BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

http://fit.vut.cz

Project practice 2024/2025

Extending Whisper for Target Speaker ASR

Daniel Bohata*

Abstract
The problem of transcribing the speech of a single person in multi-speaker environments is a complex one with some existing solutions. In this work I have sought to replicate and attempt to improve one such approach. The chosen approach is extending a well known single speaker transcription model (Whisper) for this task using prompt tuning [3]. That means working with a frozen model by injecting soft prompts into the encoders, possibly decoders as well. My results have been achieved by replicating a paper by the authors of this method [3]. Furthermore, I analyzed the results when applying more training epochs, using different schedulers and mixing both clean and noisy datasets together. Training and evaluation has been done using GPU clusters provided by MetaCentrum. I have obtained results with an average difference of 1.12% from the original. Further experiments have not shown significant improvement over previous results. My results are within minimal margin from results claimed by the authors of this approach and my experiments did not succeed in improving them.

Keywords: Target Speaker — Automatic Speech Recognition — OpenAI Whisper

Supplementary Material: Original code — Improved code

*xbohatd00@fit.vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Target speaker automatic speech recognition (TS ASR) is a well-known problem with many use cases. First, it has been solved by training a model from scratch. Then, fine-tuning single speaker models came about, but that takes a lot of time and resources. In this work, I aim to reproduce results of another method. The method of choice is extending an already trained single speaker model, OpenAI's Whisper, for this task using prompt tuning. This should save both time and resources required to achieve transcription. The method of choice has already been studied and published with results; hence, I am only aiming to reproduce the results claimed and attempt to improve them.

The aim of TS ASR is to accurately transcribe speech from a specific target speaker in an audio that may contain overlapping speakers or background noise. The core problem is that existing state-of-the-art methods either train a model from scratch or fine-tune an already existing one, both of which are time-consuming and resource-consuming [2]. The correct solution should ideally decrease both the time and the resources needed to successfully complete the transcription as quickly as possible while minimizing the word error rate (WER).

A solution to this already exists and is the core of this work. It is based on expanding Whisper's capabilities using prompt tuning instead of fine tuning. The code and the paper have been published by Hao Ma, Zhiyuan Peng, et al. [2]. Their code is used as the basis for my implementation [3].

My solution is to expand on the existing implementation by achieving better results eg. lower word error rate. The aim is to do that by using different schedulers, longer training, or training on a combined dataset. Unfortunately, the code and instructions that came with it had some problems that had to be solved. First, the parameter *experiment name* was actually the name of checkpoints that would be created in training. Second, there was a type conversion error in the

user	fairshare	job count				
	m1	total	queued	running	completed	other
tasevsky	189	10001	0	0	0	10001
galaxyelixir	188	102	0	8	94	0
galaxyumsa	187	46	0	0	46	0
mark_cheeky	186	3	0	1	2	0
xveselsky	185	32	0	0	32	0
lunak	184	1	0	0	0	1
janavisovanova	183	1	0	0	1	0
jendrb00	182	4	4	0	0	0

Figure 1. Efficiency when running jobs is key, as we can see that someone with 10000 jobs has the highest fairshare

code that needed to be fixed or else training was not possible.

2. Methodology

In this section, experimental setup, Whisper and prompt tuning is introduced. Furthermore, the process of obtaining the training data set and speaker embeddings is also introduced.

2.1 Experimental setup

To replicate the results, a powerful GPU was needed as Hao Ma's group used a 24GB Nvidia 3090, which I do not have. For that, the MetaCentrum cluster network was chosen. A logging tool was also missing in the code, for that weights and biases was chosen and implemented. In this section, MetaCentrum, it's scheduling system PBS and metrics logging will be described.

2.1.1 MetaCentrum

MetaCentrum is a free platform for students and employees of Czech universities that allows students to try out high performance computing. There are about 480 available GPUs, including models A10, A40, and A100, and around 53000 CPU cores [1]. Each user is allowed to store about 10TB of data across different clusters.

2.1.2 Scheduling system

To compute a job, it must be submitted to a queue using the *qsub* command. MetaCentrum's scheduler then gives you access to resources based on their special metric **fairshare** and the resources you requested. Fairshare is a metric that evaluates how well you used the resources available to you. Having low fairshare means that you have a lower priority compared to other researchers, which results in longer queue times [4]. Fairshare ranking example can be seen in Figure 1. When training, I always requested 2 CPUs, 1 GPU and 24/32/48GB of RAM and VRAM based on the size of the Whisper model in use - small/medium/largev2.

2.1.3 Logging

Since the original script lacked any form of logging except plain text to standard and error output, logging to weights and biases was integrated. That is where all the graphs in this paper come from. The chosen model and other parameters are passed either as arguments or as environment variables.

2.2 Dataset

In order to replicate the results, the same data set that the original paper used and the same speaker embeddings had to be obtained. That being the Libri2Mix data set and its x-vector embeddings [2] .

2.2.1 Speech mixtures

I obtained the speech mixtures using LibriMix GitHub repository available here. When generating, I made sure to generate only 16kHz audio and only Libri2Mix.

2.2.2 Metadata and embeddings

Metadata used for training has been generated using a script provided here and embeddings were downloaded from here. The created metadata are called *train-100.json*, *train-100-noisy.json*, *test.json* and *test-noisy.json*.

2.3 Whisper

Whisper is a state-of-the-art, open source, multilingual speech transcription model developed for single-speaker ASR instances. It can also produce speech timestamps, perform inverse text normalization, etc. It's input is an 80-mension log-Mel spectrogram and its output is transcribed text [2].

2.4 Prompt tuning

Prompt tuning is a method of extending a model's capabilities without changing the original weights (which would be fine-tuning). It is a lightweight technique that prepends a small number of trainable tokens to the input called soft prompts. In our case, these tokens are trained on target speaker embeddings [2].

2.4.1 Deep prompting

Deep prompting is an extension of prompt tuning, as now prompts are inserted not only at the input but also in multiple layers within the model. This results in lower control over the model, but a possibility of better results [2].

2.4.2 Soft prompt reparametrization

It has been reported that directly optimizing soft prompts may make the training process unstable, which is why a two-layer multilayer perceptron is implemented to make it more stable [2].

3. Validation

To validate the claimed results, I had to train the model myself and compare the results. The code is public, so all I needed to do was to run it. The first training run unfortunately crashed due to a type error where the script was expecting a tuple in place of a list, so that had to be fixed.

3.1 Training

In training, the script first runs a hard coded ten epochs of training on the clean data and only one on noisy data. The learning rate is scheduled using a step scheduler that changes the learning rate from 10^{-4} to 10^{-5} after five epochs. The suggested prompt length is sixteen, and my experiments confirm that [3]. In particular, the proposed script functions the best with a batch size of one, which is quite unusual. The script also offers the ability to choose whether to use deep prompting, reparametrization, both, or neither. Training is performed on metadata *train-100.json* and *train-100-noisy.json* created in Section 2.

3.2 Evaluation

Evaluation is performed on metadata that come from the same dataset as the metadata used for training. The metadata is called *test.json* and *test-noisy.json* and was created in Section 2. The evaluation is performed at the last checkpoint created in the training, and its output is the word error rate.

Table 1. Performance under different settings claimed by Hao Ma and his team [2]. DP stands for deep prompting and MLP stands for multi layer perceptron. L/M/S stands for Whisper LargeV2/Medium/Small respectively

		test-clean			test-both		
DP	MLP	L	M	S	L	M	S
×	×	17.59	17.73	31.03	37.26	39.38	56.30
×	✓	15.92	14.61	24.27	32.25	32.34	45.72
✓	×	14.82	13.89	24.62	**30.19**	**29.81**	45.46
✓	✓	**14.78**	**13.54**	**23.08**	30.71	30.72	**44.16**

4. Experiments

In my attempt to further improve this approach, I chose to use only the medium model, as it proved to be the best one in Section 3. The second choice I made was to only experiment while using both deep prompting and reparametrization as it, in total, proved the best in seven out of twelve cases in Tables 1 and 2. Then I and my supervisor came up with some ideas that I would like to try:

Table 2. Performance under different settings achieved by me

		test-clean			test-both		
DP	MLP	L	M	S	L	M	S
×	×	17.43	16.06	30.10	37.82	36.57	51.54
×	✓	16.53	14.72	23.87	33.47	31.51	44.74
✓	×	**16.12**	14.64	23.86	**31.66**	**30.75**	44.47
✓	✓	16.80	**14.62**	**23.07**	32.32	31.04	**43.57**

1. Training more epochs with a larger batch size
2. Using different schedulers for training
3. Mixing clean and noisy data into one dataset

4.1 More epochs and a larger batch size

In this attempt, I increased the total number of training epochs on clean data from ten to twenty and kept the total number of epochs on noisy data. Then I increased the batch size from one to sixteen. This has proven to be a bad approach as the word error rate on clean data increased to **17.63** and to **34.28** on noisy data which is way higher than WER obtained in Section 3.

4.2 Using different schedulers

In this approach, I set the batch size at one and the clean training epochs at ten. Then I increased the training epochs on noisy data to five. After that, I chose five different commonly used schedulers to determine the best one. The hypothesis that using different learning rate schedulers can affect the final performance of a speech model by better adapting the optimization process. The results can be seen in Table 3.

Scheduler	Clean	Both
Step	15.12	31.62
Linear	**14.72**	30.91
Cosine	15.57	**29.96**
Exponential	15.34	30.06
One cycle	15.89	32.48

Table 3. Performance of different schedulers on clean and noisy data

The results of individual schedulers on clean data turned out to be about the same for all of them and just a little worse than the results in Table 2. This could probably be attributed to longer training on noisy data.

However, on noisy data, we can see that the Cosine annealing scheduler managed to get below 30% WER which is about a 1% improvement on Table 2. Most importantly, it beats the step scheduler used in the original code by almost 2% on the same task, suggesting that it could be better for instances where a noisy environment is expected. Also notable is that the linear scheduler beats the step one on both clean and noisy utterances, suggesting that it could be better for general use.

4.3 Mixing clean and noisy datasets

In this approach, I chose to mix the training data into one large file. Now instead of training on clean data and on noisy after that, the model trains on both at the same time. To make sure I get the most information from this approach, I also used five different schedulers. The hypothesis was that by having the model exposed to a broader range of acoustic conditions, the results would be more consistent. The results can be seen in Table 4.

Scheduler	Clean	Both
Step	19.49	33.73
Linear	18.93	33.74
Cosine	19.80	34.29
Exponential	**18.20**	**32.65**
One cycle	20.76	34.32

Table 4. Performance of different schedulers after training on mixed data

The results of this experiment are much worse than those in Table 2. The best result was achieved by the exponential scheduler in both clean and noisy environments, but it was still about 3% worse than the previous results. It seems that this experiment has introduced some conflicting acoustic patterns that hindered the model's learning capabilities. It is clear that training on a mixed set of data is not a good approach.

4.4 Results

In the end, I have managed to get results within close range of those claimed by Hao Ma, et al. in their paper [2]. To be exact, the average difference between mine and their results was 1.12%. When taking into consideration the training time required to get the results, I can say that their solution is a strong one when it comes to environments without background noise, with WER around 14-15%, but in noisy environments, not so much as the WER is way higher at around 30-31%. The obtained results are shown in Table 2 and

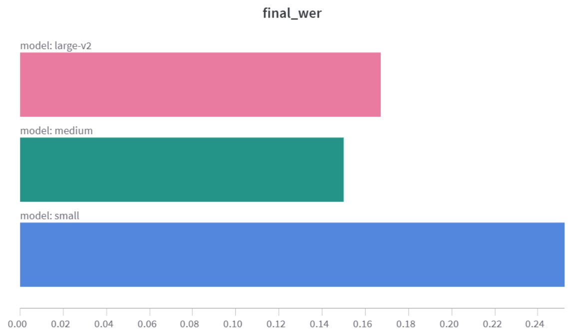

Figure 2. Graph showing performance by model on clean data

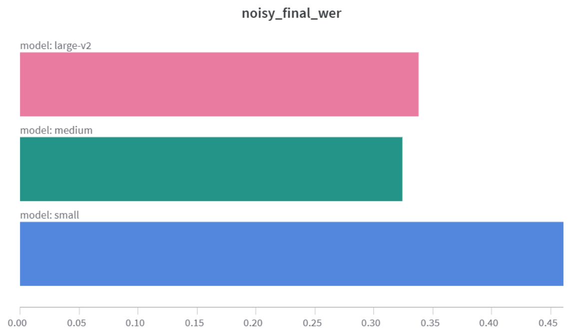

Figure 3. Graph showing performance by model on noisy data

those of the original article are shown in Table 1. What can be seen in the results is that using both yields the best results only about half of the time, the other half being deep prompting by itself. Further, what can be seen from the results is that the medium model achieves the best results in both clean and noiys environments. That can be best seen in Figures 2 and 3 where the medium model beats the small one by a large margin and the large one by a little bit.

5. Conclusions

This work has replicated the results claimed by Hao Ma, Zhiyuan Peng, Mingjie Shao, Jing Li, and Ju Liu. Prompt tuning is a fair approach when dealing with target speaker automatic speech recognition. This approach avoids having to do expensive fine-tuning or building a model from scratch and achieves a much lower amount of resources and time needed than previous state-of-the-art techniques [2].

The best result on clean data is a word error rate of 14.62% and the best result on noisy data is a word error rate of 29.96% with a step scheduler as shown in Tables 2 and 3. Both have been obtained using Whisper medium and with both deep prompting and reparametrization enabled. Each of them used different schedulers, step and cosine, respectively, and different numbers of training epochs on noisy data. The most consistent scheduler is the linear one, beating the step one in the same number of training epochs in both clean and noisy utterances, as well as beating the cosine one on clean data. However, all of this still falls well short of the 6.9% WER on clean data and 15.9% WER on noisy data reported by Alexander Polok's team in the paper on Diarization-Conditioned Whisper (DiCoW) [5].

Further work should be focused on implementing support for multi-GPU training as that will cut training time even more. The difference between using speaker embeddings and diarization on noisy data suggests that taking the path of diarization, such as DiCoW, is the better choice for TS ASR.

Acknowledgements

I would like to thank my supervisor Alexander Polok for leading me and helping me with this work.
This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140), provided within the program Projects of Large Research, Development and Innovations Infrastructures. Computational resources were supplied by the project "e-Infrastructure CZ" (e-INFRA CZ) and MetaCentrum.

References

[1] CESNET. *Grid Service MetaCentrum*. 2024. Accessed: 2025-06-10. Available at: https://www.cesnet.cz/gimg/default/8/0/7/807-Grid%20service%20MetaCentrum.pdf.

[2] Ma, H., Peng, Z., Shao, M., Li, J. and Liu, J. *Extending Whisper with Prompt Tuning to Target-Speaker ASR*. 2023. Accessed: 2025-06-08. Available at: https://arxiv.org/abs/2312.08079.

[3] Ma, H., Peng, Z., Shao, M., Li, J. and Liu, J. Extending Whisper with Prompt Tuning to Target-Speaker ASR. In: *IEEE ICASSP*. 2024.

[4] MetaCentrum. *Welcome to MetaCentrum Documentation*. 2025. Accessed: 2025-06-08. Available at: https://docs.metacentrum.cz/en/docs/welcome.

[5] Polok, A., Klement, D., Kocour, M., Han, J., Landini, F. et al. *DiCoW: Diarization-Conditioned Whisper for Target Speaker Automatic Speech Recognition*. 2024. Available at: https://arxiv.org/abs/2501.00114.