Parameter-Efficient Fine-Tuning Methods in mWhisper-Flamingo

Videet Mehta
Computer Science and Artificial Intelligence Laboratory (CSAIL)
Spoken Language Systems Group
Massachusetts Institute of Technology

May 2025

1 Introduction

Audio-Visual Speech Recognition combines visual features from the lips and the audio which has been shown to improve performance in noise. Recently, mWhisper-Flamingo was proposed for multilingual Audio-Visual Speech Recognition (AVSR) which combines the strengths of a pre-trained audio model (Whisper) and video model (AV-HuBERT)[1,7,8]. Through previous research, mWhisper-Flamingo achieves state-of-the-art word error rate (WER) on MuAViC, an AVSR dataset of 9 languages. However, this model consists of nearly 3 billion parameters, and fine-tuning such models requires excessive compute, memory, and resources that may not be available to all those that would like to adapt mWhisper-Flamingo to their specific domain. To make such models accessible to everyone, we propose to use simple adapters to reduce the number of trainable parameters by as much as possible without reducing the performance. In this framework, new adapter layers are inserted into the models and are trained while the pre-trained weights of the original architecture remain frozen. With this, the number of parameters decreases from billions to millions. We try three different implementations of parameter-efficient fine-tuning in automatic-speech recognition (ASR) that have proven in previous research to not result in a loss of performance [2]. The baseline is zero-shot WER of mWhisper-Flamingo, both in the audio and the audio-visual modalities.

2 Linear Adapters

Linear adapters are small, trainable modules inserted into the frozen pre-trained LLM. Instead of updating all the parameters, these adapters will allow for task-specific tuning of just a small number of added weights. We add these adapters after the multi-headed attention and feedforward layers, operating on these hidden states at each layer of the model [9]. Skip connections are also applied and then passed through the following layer normalization. Let $\mathbf{x} \in R^{B\times T\times d}$ be the input hidden states, where B is the batch size, T is the sequence length, and d is the hidden dimension. Define the adapter transformation as:

$$\mathbf{x}_{\text{out}} = \mathbf{x} + \left(\text{GELU}\left(\text{LayerNorm}(\mathbf{x}) \cdot W_{\text{down}}\right) \cdot W_{\text{up}}\right)$$

Here:

- $W_{\text{down}} \in R^{d\times r}$ is the down-projection matrix
- $W_{\text{up}} \in R^{r\times d}$ is the up-projection matrix
- $r \ll d$ is the bottleneck dimension
- $\text{GELU}(\cdot)$ is the Gaussian Error Linear Unit activation [10]
- $\text{LayerNorm}(\cdot)$ denotes layer normalization [11]

We try with different values for the rank $r \in 64, 128, 256, 512, 1024$. Additionally, we run experiments with and without layer initialization. Near-identity layer initialization has been shown to increase the robustness of adapters during fine-tuning, accelerate convergence, and retain the pretrained model's behaviors. The weights are randomly chosen with a normal distribution within $[-0.02, 0.02]$ with a mean of 1 and SD to be 0.001 [9].

Layer Norm tuning is a method that fine-tunes only the gain and bias parameters of LayerNorm layers; previous research demonstrates that this approach of updating just 0.03 % of the model's parameters achieves performance comparable to or exceeding that of other PEFT methods [3,11]. They also find that combining LN-tuning with prefix-tuning and adapter-based methods leads to state-of-the-art results. With this in mind, we run three different experiments with traditional adapters across all ranks r as mentioned above.

1. Original adapter implementation
2. Original adapters + layer initialization
3. Original adapters + layer intialization + layer norm fine tuning

Table 1: Clean and 0-SNR WER for `whisper_en_small` baselines and adapter variants. Lower is better.

Model	**Clean WER (%)**	**Noisy WER (%) (0 SNR)**
Baselines		
W-S Zero-shot	2.23	37.84
W-S FT (base)	**1.19**	**17.65**
Adapters (no init, no LN-FT)		
A64 (1.2M)	1.71	25.80
A128 (2.4M)	1.64	25.70
A256 (4.9M)	1.70	25.34
A512 (9.5M)	1.51	23.98
A1024 (19.4M)	**1.44**	**22.94**
Adapters + Init. (no LN-FT)		
A64+I (1.2M)	1.87	27.73
A128+I (2.4M)	1.85	28.30
A256+I (4.9M)	1.89	28.66
A512+I (9.5M)	1.88	28.29
A1024+I (19.4M)	**1.64**	**26.79**
Adapters + Init. + LN-FT		
A64+I+LN (1.7M)	1.92	27.12
A128+I+LN (2.9M)	1.89	28.71
A256+I+LN (5.4M)	1.89	28.66
A512+I+LN (10.0M)	1.88	28.29
A1024+I+LN (19.9M)	**1.61**	**25.79**

3 LoRA

Low-Rank Adaptation (LoRA) is a parameter-efficient fine-tuning technique that approximates updates to weight matrices via a pair of smaller matrices [12]. Instead of updating the full weight matrix $W_0 \in R^{d \times k}$, LoRA freezes it and adds a low-rank residual ΔW:

$$W = W_0 + \Delta W = W_0 + \frac{\alpha}{r} AB$$

where:

- $A \in R^{d \times r}$ is the down-projection
- $B \in R^{r \times k}$ is the up-projection
- $r \ll \min(d, k)$ is the rank of the approximation
- $\alpha \in R$ is a scaling factor

In our architecture, LoRA modules are inserted into the multi-head attention block of the Whisper encoder and decoder by modifying the following projection layers, as done in [4]:

- Query: $W_q + \frac{\alpha}{r} A_q B_q$
- Key: $W_k + \frac{\alpha}{r} A_k B_k$
- Value: $W_v + \frac{\alpha}{r} A_v B_v$
- Output: $W_o + \frac{\alpha}{r} A_o B_o$

Each LoRA module is a parallel branch to the frozen layer and injects a rank-constrained update that allows the model to adapt while keeping most of its parameters unchanged. We experimented with ranks $r \in 4, 16, 32$. We maintain the same parameters for layer initialization and keep turn off the tuning for layer norms. Recent work by Yao et al. (2024) shows that fine-tuning only the query (W_q) and value (W_v) projection matrices, while keeping the key (W_k) matrix frozen, is often optimal for both performance and efficiency [5]. Their results across benchmarks demonstrate that QV-only fine-tuning matches or exceeds full QKV tuning, while reducing parameter count by a third. This effect arises because during backpropagation, the gradient flow through the value matrix W_v is directly modulated by the softmax attention weights—ensuring strong signal even at initialization—whereas gradients for W_q and W_k depend on each other and tend to vanish when both are near-zero, making W_v far more effective for early and stable learning.

Table 2: Clean and 0-SNR WER for `whisper_en_small` with Low-Rank Adaptation.

Model	**Clean WER (%)**	**Noisy WER (%) (0 SNR)**
Low-Rank Adapters & Layer-Norm FT (QKV)		
r =4 (1.1M)	1.31	21.45
r =16 (3.8M)	1.26	**21.35**
r =32 (8.9M)	**1.20**	21.51
Low-Rank Adapters & Layer-Norm FT (QV)		
r =4 (0.9M)	1.29	23.32
r =16 (3.1M)	**1.16**	**21.99**
r =32 (7.3M)	1.30	22.49

4 Soft Prompting

Building on the best-performing LoRA model (r = 16, QV fine-tuning) from LoRA fine-tuning, we now integrate two additional components—soft prompting and a lightweight visual adapter—to directly tune Av-HuBERT's visual feature extractor while preserving the frozen Whisper-Audio backbone.

Finally, another common approach to fine-tuning large models is soft prompting. Soft prompting, along with a visual adapter, can effectively guide H_v to generate the subsequent contextual response via attention layers in each transformer block. We prepend $d \in 8, 64$ tunable embeddings along the time dimension. Additionally, we add an adapter module, similar to Yang et al [6]. For the mWhisper-Flamingo architecture, we add these soft prompting layers in the decoding layer, right before gated cross attention is applied along with using the original Whisper decoder.

Table 3: Clean and 0-SNR WER for `whisper_en_small_lora_rank16_QV` with soft prompting.

Model	**Clean WER (%)**	**Noisy WER (%) (0 SNR)**
Soft Prompting w/ LoRA Whisper-Audio Model		
r = 8	1.56	**22.73**
r = 64	**1.48**	23.15

5 Discussion

This paper ran experiments on many different parameter-efficient fine-tuning methods for the existing mWhisper-Flamingo model from [1]. We ran experiments with adding traditional linear adapters, LoRA, and adding soft-prompting. We discuss our observations below.

Our key observations show that LoRA rank 16 updates $< 1\%$ of the model yet recovers all clean-set accuracy and nearly 85% of the noisy audio that full fine-tuning provides. Additionally, we see that using LoRA requires 3x less trainable parameters to reach similar noise-level WER, and never fully reach noisy audio decoding accuracy. One key distinction lies in where and how the parameter updates intervene in the attention computation. Attention layers dominate the sequence modeling in encoders. By targeting these layers during task-specific fine-tuning, we change how information is routed between tokens rather than re-modeling post-attention features. When we introduce low-rank adapters directly into the query, key, and value projections–i.e. augmenting each weight matrix W_q, W_k, W_v with a small trainable update–it immediately modifies the full attention score matrix

$$A = softmax((Q + \Delta Q)(K + \Delta K)^T/\sqrt{d})$$

so every pairwise token affinity can shift in a single forward pass. Changing all these attention scores grants a global, holistic adjustment to how tokens attend to one other.

In contrast, Houlsby-style post-projection adapters, by contrast, are inserted after the attention output [9] and only influence scores in the next layer. Consequently, the adapter's influence remains confined to the current layer's activations, and it must propagate through subsequent residual and projection steps before it can modify the attention scores. The snowballed effect on the attention weights becomes less expressive.

In agreement with Yao et al, freezing W_k while adapting W_q, W_v performed better: gradients on W_v remain well-conditioned through the soft-max, avoiding the mutual-dependency problem of $Q * K$. This configuration also trims the trainable parameters, reducing the time to convergence [5].

With visual soft prompting, adding these embeddings to the already encoded audio features performed sub-optimally, requiring more research in this method of PEFT. This indicates that (i) cross-modal alignment is already good after QV-LoRA, and (ii) extra prompts introduce noise to the original audio-visual features. Benefits may emerge on truly audio-masked samples or multilingual splits where lip-reading cues may be stronger and more necessary to get a better decoding WER.

While our results establish LoRA as the most effective audio-only PEFT strategy, identifying equally efficient methods for the audio-visual setting remains an open challenge—one that calls for future work in cross-modal alignment and visual grounding under noise.

6 Citations

1. Rouditchenko, Andrew, et al. ”mWhisper-Flamingo for Multilingual Audio-Visual Noise-Robust Speech Recognition.” arXiv preprint arXiv:2502.01547 (2025).
2. Xu, Lingling, et al. ”Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment.” arXiv preprint arXiv:2312.12148 (2023).
3. Qi, Wang, et al. ”Parameter-efficient tuning on layer normalization for pre-trained language models.” arXiv preprint arXiv:2211.08682 (2022).
4. Song, Zheshu, et al. ”Lora-whisper: Parameter-efficient and extensible multilingual asr.” arXiv preprint arXiv:2406.06619 (2024).
5. Yao, Xinhao, et al. ”Theoretical Insights into Fine-Tuning Attention Mechanism: Generalization and Optimization.” arXiv preprint arXiv:2410.02247 (2024).
6. Yang, Z., et al. ”Injecting Visual Features into Whisper for Parameter-Efficient Noise-Robust Audio-Visual Speech Recognition.” Proceedings of the 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
7. Radford, Alec, et al. ”Robust speech recognition via large-scale weak supervision.” International conference on machine learning. PMLR, 2023.
8. Shi, Bowen, et al. ”Learning audio-visual speech representation by masked multimodal cluster prediction.” arXiv preprint arXiv:2201.02184 (2022).
9. Houlsby, Neil, et al. ”Parameter-efficient transfer learning for NLP.” International conference on machine learning. PMLR, 2019.
10. Hendrycks, Dan, and Kevin Gimpel. ”Gaussian error linear units (gelus).” arXiv preprint arXiv:1606.08415 (2016).
11. Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. ”Layer normalization.” arXiv preprint arXiv:1607.06450 (2016).
12. Hu, Edward J., et al. ”Lora: Low-rank adaptation of large language models.” ICLR 1.2 (2022): 3.