technology, especially in real-time situations, we have decided to further study the performance of Whisper. In particular, we focus on the model performance in a noisy urban environment, where many future users reside. A challenge is that in an urban environment, noises such as construction, traffic and general population sounds will pollute dialogues and conversations, decreasing the signal-to-noise ratio and potentially weakening machine learning speech-to-text models. Although the Whisper paper assesses the model performance in a noisy environment, the noise is artificially generated from the music recording setting. It is hard to know whether that analysis carries over to a real-world noisy urban setting.

We assess how Whisper performs in a noisy urban setting, fine-tune the model with noisy conversations data, and improve its performance. To that end, we collect a dataset of English speeches recorded in urban environments. The first comes from manual recording: the three of us (authors) read four English texts of different length out loud in various urban settings. We record both in "quieter" environments (home) and in "noisier" environments (subway, street and cafe), to contrast Whisper's performance with respect to noises.

We note that Whisper is an industry-grade model with sizes ranging from 39 million to 1.5 billion parameters, 60 times more parameters than ResNet50 which has only 25 million parameters (He et al, 2016). Thus, computation — both fine-tuning and inference — is non-trivial and requires accelerated hardwares. To that end, we spin up three AWS p3dn.24xlarge instances (having 8 NVIDIA V100 GPUs, each with 32 GB GRAM), and schedule fine-tuning jobs using the Ray distributed computing framework (Moritz et al, 2018).

Radford et. al. (2022) claim that Whisper performs high-quality zero-shot transcription in a wide range of environments. However, we observe that Whisper performance severely degrades on speeches recorded in noisy environments, than on those recorded in quiet environments. We find this performance gap is particularly severe for smaller models ("tiny" and "base" sizes). We demonstrate that the inference-time of small Whisper models is much lower

compared to larger models, and thus small models are most likely to be adapted on handheld devices, thus being more likely to be exposed to noisy environments. This motivates the need to bridge the performance gap between the noisy and the quiet speeches for small models, by fine-tuning the model weights on noisy speech. As Whisper is based on the Transformer architecture, fine-tuning will not cause an increase in inference-time, thus we get this performance benefit with no downsides.

To improve performance, we fine-tune the fastest Whisper model ("tiny") on the dataset we have manually collected. We fine-tune a custom model for each speaker and show that for speeches recorded in noisy urban environments (subway, street and cafe), we achieve an average 2.81 decrease in Word Error Rate (WER) from 28.76 to 25.95. This improvement was achieved using just 14.08 minutes of annotated speech data for each speaker. With more data, Whisper models may be even more performant on noisy recordings.

We additionally evaluate a "multi-speaker" model, wherein we fine-tune on noisy recordings from all speakers. We observe that this improves average WER by 2.61 (from 28.76 to 26.15), and thus does not outperform single-speaker fine-tuning.

The rest of the paper is organized as follows. Section 2 reviews notable speech-to-text deep learning models. Section 3 documents our data collection-efforts. Section 4 explains our method behind assessing the model performance and fine-tuning. Section 5 summarizes our findings. Section 6 concludes our findings.

2. LITERATURE REVIEW

Deep Learning application to speech-to-text came from Baidu's AI team led by Andrew Ng (Hannun et al, 2014; Amodei et al, 2016). Before 2014, speech-to-text systems were rule-based written by human programmers. The Baidu's Deep Speech model is a supervised model that encodes speech spectrograms using CNN (LeCun et al, 1989) and applies GRU (Chung et al, 2014) to encode and get text transcription. They train the model with Baidu's internal data consisting of 11,940 hours of English and 9,400 hours of Mandarin recordings. The model performs as well as humans on clean speech, easily beating the state-of-the-art rule-based models. For example, for European accented English data, the human error-rate is 17.55 WER while the model error-rate is 13.56 WER. However, the model underperforms on accented speech.

Another break-through came in 2019 and 2020 when Facebook released its model Wav2Vec (Schneider et al, 2019; Baeveski et al, 2020). The Wav2Vec model is a self-supervised model, similar to Word2Vec (Mikolov et al, 2013), that takes untranscribed (unsupervised) speech data, masks some segments of speech, and predicts the masked segment using unmasked segments. This self-supervision encodes the speech. The encoder architecture is similar to Baidu's Deep Speech except it uses Transformer (Vaswani et al, 2017) instead of GRU. They then build a decoder using a small dataset of transcribed speeches. They train the model on 53,200 hours of untranscribed English speech and on hundreds of hours (the exact number depends on the experiments) of transcribed speech. The model performance was an order of magnitude better than the state-of-the art models including Deep Speech mentioned above; the model error-rate is 1.8 to 3.3 WER.

The most recent break-through in this space is OpenAI's Whisper model (Radford et al, 2022). The Whisper model architecture is similar to Facebook's Wav2Vec, using a CNN to encode speech spectrograms and applying multiple Transformer layers (Vaswani et al, 2017). The innovation is that they use weak supervision (data with incomplete or noisy labels) instead of unsupervised audio or gold-standard annotations. They train on 680,000 hours of multilingual data via weak supervision, using an end-to-end encoder-decoder architecture. The performance is remarkable, approaching and sometimes beating human performance across a variety of datasets, despite not fine-tuning on any specific dataset.

A particularly relevant finding of Whisper to us is its performance on noisy data. Using an artificial noise injector system called the Audio Degradation Toolbox (Mauch and Ewert, 2013), Radford et al. (2022) show the Whisper model performs better than other existing models in a noisy setting.

But this discussion is limited for two reasons. First, it lacks an objective performance benchmark. Outperforming other models isn't enough if other models don't perform well. The paper shows the Whisper model error rate is as high as 100 WER (almost ten times that of the first Deep Learning model from Baidu) when injecting a high level of noise. Second, the noise tool is artificially generated for music settings and may not mirror the real world noises. Particularly, in an urban environment, noises such as construction, traffic, and background speech sounds pollute dialogues and conversations, and those noises may not be well captured by

noises from the music context. We fill this gap in this paper by zooming on Whisper's performance with respect to speech recorded in real-world urban settings.

3. DATASET AND EVALUATION METRICS

We have constructed the dataset with our own audio samples. For recording the audio samples, we perform readings from four different text-sources: (i) the introduction section of the Whisper paper (Radford et al, 2022), (ii) conversational dialogue from the last episode of TV show "Friends" (Bright, 2004), (iii) an excerpt of Amazon CEO Jeff Bezos's congressional statement (Amazon, 2020), and; (iv) our syllabus for the UT Austin Case Studies in Machine Learning course (Jiao, 2022). All the sources are paired with their transcripts, taken from the Internet.

The recordings for all the above-mentioned text-sources have been done in two different environmental settings - quiet and noisy. This helped us to get a diverse dataset covering a broad distribution of audio from two different environments, recording setups, and speakers. We followed a uniformity in the recording language which is English. Moreover, all these recordings have been done at the sampling rate of 44100 Hz and 16 bit signed PCM in WAV format, using Audacity tool[1].

As mentioned, we recorded the data in two environmental settings - one inside a quiet room at home and the other in a noisy atmosphere. We capture "noisy" speech data in different urban settings, with the purpose of having a good comparison in the performance of the model. One author recorded the noisy data in the ambience of Starbucks cafe in Bangalore, India; second recorded on the New York City subway; third recorded on a busy crowded street in Ranchi, India. So, each of the authors produced 8 recordings (24 total files), corresponding to the four text-sources in two environments. Each of the text-sources varies in length. Jeff Bezos's congressional statement is the longest transcript and contains about 1888 words, whereas Friends' episode's conversational dialogue holds 459 words, the introduction to Whisper paper consists of about 771 words and the syllabus for our subject CSML, being the shortest transcript, carries 228 words. However, reading all of these sources we were able to collect around 2 hours and 14 minutes of recording in total.

[1] Audacity® software is copyright © 1999-2021 Audacity Team.
The name Audacity® is a registered trademark.

Since the Whisper model has a limitation of decoding speech no longer than 30 seconds of duration, we further chunked our 24 recordings, producing 360 audio chunks each of length under 30 seconds. For splitting each recording into chunks, we used Audacity to listen to the data, and then add timestamps to the text-transcripts corresponding to each recording. The timestamps were marked in such a way that we preserved the boundary of the sentences in the text, along with ensuring the duration of each chunk to be less than 30 seconds. We use Audacity to track the waveform of each of our recordings to mark on the pauses before the start of a long sentence, keeping in mind not to exceed the duration of the chunks (sentences between two timestamps) beyond 30 seconds. In total, each of the three authors produced 8 annotated transcripts, one for each audio file.

The average length of chunk for the three speakers is shown in ***Table 1***. We anonymised author names to prevent any implicit bias to our experimental approach.

Speaker	**Urban noise setting**	**Total audio length (in seconds)**	**Number of chunks**	**Average chunk length (in seconds)**
Author A	Cafe	2564	128	20.03
Author B	Busy street	2789	119	23.44
Author C	Subway	2693	113	23.83

Table 1*:* Audio recording statistics of different speakers

We fine-tuned the Whisper model on two of the sources: Jeff Bezos's congressional statement and Friends' last episode dialogue (dataset shorthands: "Bezos" and "Friends"). We used the other two data sources as our test dataset: Whisper paper's Introduction and syllabus of Case studies in Machine Learning (dataset shorthards: "Whisper" and "CSML"). As we are interested in performance in noisy urban environments, we only use the "noisy" portion of our test sets.

The metric that we have used for measuring the performance is Word Error Rate (WER) (Woodard et al, 1982). WER is a common metric for evaluating the performance of an Automatic Speech Recognition task. It is based on the Levenshtein distance, which is a measurement of the differences between the sequences of words in a transcription. A higher value of WER,

represents lower accuracy of the ASR system, and a lower value of WER indicates better accuracy in recognizing speech.

The formula for calculating WER is:

Word Error Rate = (Subs + Ins + Del) / Number of Spoken words

where, **Subs** is the number of substitutions, **Ins** refers to the number of insertions and **Del** is the number of deletions. In Section 5 we discuss the WER of Whisper models before and after fine-tuning on our data.

4. METHODS

Figure 1 summarizes the architecture of the Whisper model. It is an encoder-decoder Transformer model which has been trained on multiple tasks related to speech processing such as speech translation, spotting of spoken language, multilingual speech recognition and voice activity detection. The Transformer decoder block predicts all of the above listed tasks as a sequence of tokens, allowing replacement of multiple stages of a conventional speech processing pipeline with a single model. OpenAI has released nine Whisper checkpoints in two categories: (i) English-only checkpoints: four models are released ("tiny" to "medium" size), and (ii) Multi-lingual: five models are released ("tiny" to "large" size) with coverage over 96 languages.

All varieties of Whisper are trained on weakly supervised datasets, where each audio recording is broken into 30-second chunks. To adapt Whisper to noisy urban environments, we must first preprocess the data to ensure the data is fed to the Whisper model appropriately. Concretely, we first downsample our recorded audio from 44,100 Hz to 16,000 Hz which is the typical sampling rate for human speech. Then, we use the transcripts (Section 3) which have hand-annotated timestamps of up to 30 seconds. We use these timestamps to split the 1-dimensional amplitude vector into chunks, and ensure that each chunk has length less than 480,000 (30 seconds x 16,000 Hz sampling rate). We thus end up with a pair of (i) audio chunk and (ii) transcript chunk, which can be used for training and inference of Whisper. Each such pair becomes a row in our dataset.

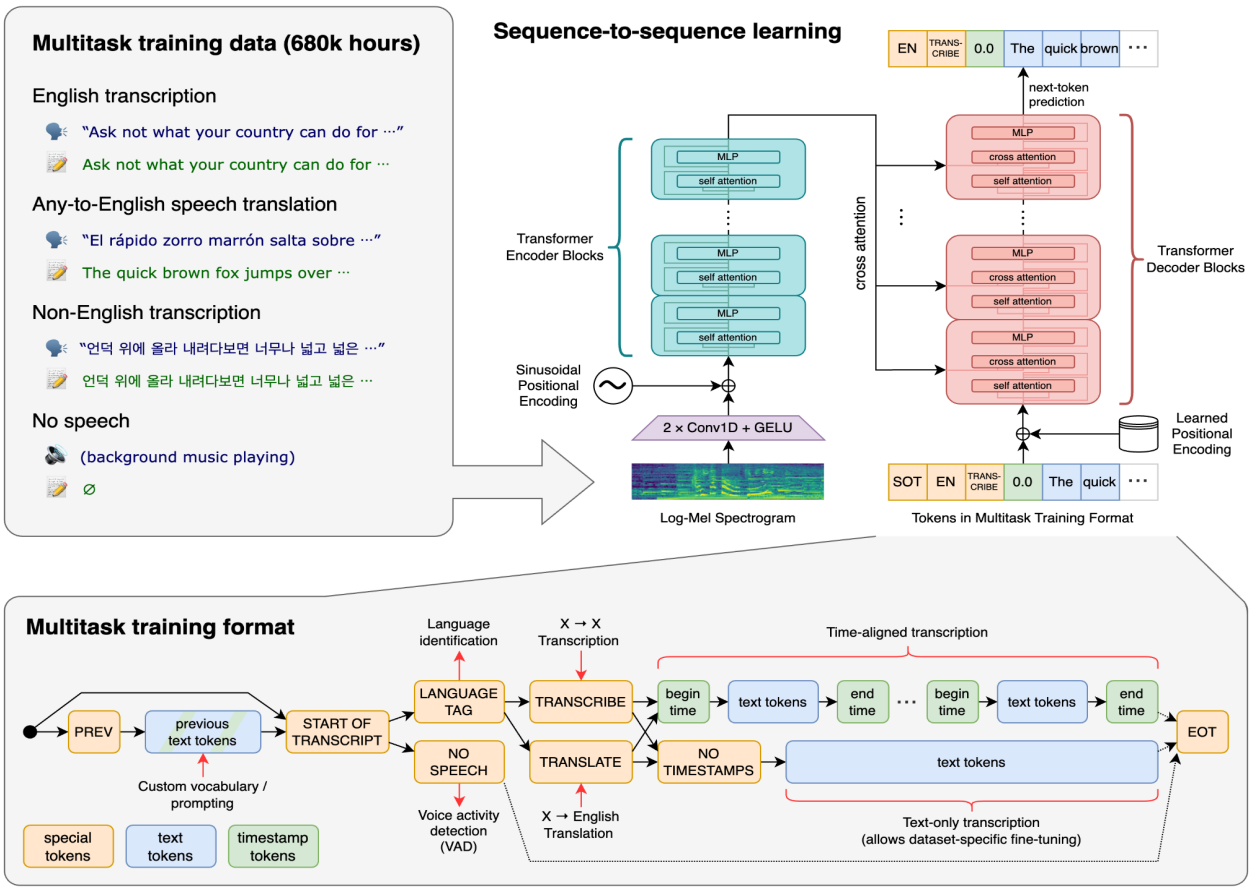

Figure 1. Overview of Whisper model architecture

We fine-tune the "whisper-tiny.en" model, as it is the fastest one and can thus be adapted for a handheld device. We evaluate the following hypothesis: can we improve Whisper's WER on noisy urban data, when we fine-tune on audio from a similar noisy setting? Within this hypothesis we have two sub-criterion: (i) does single-speaker or multi-speaker training data provide better performance? (ii) does a fine-tuning on a mix of different urban environments provide a holistic performance improvement?

To perform fine-tuning, we take a baseline as the off-the-shelf Whisper model which is implemented in HuggingFace Transformers library (Wolf et al, 2020). The library exposes all nine model checkpoints released by OpenAI, which can be fine-tuned using the Seq2SeqTrainer class. Each input data must be passed through the WhisperProcessor class, which (i) tokenizes the ground-truth transcript using WhisperTokenizer (ii) converts the 1-dimensional audio vector (amplitude wave) into a log-Mel spectrogram of dimensions (80, 3000), where 80 is the

feature-size of the spectrogram. HuggingFace's Whisper implementation greatly sped up the experimentation process, but has two limitations (i) it is needed to disable HuggingFace's caching of intermediate results so as to ensure reproducibility across runs (ii) at the moment, long-form text (i.e. greater than 30 seconds) is not supported for fine-tuning or inference, thus we needed to use our chunking procedure for both.

We note that Whisper is an industry-grade model with non-trivial computation needs, and thus our training setup required a fair amount of machine learning engineering. For all experiments, we spun up three AWS p3dn.24xlarge instances (8 NVIDIA V100 GPUs, each equipped with 32 GB RAM), with a shared AWS Elastic File System for data storage. This gives us 24 GPUs across which we can parallelize our experiments. As we have a small number of training rows (only 124 for noisy data across all speakers), the distributed-data parallelism paradigm cannot be used to speed up computation. Instead, we train 24 models, exploring different hyperparameter combinations in parallel. To do so in a principled manner, we use Ray (Moritz et al, 2018). Ray is a distributed computing framework, which makes it easy to write distributed programs. We deploy a Ray cluster using one p3dn.24xlarge instance as the "head" node and the other two instances as "worker" nodes. By configuring a Ray Task using @ray.remote, we can ensure that each copy of the model utilizes a single GPU and 10 CPUs for fine-tuning. Each node has its own in-memory Redis cache, which stores global object references which are synchronized across nodes; for complete details see (Moritz et al, 2018). We push our dataset (including the tensors) as a remote object reference, so that they can be downloaded on each worker using ray.get(...) command. This minimizes the disk-reading overhead and ensures each training run has a consistent view of the dataset. During each run, we apply filters based on speaker and text-source, so as to create the relevant train and test splits.

With this setup, we can write a model-orchestration code such that we can submit many fine-tuning jobs (having different hyperparameters) to the cluster, and they are executed in parallel. Ray thus becomes a convenient interface to tune hyperparameters. In ***Table 2***, we describe the various hyperparameter-ranges which we explored when tuning the whisper-tiny.en model on our noisy dataset. Note that we save the best model across epochs.

In total, we fine-tuned 178 whisper-tiny.en models, where each model's forward pass requires 15.339 milliseconds for each second of audio to be transcribed.

Hyperparameter	Ranges explored
Train batch size	2 to 16
Training epochs	10 (best model across epochs is saved)
Learning Rate	1e-6 to 3e-5
Optimizer	AdamW
fp16	True
Gradient accumulation steps	1
Warmup steps	250 to 1000
Gradient checkpointing	True

Table 2. Hyperparameter ranges explored during fine-tuning of whisper-tiny.en. These are forwarded to the HuggingFace Seq2SeqTrainer

Modern deep learning models include several sophisticated techniques to ensure that they are trained appropriately and quickly. We use the following techniques: (i) mixed-precision training: here, we use a combination of 16-bit and 32-bit floating-point operations. Even though the underlying NVIDIA V100 hardware is 32-bit, representing model weights, activations and gradients as 16-bit floating-point tensors can speed up computation by up to 3x. This is controlled by the "fp16" parameter; (ii) gradient accumulation: this performs several forward and backward passeses and sums up the gradients, before finally updating the model. This is most useful when we have limited GPU memory and we want to simulate a larger batch size: e.g. if we set a batch size of 8 and gradient accumulation of 6, then we do 6 forward-backward passes and accumulate gradients of 8 x 6 = 48 examples, before updating the model as one "step" of gradient descent. In our situation we have sufficiently high GPU memory (32GB) so we can set this to 1, which means we update model weights after each forward-backward pass (standard procedure); (iii) warmup steps: in many pre-trained deep learning models (including Whisper), setting a very high learning rate during fine-tuning can make the model forget pre-trained representations. Additionally, when using adaptive optimizers such as Adam, AdamW, or RMSProp, a few steps of gradient descent are needed to calculate accurate statistics of the descent procedure. Thus, we use warmup, wherein for a few steps, we linearly increase the

learning rate up to the final value, thus gathering accurate statistics for the rest of the training steps; (iv) gradient checkpointing: modern deep learning models have several million to billions of parameters. When we perform a forward-pass, we compute the activations for each layer in the so-called "computation graph". The activations of any particular layer is a tensor of dimension B x layer_output_shape, where B is the batch size during training. If we increase the batch size, we end up storing larger and larger tensors in memory during each forward-backward pass. With gradient checkpointing (Chen et al, 2016), we can train an n-layer network with O(√n) memory cost, by discarding activations during the forward pass and recomputing them as needed in the backward pass. Thus, we can try out larger batch sizes and deeper models without worrying about exceeding GPU memory.

5. RESULTS

Figure 2 shows the performance of various Whisper models on speeches recorded in quiet and noisy environments. Each panel corresponds to a different Whisper model, where the model size increases as we move from left to right and from top to bottom. The models are all English-only transcription models: whisper-tiny.en (39M parameters), whisper-base.en (74M parameters), whisper-small.en (244M parameters), whisper-medium.en (769M parameters). We do not evaluate whisper-large (1.5B parameters) as there is no English-only version available. The y-axis is the Word Error Rate (i.e. high number means less performant), while the x-axis denotes different speeches. For each speaker, we show the model's error rate on the person's noisy (blue) and quiet (orange) speeches.

We note three findings. First, we see that across all models and speakers, the error rate on speeches recorded in noisy environments (blue) is higher than that recorded in quiet environments (orange). Second, the error rate decreases as we move from small to large models, meaning large models perform better on all of our data. Given larger models' expensive computation, there is a trade-off between accuracy and inference time. Third, although larger models improve on both noisy and quiet speeches, there always exists a performance gap between the two.

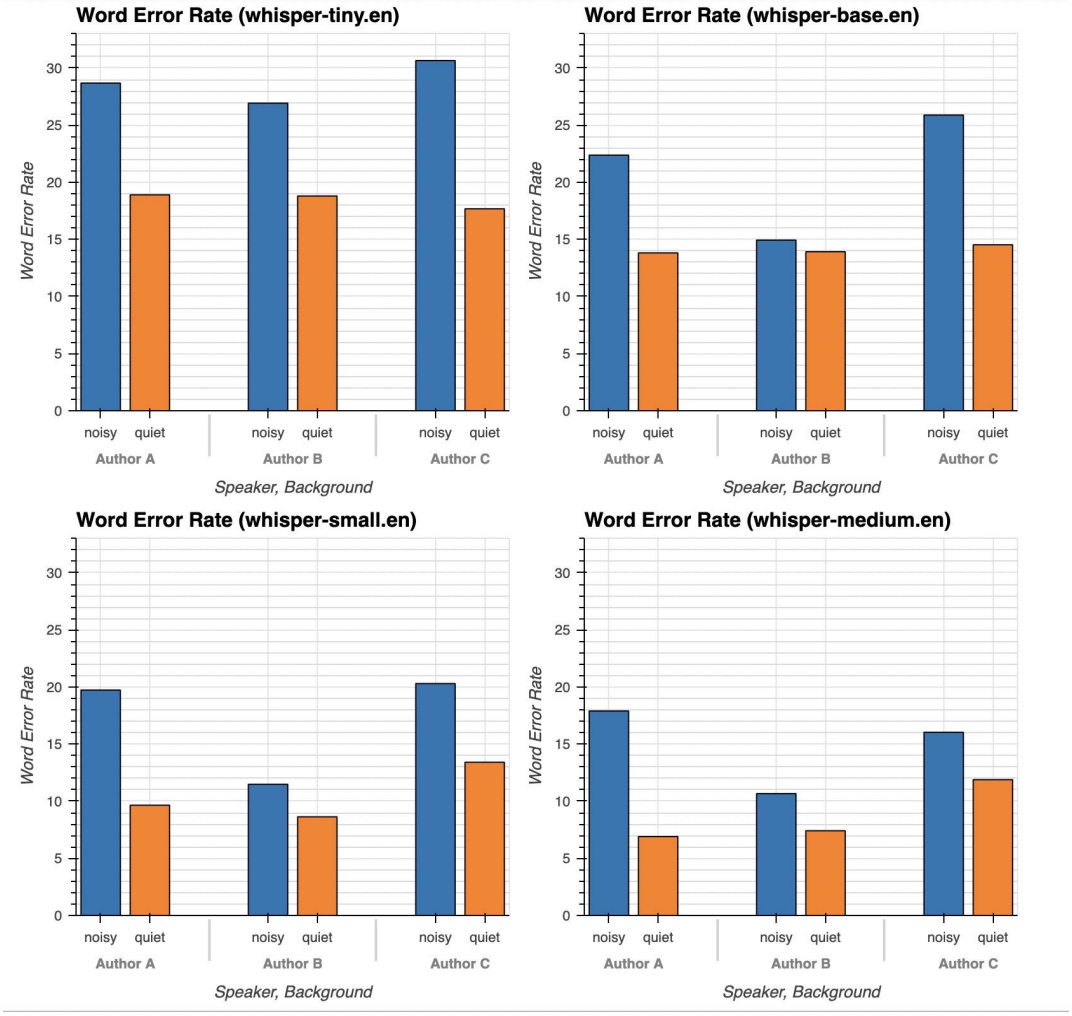

Figure 2. Out-of-box performance of various Whisper models. We evaluate the two test datasets (i.e. CSML+Whisper) as a single combined test-set. We measure WER in both noisy and quiet settings, to observe the performance degradation due to urban noise.

The third finding needs further elaboration. We see that small Whisper models ("tiny" and "base") perform particularly worse on noisy speeches, in all urban settings. This is concerning because handheld devices (e.g. smartphones) are most likely to be exposed to noisy environments (e.g. on a train to work) and these devices have limited computing resources and thus need to use smaller models. Thus, we are motivated to enhance the small models so that the performance gap between the noisy and the quiet speeches is not as dramatic, while still maintaining low prediction times. In ***Figure 3***, we showcase the differences in prediction time between Whisper variants: the "medium" model (769M params) is 3.65x slower than the "tiny"

model (39M). We note that these numbers were generated on a NVIDIA V100 GPU rather than a mobile device, so the exact prediction time might differ on a mobile device, but we expect the relative difference in inference times to be similar.

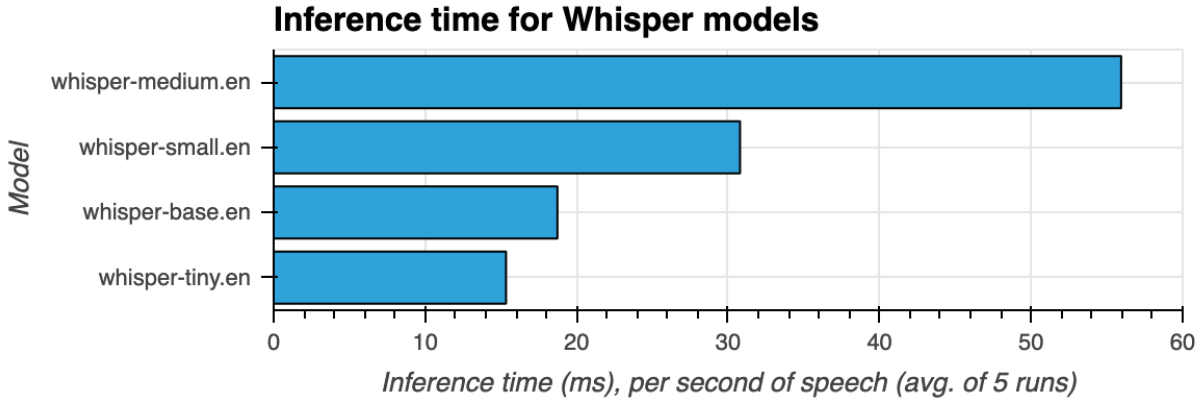

Figure 3. Inference time for various Whisper model sizes. The x-axis measures inference time to transcribe one second of monophonic speech with a sampling rate of 16 kHz.

Now we turn to our attempts to improve Whisper's performance on noisy speeches. We fine-tune "whisper-tiny.en" on the training data of our noisy recordings using the "Bezos" and "Friends" text-sources. We tried various hyperparameters as shown in ***Table 2***. and chose the most performant configuration. ***Table 3*** showcases our results in various settings.

Experiment	**Train dataset**	**Train WER**	**Test dataset**	**Test WER**	**Best Hyperparameters**
Baseline	N/A	N/A	Author A Noisy (CSML+Whisper)	28.69	N/A
Baseline	N/A	N/A	Author B Noisy (CSML+Whisper)	26.93	N/A
Baseline	N/A	N/A	Author C Noisy (CSML+Whisper)	30.66	N/A
Single-speaker finetuning	Author A Noisy (Bezos+Friends)	24.61	Author A Noisy (CSML+Whisper)	24.52	batch_size=8 fp16=True gradient_accumulation=1

					gradient_checkpointing=True train_epochs=10 learning_rate=1e-05 warmup_steps=100
Single-speaker finetuning	Author B Noisy (Bezos+Friends)	22.64	Author B Noisy (CSML+Whisper)	24.39	batch_size=2 fp16=True gradient_accumulation=1 gradient_checkpointing=True train_epochs=10 learning_rate=3e-05 warmup_steps=500
Single-speaker finetuning	Author C Noisy (Bezos+Friends)	23.18	Author C Noisy (CSML+Whisper)	28.93	batch_size=2 fp16=True gradient_accumulation=1 gradient_checkpointing=True train_epochs=10 learning_rate=3e-06 warmup_steps=250
Multi-speaker finetuning	Author A+B+C Noisy (Bezos+Friends)	23.9	Author A Noisy (CSML+Whisper)	25.43	batch_size=2 fp16=True gradient_accumulation=1 gradient_checkpointing=True train_epochs=10 learning_rate=1e-05 warmup_steps=500
Multi-speaker finetuning	Author A+B+C Noisy (Bezos+Friends)	27.87	Author B Noisy (CSML+Whisper)	23.27	batch_size=2 fp16=True gradient_accumulation=1 gradient_checkpointing=True train_epochs=10 learning_rate=3e-06 warmup_steps=500
Multi-speaker finetuning	Author A+B+C Noisy (Bezos+Friends)	34.6	Author C Noisy (CSML+Whisper)	29.75	batch_size=2 fp16=True gradient_accumulation=1 gradient_checkpointing=True train_epochs=10 learning_rate=1e-06 warmup_steps=250

Table 3. Hyperparameters and Word Error Rate (WER) for various experimental settings

We see that during single-speaker finetuning, model performance improves by an average of 2.81 WER (28.76 to 25.95) and with multi-speaker finetuning, model performance improves by an average of 2.61 WER (28.76 to 26.15). This indicates that we should attempt to fine-tune a customized model for each speaker and noise setting, rather than build a single model which works better across noise settings. As future work, we can explore training a higher-level audio classifier to detect which urban noise setting the user is in (expert model), and select a more performant model accordingly.

6. DISCUSSION AND CONCLUSION

In this paper, we study the performance of Whisper in a noisy urban environment. We show that Whisper performs worse on speeches recorded in noisy environments than on those recorded in quiet environments. In particular, small Whisper models perform particularly worse on noisy speeches. This is concerning because handheld devices (e.g. smartphones) are likely to be exposed to noisy environments (e.g. on a train to work) and these devices probably need to use smaller models (e.g. whisper-tiny) due to limited computational resources. Thus, we attempt to enhance the small model to bridge the performance gap between the noisy and the quiet speeches.

We show that training on a small dataset of speeches recorded in noisy environments improves the model performance. We see that during single-speaker fine-tuning, model performance improves by an average of 2.81 WER (28.76 to 25.95) and with multi-speaker finetuning, model performance improves by an average of 2.61 WER (28.76 to 26.15). Thus, single-speaker fine-tuning is preferred.

There are several next steps. First, we plan to share the results with HuggingFace's Whisper Fine-Tuning Sprint running from December 2, 2022 through December 22, 2022 (Hugging Face, 2022). Here, we will share our results with the community of machine learning scientists and engineers who work at the fore-front of ASR technology. Second, we plan to collect more noisy Speech-to-text data and further fine-tune Whisper. This requires scaling to beyond a few people reading out scripts as performed in the current work. One avenue is to gather volunteers (e.g., HuggingFace community, YouTubers whose videos are set outdoors) who are willing to share their own recordings. Third, we may build an auxiliary noise-canceling deep

learning model to "clean out the noise" as a preprocessing procedure. Although we could not identify an organization pursuing this project as of now, we can imagine commercial opportunities, suggesting companies pursuing this project in the future.

REFERENCES

1. Amazon. (2020). Jeff Bezos Congressional statement. *Statement by Jeff Bezos to the U.S. House Committee on the Judiciary*. https://www.aboutamazon.com/news/policy-news-views/statement-by-jeff-bezos-to-the-u-s-house-committee-on-the-judiciary
2. Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... & Zhu, Z. (2016). Deep speech 2: End-to-end speech recognition in English and Mandarin. *In the International Conference on Machine Learning* (pp. 173-182). PMLR.
3. Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33, 12449-12460.
4. Bright, K. (Director). (2004). Friends Last Episode (Season 10, Episode 1018) [TV series episode]. In *Friends*. https://www.livesinabox.com/friends/scripts.shtml
5. Cartwright, M., Cramer, J., Mendez, A. E. M., Wang, Y., Wu, H. H., Lostanlen, V., ... & Bello, J. P. (2020). SONYC-UST-V2: An urban sound tagging dataset with spatiotemporal context. *arXiv preprint arXiv:2009.05188*.
6. Chen, T., Xu, B., Zhang, C., & Guestrin, C. (2016). Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.
7. Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
8. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., ... & Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.
9. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

10. Hugging Face. (2022). Whisper Fine-Tuning Event. *GitHub repository.* https://github.com/huggingface/community-events/tree/main/whisper-fine-tuning-event
11. Jiao, D. J. (2022). Case Studies in Machine Learning Syllabus. *Machine Learning and its Applications.* https://courses.edx.org/asset-v1:UTAustinX+CSMS.ML.312+2T2022+type@asset+block@ML_and_Applications_Revised_Syllabus.pdf
12. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4), 541-551.
13. Mauch, M., & Ewert, S. (2013). The audio degradation toolbox and its application to robustness evaluation.
14. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
15. Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., ... & Stoica, I. (2018). Ray: A distributed framework for emerging AI applications. In the *13th USENIX Symposium on Operating Systems Design and Implementation*.
16. Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust speech recognition via large-scale weak supervision. *OpenAI Blog*.
17. Schneider, S., Baevski, A., Collobert, R., & Auli, M. (2019). Wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*.
18. United States Census Bureau, (2022). Deaf History Month. United States Census Bureau. https://www.census.gov/library/audio/profile-america/profileodd/profile-odd-13.html
19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
20. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*.

21. Woodard, J. P., & Nelson, J. T. (1982). An information theoretic measure of speech recognition performance. In *Workshop on standardization for speech I/O technology, Naval Air Development Center, Warminster, PA*.