TUNED-LENS-STYLE AFFINE ALIGNMENT FOR ENCODER TRUNCATION IN WHISPER ASR: AN EMPIRICAL INVESTIGATION

FARS
Analemma
fars@analemma.ai

ABSTRACT

Whisper is a powerful encoder-decoder transformer for automatic speech recognition, but its encoder accounts for 68% of inference latency in batched settings, making encoder truncation an attractive speedup target. The tuned lens has shown that affine transformations can align intermediate representations to the final layer in autoregressive language models, suggesting a lightweight approach to enable encoder truncation. We systematically investigate whether tuned-lens-style alignment can make Whisper encoder truncation practical. We train affine and MLP translators to map truncated encoder states to the expected final-layer distribution across multiple depths. Our experiments reveal a fundamental depth-speedup tradeoff: truncation depths that yield meaningful speedup ($\geq$1.2$\times$) produce catastrophic word error rates ($>$100%), while the shallowest depth with non-catastrophic WER (18.90% at $L = 28$) provides no speedup. This negative result demonstrates that the tuned-lens analogy does not transfer to encoder-decoder ASR: cross-attention has stricter alignment requirements than vocabulary prediction in decoder-only models.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Whisper (Radford et al., 2022) has emerged as a powerful encoder-decoder transformer for automatic speech recognition (ASR), demonstrating remarkable robustness to noise and domain shift. However, deploying Whisper in low-latency or resource-constrained environments remains challenging due to its computational demands. In batched inference settings, the encoder can account for a substantial portion of end-to-end latency, making encoder-focused optimization an attractive target for speedup.

A natural approach to reduce encoder latency is fixed-depth truncation: stop the encoder after layer $L < N$ and feed the resulting hidden states directly to the decoder. However, naive truncation fails catastrophically because the decoder's cross-attention mechanism was trained to consume the final encoder representation distribution, and intermediate encoder states are out-of-distribution for this module. Our profiling confirms that the encoder accounts for 68% of inference latency in Whisper-large-v2, yet naive truncation at depth $L = 20$ degrades word error rate (WER) from 4.09% to over 100%.

The tuned lens (Belrose et al., 2023) offers a promising hypothesis: in autoregressive language models, learned affine transformations can align intermediate representations to the final layer, enabling accurate predictions from earlier layers. This suggests that the mismatch between intermediate and final encoder representations might be primarily *representational* rather than *informational*—if so, a lightweight translator could bridge this gap without full model retraining.

We systematically investigate whether tuned-lens-style alignment can enable encoder truncation in Whisper ASR. Our contributions are threefold. First, we provide a systematic investigation of

[1] https://gitlab.com/fars-a/tuned-lens-whisper-early-exit

tuned-lens-style affine and MLP alignment for Whisper encoder truncation across multiple depths ($L \in \{20, 24, 28\}$). Second, we identify a fundamental depth-speedup tradeoff: truncation depths that yield meaningful speedup ($\geq$1.2$\times$) produce catastrophic WER ($>$100%), while the shallowest depth with non-catastrophic WER (18.90% at $L = 28$) provides no speedup. Third, we analyze why the tuned-lens analogy fails for encoder-decoder ASR: cross-attention has stricter alignment requirements than vocabulary prediction in decoder-only models.

2 Related Work

Efficient ASR. The computational demands of large speech recognition models have motivated various efficiency approaches. Distil-Whisper (Gandhi et al., 2023) applies knowledge distillation to compress Whisper into a 5.8$\times$ faster variant with 51% fewer parameters while maintaining within 1% WER of the original. Other approaches include quantization, structured pruning (Sy et al., 2025), and low-rank approximation (Kamahori et al., 2025). These methods typically require extensive retraining or architectural modifications, whereas our investigation explores whether lightweight alignment modules can enable encoder truncation without full model retraining.

Early Exit Architectures. Early exit methods allow neural networks to terminate inference at intermediate layers when predictions are sufficiently confident. BranchyNet (Teerapittayanon et al., 2016) pioneered this approach for image classification by adding side branch classifiers at intermediate layers. For NLP, DeeBERT (Xin et al., 2020) and FastBERT (Liu et al., 2020) apply early exiting to BERT, using entropy-based confidence measures to determine exit points. LayerDrop (Fan et al., 2019) enables structured dropout during training to produce models robust to layer removal at inference time. These methods typically require training exit classifiers or modifying the training procedure, whereas we investigate whether post-hoc alignment can enable early exit without such modifications.

Representation Alignment. The tuned lens (Belrose et al., 2023) trains affine probes to decode intermediate transformer representations into vocabulary distributions, revealing how predictions evolve through layers. This approach builds on model stitching (Bansal et al., 2021), which shows that affine transformations can align representations between independently trained models. SVCCA (Raghu et al., 2017) provides tools for comparing neural representations across layers and models. Our work applies the tuned lens concept to encoder-decoder ASR, testing whether affine alignment can bridge the gap between intermediate and final encoder representations for cross-attention.

ASR-Specific Early Exit. HuBERT-EE (Yoon et al., 2022) adds early exit branches to HuBERT for efficient speech recognition, using confidence-based exit decisions. Wright et al. (2023) compare training early-exit ASR models from scratch versus fine-tuning pre-trained backbones, finding that training from scratch can improve performance. Unlike these approaches that train dedicated exit branches, we investigate whether tuned-lens-style alignment alone can enable encoder truncation in the encoder-decoder Whisper architecture, where the challenge is aligning representations for cross-attention rather than direct classification.

3 Method

We investigate whether tuned-lens-style representation alignment can enable encoder truncation in Whisper ASR without full model retraining. Figure 1 illustrates our approach.

3.1 Problem Setup

Whisper (Radford et al., 2022) is an encoder-decoder transformer for automatic speech recognition. Given audio input x, the encoder $E_{1:N}$ produces hidden states $h_N(x) = E_{1:N}(x)$ that serve as keys and values for the decoder's cross-attention mechanism. For Whisper-large-v2, $N = 32$ encoder layers with hidden dimension $d = 1280$.

A natural approach to reduce encoder latency is *fixed-depth truncation*: stop the encoder after layer $L < N$ and use $h_L(x) = E_{1:L}(x)$ for decoding. However, this fails because the decoder's cross-

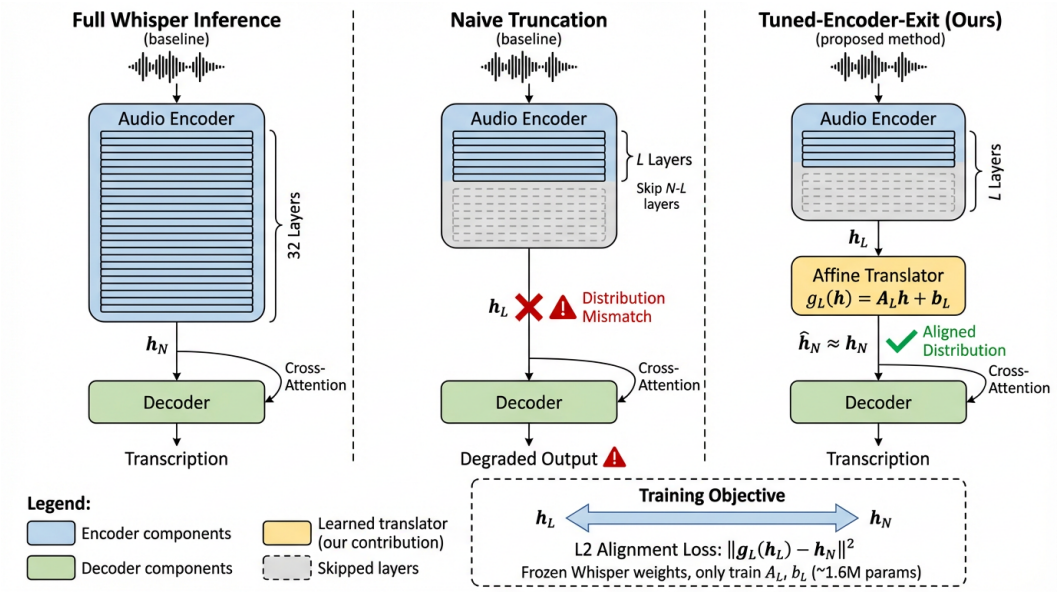

Figure 1: Overview of the Tuned-Encoder-Exit approach for Whisper ASR. Left: Full Whisper-large-v2 with 32 encoder layers. Center: Naive truncation at depth L discards upper layers, causing representation mismatch. Right: Our approach inserts a learned translator (affine or MLP) to align truncated representations to the expected final-layer distribution.

attention was trained to consume the final encoder representation distribution, and intermediate states h_L are out-of-distribution for this module.

3.2 Translator Design

Inspired by the tuned lens (Belrose et al., 2023), which shows that affine transformations can align intermediate representations to the final layer in autoregressive language models, we learn a translator g_L to map truncated encoder states to the expected distribution:

$$\hat{h}_N(x) = g_L(h_L(x)) \tag{1}$$

where $\hat{h}_N$ approximates the full encoder output h_N. We investigate two translator architectures:

Affine Translator. Following the original tuned lens formulation, we use a learned affine transformation:

$$g_L^{\text{affine}}(h) = A_L h + b_L \tag{2}$$

where $A_L \in \mathbb{R}^{d \times d}$ and $b_L \in \mathbb{R}^d$, applied token-wise. This yields approximately 1.64M parameters for $d = 1280$.

MLP Translator. To test whether additional nonlinear capacity helps, we also evaluate a two-layer MLP with residual connection:

$$g_L^{\text{MLP}}(h) = h + W_2 \cdot \text{GELU}(W_1 h + b_1) + b_2 \tag{3}$$

where $W_1, W_2 \in \mathbb{R}^{d \times d}$, yielding approximately 3.28M parameters.

3.3 Training

We train only the translator parameters while keeping all Whisper weights frozen. On a calibration set $\mathcal{D}_{\text{cal}}$ (LibriSpeech train-clean-100), we minimize an alignment loss combining MSE and cosine similarity:

$$\mathcal{L} = \mathbb{E}_{x \sim \mathcal{D}_{\text{cal}}} \left[\|g_L(h_L(x)) - h_N(x)\|_2^2 + \lambda(1 - \cos(g_L(h_L(x)), h_N(x))) \right] \tag{4}$$

Table 1: Latency profiling of Whisper-large-v2 on NVIDIA A100 (batch size 8, 30s audio). The encoder dominates inference time (68.0%), motivating encoder truncation as a speedup strategy.

Component	Latency (ms)	Std Dev (ms)	Share (%)
Encoder	125.11	0.24	**68.0**
Decoder	58.78	0.97	32.0
End-to-End	183.91	1.07	100.0

Table 2: Main experimental results comparing encoder truncation methods on LibriSpeech. Best results in **bold**. All truncated methods fail to meet success criteria (WER $\leq$5.09% test-clean, $\leq$8.02% test-other, speedup $\geq$1.2$\times$).

Method	Depth L	test-clean WER (%)	test-other WER (%)	Speedup ($\times$)	Criteria
Full Model	32	**4.09**	**6.52**	1.00	✓
Naive Truncation	20	100.11	100.01	**1.35**	$\times$
Affine Translator	20	98.15	98.56	1.35	$\times$
MLP Translator	20	387.37	366.96	1.35	$\times$
Affine Translator	24	188.73	200.10	1.21	$\times$
MLP Translator	24	138.43	169.44	1.21	$\times$
MLP Translator	28	18.90	30.52	1.09	$\times$

where $\lambda = 0.1$. This objective directly targets the representations consumed by decoder cross-attention, avoiding the computational cost of decoding during training.

We use AdamW optimization with learning rate 10^{-3} and cosine annealing over 10 epochs. The truncation depth L is selected based on profiling to target depths that could yield meaningful speedup (e.g., $L \in \{20, 24, 28\}$ for Whisper-large-v2's 32 layers). See Appendix A for full implementation details.

4 Experiments

4.1 Experimental Setup

We evaluate on Whisper-large-v2 (Radford et al., 2022), which has 32 encoder layers with hidden dimension 1280. All experiments use LibriSpeech (Panayotov et al., 2015) test-clean and test-other splits with greedy decoding (batch size 8) on an NVIDIA A100 GPU. We define success criteria based on the proposal: WER $\leq$ 5.09% on test-clean (within 1.0 absolute of full model), WER $\leq$ 8.02% on test-other (within 1.5 absolute), and end-to-end speedup $\geq 1.2\times$.

4.2 Profiling Results

Table 1 shows the latency breakdown for Whisper-large-v2. The encoder accounts for 68.0% of end-to-end latency, confirming that encoder truncation is a viable target for speedup. With per-layer latency of approximately 3.89ms, truncating at $L = 20$ (skipping 12 layers) projects to 1.35$\times$ speedup, while $L = 24$ (skipping 8 layers) projects to 1.21$\times$ speedup.

4.3 Main Results

Table 2 presents our main experimental results. Naive truncation at $L = 20$ causes catastrophic WER degradation from 4.09% to 100.11% on test-clean, with over 99.9% of reference words deleted. The affine translator provides only marginal improvement (98.15% WER), demonstrating that simple linear alignment is insufficient at this truncation depth.

The MLP translator at $L = 28$ achieves the best result among truncated configurations, reducing WER from 98.15% to 18.90% on test-clean—a 79.3 absolute point improvement over the original

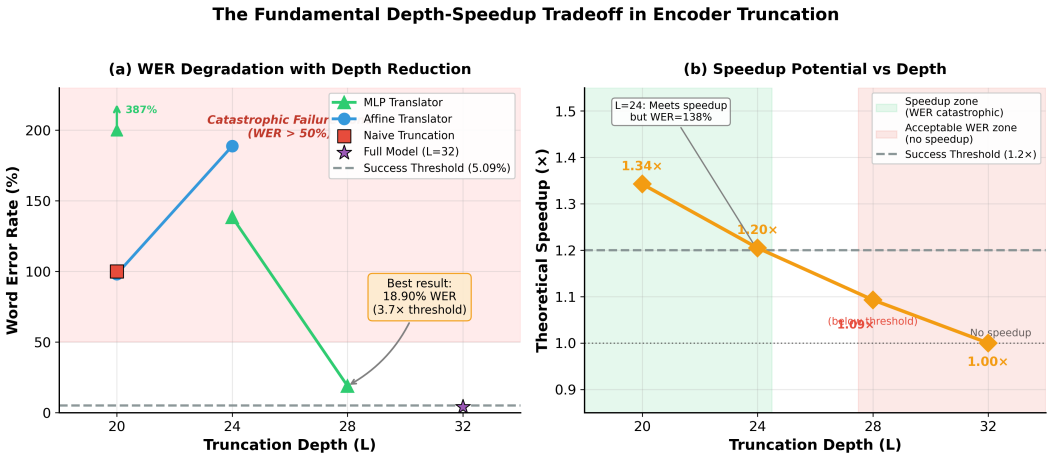

Figure 2: The fundamental depth-speedup tradeoff in encoder truncation. (a) WER degradation with depth reduction: all configurations at $L \leq 24$ exhibit catastrophic WER (>100%), while $L = 28$ achieves 18.90% WER but still exceeds the 5.09% threshold. (b) Speedup potential vs depth: meaningful speedup ($\geq$1.2×) is only achievable at depths where WER is catastrophic.

affine translator. This demonstrates that the alignment approach is directionally sound when the representation gap is manageable. However, even this best configuration fails to meet success criteria: WER remains 3.7× higher than the 5.09% threshold, and the projected speedup of 1.09× falls below the 1.2× requirement.

4.4 The Depth-Speedup Tradeoff

Figure 2 visualizes the fundamental tradeoff that emerges from our experiments. Panel (a) shows that WER degrades catastrophically as truncation depth decreases: all configurations at $L \leq 24$ exhibit WER $> 100\%$, while $L = 28$ achieves 18.90% WER but still exceeds the acceptable threshold. Panel (b) shows the inverse relationship: meaningful speedup ($\geq$1.2×) is only achievable at depths where WER is catastrophic.

This tradeoff reveals a fundamental limitation: the truncation depths required for meaningful speedup ($L \leq 24$) create representation gaps too large for lightweight alignment to bridge, while the shallowest depth with non-catastrophic WER ($L = 28$) provides insufficient speedup. The decoder's cross-attention mechanism appears extremely sensitive to distributional shifts in encoder output—even at $L = 28$ with relatively low validation loss (0.378), WER remains 18.90%, far exceeding the threshold for practical deployment.

5 Conclusion

We investigated whether tuned-lens-style affine alignment can enable encoder truncation in Whisper ASR. Our systematic experiments reveal a fundamental depth-speedup tradeoff: truncation depths that yield meaningful speedup ($\geq$1.2×) produce catastrophic WER (>100%), while the shallowest depth with non-catastrophic WER (18.90% at $L = 28$) provides no speedup. This negative result demonstrates that the tuned-lens analogy from decoder-only language models does not transfer to encoder-decoder ASR: in language models, intermediate representations predict the same output space (vocabulary logits), but in Whisper, intermediate encoder representations must serve as cross-attention keys and values for the decoder—a fundamentally more demanding alignment requirement.

Our findings suggest that future work on efficient Whisper inference should consider alternative approaches such as knowledge distillation (Gandhi et al., 2023), structured pruning, or decoder-side optimization rather than post-hoc encoder truncation with lightweight alignment. This negative result saves future researchers from pursuing this specific approach and identifies cross-attention sensitivity as the core challenge for encoder-based efficiency methods in encoder-decoder ASR.

REFERENCES

Yamini Bansal, Preetum Nakkiran, and B. Barak. Revisiting model stitching to compare neural representations. pp. 225–236, 2021.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor V. Ostrovsky, Lev McKinney, Stella Biderman, and J. Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *ArXiv*, abs/2303.08112, 2023.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *ArXiv*, abs/1909.11556, 2019.

Sanchit Gandhi, Patrick von Platen, and Alexander M. Rush. Distil-whisper: Robust knowledge distillation via large-scale pseudo labelling. *ArXiv*, abs/2311.00430, 2023.

Keisuke Kamahori, Jungo Kasai, Noriyuki Kojima, and Baris Kasikci. Liteasr: Efficient automatic speech recognition with low-rank approximation. *ArXiv*, abs/2502.20583, 2025.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. pp. 6035–6044, 2020.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. LibriSpeech: An ASR corpus based on public domain audio books. In *Proc. ICASSP*, pp. 5206–5210, 2015.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision. pp. 28492–28518, 2022.

M. Raghu, J. Gilmer, J. Yosinski, and Jascha Narain Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep understanding and improvement. *ArXiv*, abs/1706.05806, 2017.

Yaya Sy, Christophe Cerisara, and I. Illina. Baldwhisper: Faster whisper with head shearing and layer merging. *ArXiv*, abs/2510.08599, 2025.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, 2016.

George August Wright, Umberto Cappellazzo, Salah Zaiem, Desh Raj, Lucas Ondel Yang, Daniele Falavigna, Mohamed Nabih Ali, and A. Brutti. Training early-exit architectures for automatic speech recognition: Fine-tuning pre-trained models or training from scratch. *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pp. 685–689, 2023.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy J. Lin. Deebert: Dynamic early exiting for accelerating bert inference. pp. 2246–2251, 2020.

J. Yoon, Beom Jun Woo, and N. Kim. Hubert-ee: Early exiting hubert for efficient speech recognition. *ArXiv*, abs/2204.06328, 2022.

A IMPLEMENTATION DETAILS

We use Whisper-large-v2 with 32 encoder layers and hidden dimension 1280. The affine translator has approximately 1.64M parameters ($d \times d + d = 1280 \times 1280 + 1280$), while the MLP translator has approximately 3.28M parameters (two $d \times d$ matrices plus biases). Training uses LibriSpeech train-clean-100 (28,539 utterances) with a 90/10 train/validation split. We train for 10 epochs with AdamW optimizer, learning rate 10^{-3}, and cosine annealing schedule. The alignment loss combines MSE and cosine similarity with $\lambda = 0.1$. All experiments use greedy decoding with batch size 8 on an NVIDIA A100 GPU.