SLIMDUCER: A STREAMABLE SPEECH LLM WITH 3M TRAINABLE PARAMETERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Speech is normally given to a language model at its input, where encoder features occupy context positions the model was never pre-trained to read. Slimducer attaches at the other end: a 3 M-parameter joint network writes into the language model's own frozen output projection, so the model sees only its own vocabulary indices, beginning at `bos`, as in pre-training. Nothing else trains — $0.38\,\%$ of 783.7 M, with the speech encoder and the language model frozen and its $151{,}936$-way head unchanged — and every model reported here was trained on a single 24 GB consumer GPU.

Keeping that head whole is what removing the label axis from the alignment lattice buys: no $T \times L \times |\mathcal{V}|$ tensor is retained for a backward pass, where predictor-side transducers must cut their vocabulary to a few thousand. Decoding is frame-synchronous, so bounded audio chunks reproduce whole-utterance decoding — streaming that an input-side coupling cannot offer at all, having to see the utterance before the language model attends over it.

The same 3 M parameters carry beyond transcription. On LibriSpeech Slimducer is slightly behind a fully fine-tuned wav2vec 2.0 of thirty times its trainable size and $18\,\%$ to $39\,\%$ better relative outside that domain; with a 3.4 M text-only adapter on the frozen backbone it translates speech into German at 14.98 BLEU against 12.09 for a cascade of the same backbone, and answers questions about the utterance just recognised.

1 INTRODUCTION

Speech is increasingly the front end to a language model rather than an end in itself C. Kim, A. Misra, K.K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani (2017). That makes the interface between the two — where the audio meets the language model, and what the language model must give up to accept it — as much the design question as recognition accuracy is. All-neural recognisers have long since overtaken HMM/WFST systems Chiu et al. (2018); C. Kim et al. (2019), and CTC Graves et al. (2006) and the transducer Graves (2012a) are the two alignment-free losses on which most of them are built.

Almost every system that attaches such a recogniser to a large language model couples at the LLM's *input*: encoder features are projected into the token-embedding space and placed among the text tokens (Sec. 2.3). Freezing the LLM's weights is not the same as leaving it alone. A text LLM was never pre-trained on a context holding anything but its own token embeddings, so an input-side coupling runs it off its training distribution even when no parameter moves. The few systems that attach at a transducer's predictor do keep the LLM reading token indices, but pay for it by retraining its output side and shrinking the vocabulary it predicts over. Appendix A sets out what each system changes; no row but ours leaves the language model as it was.

Slimducer attaches at that same end and changes nothing. The joint network writes into the LLM's own frozen `lm_head`; because Qwen3 and Llama-3.2 tie that matrix to their input embeddings, the audio is projected into exactly the space an input-side system aims at, but read out as the nearest token rather than injected at layer zero. The language model then sees a sequence of its own vocabulary indices beginning at `bos`, as in pre-training. Three million parameters, $0.38\,\%$ of the model, are all that learn anything — and because the backbone is untouched, what it could already do from a text prefix it can do from the recognition cache.

Contributions.

- **One extremely small trainable fraction, carried across speech tasks and domains.** $3.0\,\mathrm{M}$ parameters train, $0.38\,\%$ of the model, over Qwen3's unmodified $151{,}936$-way head where prior predictor-side systems cut theirs to $5{,}000$ (Deng et al., 2025) or $20{,}000$ (Tang et al., 2026). Because the backbone is untouched the same parameters reach past transcription: the identical checkpoint translates English speech into German its joint network never saw and answers questions about the utterance just heard, reaching 14.98 BLEU against 12.09 for a cascade of the same backbone once a $3.4\,\mathrm{M}$ text-only adapter is added (Sec. 4.3), and it transcribes $18\,\%$ to $39\,\%$ better relative than a matched fully fine-tuned wav2vec 2.0 outside the training domain (Sec. 4.6).
- **A collapsed lattice that keeps the vocabulary whole.** Removing the label axis removes the $T \times L \times |\mathcal{V}|$ tensor a transducer retains for its backward pass — the term that forces predictor-side systems to shrink their output space. One Viterbi alignment plus ordinary cross-entropy then suffices, with no forward–backward and no custom gradient.
- **Streaming, which the existing couplings cannot do at all.** An input-side system writes the encoder's output into the LLM's context and attends over those positions at every output step, so the utterance must exist before generation begins; a predictor-side transducer does stream, but over a vocabulary that is not the LLM's, so what streams out needs a conversion step before the language model can read it. Slimducer has neither obstacle, and with the encoder's attention already chunk-bounded, bounded audio chunks reproduce whole-utterance decoding exactly at $8\,\mathrm{s}$ and cost 5.1 WER points at $1\,\mathrm{s}$, with no architectural change and no retraining (Sec. 4.5).

2 Related Work

2.1 End-to-End Speech Recognition: CTC and the Transducer

CTC Graves et al. (2006) sums over every monotonic alignment between a short label sequence and a long frame sequence in closed form, at the cost of assuming conditional independence between output frames. The transducer Graves (2012b) removes that assumption with a prediction network conditioned on the labels emitted so far, at the cost of a (T, L) joint lattice and a correspondingly expensive loss. The Transformer Transducer Zhang et al. (2020), Conformer Gulati et al. (2020) and ContextNet Han et al. (2020) vary the encoder under that loss and remain the standard streamable backbones. The Hybrid Autoregressive Transducer Variani et al. (2020) factorizes the output into a blank probability and a blank-free token distribution. Slimducer reuses that factorization for a different purpose than HAT's density-ratio fusion: it is what lets the token half be exactly a frozen LLM's own softmax, with a one-unit blank head carrying what the transducer normally asks the whole output layer to represent. Making both lattice moves advance time is not new: the Recurrent Neural Aligner Sak et al. (2017) and the monotonic transducer Tripathi et al. (2019) already restrict a frame to at most one token, and Viterbi training against a hard alignment is older still. What those systems did not do — and what this paper is about — is use the resulting freedom from the L axis to write into a frozen language model's own output projection, so that the token distribution is the language model's rather than a learned one.

2.2 Self-Supervised and Weakly-Supervised Speech Encoders

wav2vec 2.0 Baevski et al. (2020) and HuBERT Hsu et al. (2021) learn acoustic representations from unlabelled audio and are then fine-tuned end to end with a small CTC or transducer head — the design our baselines come from. Whisper Radford et al. (2022) instead trains supervised on $680{,}000$ hours of weakly labelled transcripts, with an autoregressive decoder learned from scratch alongside the encoder rather than reused from an existing language model. Slimducer's AuT encoder is itself distilled from a larger audio-language system, but what distinguishes Slimducer from all three lies downstream of the encoder: none of them couples its output to an otherwise untouched text LLM's own vocabulary and output projection.

2.3 Coupling Large Language Models with Speech

The dominant recipe projects encoder features into the LLM's *input* embedding space and interleaves them with the text tokens, so a pre-trained LLM attends over audio as it would over a text prefix. SALMONN Tang et al. (2024) adapts the LLM with LoRA Hu et al. (2021); Qwen2-Audio Chu et al. (2024) and AudioPaLM Rubenstein et al. (2023) fine-tune it, AudioPaLM also extending its vocabulary with discrete audio tokens; SpeechGPT Zhang et al. (2023) folds HuBERT units into the text vocabulary outright. Ultravox Fixie.ai (2025) keeps its Llama-3.1 frozen and trains only an adapter, but fine-tunes the Whisper encoder. It also adds a distillation loss asking the coupled model to match the *text* backbone's own logits — an explicit correction for a context that carries audio embeddings the backbone was never pre-trained to read. SLAM-ASR Ma et al. (2024) pares the recipe down to one linear projector between a frozen encoder and a frozen LLM, the closest prior work in spirit. The coupling is still at the input: audio occupies context positions the LLM attends over at every output step.

Coupling at the transducer's predictor. The factorized neural transducer (Chen et al., 2022) splits blank from vocabulary prediction so that the vocabulary branch is a standalone language model, and combines the branches by *adding log-probabilities* — which leaves the emitted distribution a renormalized product of two models rather than the backbone's own. Two systems place an LLM in that branch, and both shrink the output space to make the transducer grid affordable. Transducer-Llama (Deng et al., 2025) freezes the LLM's layers but retrains its embedding and output layers over $5{,}000$ ASR tokens instead of Llama3's $128{,}000$, reaching $2.47\,\%$/$6.53\,\%$ on `test-clean`/`test-other`. TRADE (Tang et al., 2026) uses LoRA, unfreezes the encoder's top six layers, and derives a $20{,}000$-token vocabulary from the LLM's, reaching $3.13\,\%$ on `test-other`. Slimducer trains neither tower and neither embedding matrix, keeping the $151{,}936$-way `lm_head` as pre-training left it; its `test-other` lands next to Transducer-Llama's ($0.064, 8$ against $0.065, 3$), while TRADE, which trains the most, is ahead of both. Appendix A tabulates what every system changes, together with the consequence for reusing the transcript.

2.4 Streaming and Chunked Encoders

Low-latency streaming ASR is usually reached by restricting self-attention to a local or growing causal window during training — Emformer Shi et al. (2021) is the widely used instance — or by training a dedicated streaming model against a latency penalty. Slimducer does neither. AuT's own chunk-bounded attention, inherited unmodified from Qwen3-ASR, already caps every attention computation at `n_window_infer` frames. The chunk–latency trade-off falls out of an encoder never trained with streaming as an objective (Sec. 4.5).

3 Slimducer Algorithm

Slimducer places a small trainable joint network between two frozen towers and reads its output through the language model's own frozen output projection. This section gives the architecture, the emission rule, and the training loop.

3.1 Architecture

Let $n(t)$ be the number of tokens emitted strictly before frame t. The joint network combines a frame representation $\mathbf{h}_t^{\mathrm{aud}} \in \mathbb{R}^{d_a}$ from the frozen speech encoder with the frozen language model's state $\mathbf{h}_\ell^{\mathrm{lm}} \in \mathbb{R}^d$ after ℓ emitted tokens,

$$\mathbf{z}_t = W_o\,\sigma\Big(W_{\mathrm{lm}}\,\mathrm{LN}_{\mathrm{lm}}(\mathbf{h}_{n(t)}^{\mathrm{lm}}) + W_{\mathrm{aud}}\,\mathrm{LN}_{\mathrm{aud}}(\mathbf{h}_t^{\mathrm{aud}}) + \mathbf{b}\Big) \in \mathbb{R}^d, \tag{1}$$

with σ a pointwise nonlinearity, d the language model's hidden size and d_a the encoder's. The encoder output is taken *below* that encoder's own projection into the language model's input space, which exists to serve the prefix-based coupling this work replaces and is discarded. The language model is used only as an autoregressive function of the emitted prefix, starting from $\langle\mathrm{bos}\rangle$, with no chat template and no instruction prefix; the audio never enters its context. The output width is fixed by the frozen projection, which consumes $\mathbf{z}_t$: the joint must return what the backbone would

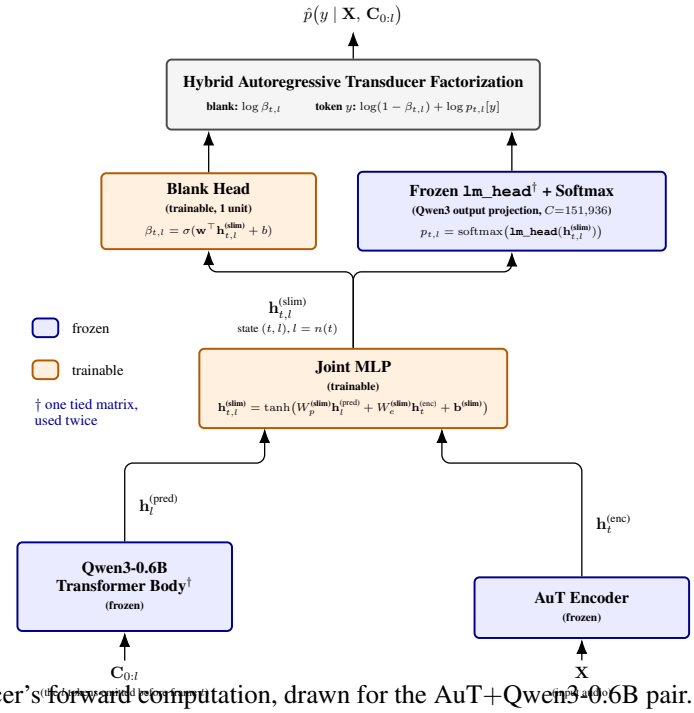


Figure 1: Slimducer's forward computation, drawn for the AuT+Qwen3-0.6B pair. **Blue** is frozen, **orange** is trainable: the joint MLP and the one-unit blank head, and nothing else. The transformer body and `lm_head`, both marked †, are the *same* tied matrix used twice. The joint output $\mathbf{h}^{(\text{slim})}_{t,l}$ at state (t,l), $l = n(t)$, feeds the blank head and the frozen `lm_head`; the HAT factorization combines them into one next-token distribution. Either tower may be swapped, so the frozen count is a property of the pair — 780.7 M here, 0.5–4.3 B across Table 1 — while the trainable side is set by the joint width and stays at 3.0 M (0.38% of 783.7 M here).

have handed its own softmax. The two layer normalizations matter — the towers were pre-trained independently and their activation scales are unrelated, so without them one term dominates by construction rather than by evidence (Sec. O.1).

Factorized output. Because frames outnumber tokens, most frames emit nothing. Rather than adding a blank symbol to the vocabulary, the emission decision is factorized out of the token distribution,

$$\beta_t = \text{sigmoid}(\mathbf{w}_b^\top \mathbf{z}_t + b_b), \tag{2}$$

$$p_t(\cdot) = \text{softmax}(W_{\text{out}}\, \mathbf{z}_t)\,, \tag{3}$$

where W_{out} is the backbone's frozen output projection and $\mathbf{w}_b \in \mathbb{R}^d$ is the only additional weight vector. The factorization follows the hybrid autoregressive transducer (Variani et al., 2020). Folding blank into the vocabulary softmax would put it in the same normaliser as the words, and equation 3 would no longer be the backbone's own distribution. Blank also gets a learned direction rather than a reserved vocabulary entry.

Fig. 1 shows the full data flow. Only the joint MLP and the single-unit blank head carry gradients; the speech encoder and the language model are both frozen, and the transformer body producing $\mathbf{h}^{(\text{pred})}_l$ and the `lm_head` mapping $\mathbf{h}^{(\text{slim})}_{t,l}$ to a token distribution are the *same* tied matrix used twice. The blank probability $\beta_{t,l}$ and the token distribution $\hat{z}_{t,l}$ are combined by the Hybrid Autoregressive Transducer factorization Variani et al. (2020) rather than folded into one shared softmax, so $\hat{z}_{t,l}$ remains exactly the frozen LLM's own output distribution.

3.2 Frame-synchronous emission

Output is produced at the frame rate, but the language model advances only when a token is emitted. Concretely, $\mathbf{h}^{\text{lm}}_{n(t)}$ is reused unchanged across every frame that emits blank, and the backbone is stepped exactly once per emitted token. The number of backbone invocations is therefore L, not T.

This is the source of the model's efficiency claim, and it compounds with the architectural one: in prefix-based coupling the acoustic frames both traverse every layer *and* lengthen the attention context for all subsequent text positions, whereas here the context holds only emitted text.

Algorithm 1 SlimDucer decoding (greedy, frame-synchronous)

$\mathbf{h}^{\mathrm{aud}}_{1:T} \leftarrow \mathrm{Enc}(\mathbf{x})$
$\mathbf{h}^{\mathrm{lm}} \leftarrow \mathrm{LM}(\langle \mathrm{bos} \rangle)$, $\hat{y} \leftarrow [\,]$
for $t = 1$ **to** T **do**
  $\mathbf{z} \leftarrow \mathrm{Joint}(\mathbf{h}^{\mathrm{lm}}, \mathbf{h}^{\mathrm{aud}}_t)$ {Eq. equation 1}
  if $\mathrm{sigmoid}(\mathbf{w}_b^\top \mathbf{z} + b_b) \geq \tau$ **then**
    continue {blank: the backbone is not invoked}
  end if
  $y \leftarrow \arg\max_v [W_{\mathrm{out}}\mathbf{z}]_v$, append y to $\hat{y}$
  $\mathbf{h}^{\mathrm{lm}} \leftarrow \mathrm{LM}(\hat{y})$ {one incremental step, cached}
end for
return $\hat{y}$

The emission threshold τ in Algorithm 1 is a deployment knob rather than a property of the weights: because the emission decision is a single sigmoid, the deletion/insertion operating point can be moved without retraining.

3.3 Training

Training alternates hard alignment and teacher forcing, in the manner of Viterbi training. Given the current model, each utterance is aligned against its reference; the resulting frame-level targets are then fitted; and the cycle repeats.

Alignment. An alignment is a monotonic assignment of the L reference tokens to L distinct frames, in order, with the remaining frames labelled blank. Writing $S[t, \ell]$ for the best score of the first t frames having emitted ℓ tokens,

$$S[t,\ell] = \max\Big\{ \underbrace{S[t-1,\ell] + \log \beta_{t,\ell}}_{\text{blank}},\ \underbrace{S[t-1,\ell-1] + \log(1-\beta_{t,\ell-1}) + \log p_{t,\ell-1}(y_\ell)}_{\text{emit } y_\ell} \Big\}, \quad (4)$$

with $S[0,0] = 0$ and $S[0,\ell>0] = -\infty$, and the alignment recovered by backtracing from $S[T,L]$.

The second subscript on β and p is what makes this a grid rather than a scan. The backbone is autoregressive, so the score of emitting y_ℓ at frame t depends on the state after $\ell - 1$ tokens: "which token" and "how many so far" are not separable, and the joint must be evaluated on the $(T, L{+}1)$ grid rather than at T points. Each training step therefore runs two passes over the joint, and they should not be conflated. The *alignment pass* above covers the grid, but under `no_grad` and chunked over t, so nothing it computes is retained for a backward pass. The *fitting pass* that follows evaluates the joint at the T points of the chosen path, and is the only one that carries gradients. Where this paper says the joint is evaluated T times, it is the fitting pass that is meant.

Objective. Given an alignment $a : \{1..T\} \to \{\varnothing\} \cup \{1..L\}$, the model is fitted with teacher forcing against the *reference* prefix, never its own predictions:

$$\mathcal{L} = -\sum_{t\,:\,a(t)=\varnothing} \log \beta_t \;-\; \sum_{t\,:\,a(t)=\ell} \Big[\log(1-\beta_t) + \log p_t(y_\ell) \Big]. \quad (5)$$

Because the alignment is fixed rather than marginalized, the objective sees $(T, |\mathcal{V}|)$ and never $(T, L, |\mathcal{V}|)$ — which is what makes a vocabulary two orders of magnitude larger than a transducer's usual inventory affordable. The price is a dependence on *when* the alignment is recomputed and on *what* produces the first one.

3.4 The Loss Is an Ordinary Cross-Entropy

CTC and the transducer both define their loss as a marginal over *every* monotonic alignment. That forces a forward–backward pass at training time and a custom autograd node whose backward is

Algorithm 2 SlimDucer training, with a rolling alignment window

initialize the joint and the blank head at random
$A \leftarrow$ UNIFORMALIGNMENT(all utterances) {round zero consults no model; Section 3.4}
$s \leftarrow 0$
while $s < S_{\max}$ **do**
  $B \leftarrow$ the next W minibatches of the shuffled stream {drawn before they are used, not after}
  for all utterances in B **do**
    build the $(T, L{+}1)$ grid, solve equation 4, store in A
  end for
  for all $b \in B$ **do**
    minimize equation 5 on b against the targets in A; $s \leftarrow s + 1$
  end for
end while

derived by hand, because the gradient has to be assembled from α and β rather than read off the forward computation.

Slimducer does not marginalise. Each utterance is aligned once by Viterbi against the model's current parameters. The loss is then an ordinary cross-entropy against that fixed alignment: binary on the blank head, categorical on the frozen `lm_head`, nothing else.

$$\mathcal{L} = \sum_{t\,:\,a(t)=\varnothing} \mathrm{BCE}(\mathrm{blank}_t) + \sum_{t\,:\,a(t)\neq\varnothing} \mathrm{CE}\big(\mathrm{token}_t,\, c_{a(t)}\big)\,. \tag{6}$$

The implementation is two calls to `binary_cross_entropy_with_logits` and `cross_entropy`, with autograd supplying the rest.

The marginal is available as an option, and we measured both on the same four seeds: the forced alignment is better on dev-clean WER and three times faster per step (Appendix N).

Two choices in this loop are decided empirically rather than by default. Round zero aligns against a uniform spacing of the L tokens over the T frames rather than against the initialized model, and the step budget is set by where the hard assignment settles rather than carried over from another recipe; both are measured in Appendix O.1 and Appendix O.1.

Slimducer is a monotonic transducer Sak et al. (2017); Tripathi et al. (2019) whose prediction network is a frozen language model and whose joint output layer is that model's frozen unembedding; the prediction network therefore never sees the paired transcripts and the label distribution is the backbone's own. Appendix E states the lattice, its two moves and the Viterbi objective formally, and places the construction against work on what a transducer's prediction network actually learns.

3.5 THE ALIGNMENT PROBLEM, AND WHERE SLIMDUCER SITS

CTC, the transducer and Slimducer face the same mismatch: T frames against $L \leq T$ tokens, with no correspondence given. All three sum or maximise over a lattice of monotonic alignments and differ in the moves that lattice permits.

CTC works over the extended label sequence $\varnothing\, c_0\, \varnothing\, c_1 \ldots c_{L-1}\, \varnothing$ of length $2L{+}1$, advancing its position by 0, 1, or 2 per frame — the skip being permitted only between two distinct labels. Every move advances t, so a path is exactly T steps, but each frame is scored on the audio alone. The transducer conditions on the emitted prefix and pays for it with a move that advances l while t stands still: a path runs $T + L$ steps and the joint is evaluated over the whole $T \times (L{+}1)$ grid.

Slimducer keeps t advancing on every move — blank leaves l alone, emission increments it — while conditioning on the prefix. A path is exactly T steps and the state is (t, l) with $l = n(t)$ fixed by the path, so one Viterbi alignment followed by an ordinary cross-entropy is sufficient (Sec. 3.4).

4 EXPERIMENTAL RESULTS

4.1 EXPERIMENTAL CONFIGURATION

What is trained. The joint network and the one-unit blank head, 3.0 M parameters. The audio tower and the language model are frozen (`-freeze_audio`), 780.7 M between them for the AuT / Qwen3-0.6B pair, and `lm_head` is used unchanged; being tied to the input embeddings, it introduces no trained output layer. Frozen towers are held in `bfloat16`, the joint and blank head in `float32`.

Optimiser and schedule. AdamW with a warmup–stable–decay schedule, linear in each phase and decaying to zero, at a joint learning rate of 5×10^{-4}. Batches are formed by duration rather than utterance count: 48 s of audio per step with no gradient accumulation. Appendix J gives the remaining hyper-parameters and the step allocation at each data scale.

Decoding and scoring. Every number is greedy, `float32`, batch size one, on the official `test-clean` and `test-other` splits, with no language model and no rescoring. At batch size one, under `feat_extract_norm="group"` the padding inside a batch changes the output at valid frames, so one checkpoint scores differently at different batch sizes. The encoder buffer is fixed at 3,000 mel frames (30 s), excluding 9 of `test-clean`'s 2,620 utterances and 7 of `test-other`'s 2,939; we score 2,611 and 2,932.

Text normalisation differs by evaluation set and is stated with every table; Appendix F says which is used where and why.

4.2 SPEECH RECOGNITION RESULTS

We fine-tune Slimducer on LibriSpeech-derived subsets of increasing size and evaluate on the official held-out `test-clean` and `test-other` splits, greedy and without a language model. The audio tower is frozen in every run, so only the joint MLP and the one-unit blank head are trained — 3.0 M parameters against 780.7 M frozen. Configuration is in Sec. 4.1; the counts scored are 2,611 and 2,932, the full splits minus the utterances longer than the encoder's 30 s buffer.

Table 1: LibriSpeech test splits, greedy, no language model, no text normalisation. Every Slimducer row trains the same 3.0 M-parameter joint and freezes everything else; wav2vec 2.0 rows are that paper's own 960 h no-LM numbers Baevski et al. (2020). The 9.74 h row is mean ± s.d. over four seeds at 25,000 steps, the rest single runs. Appendix tables quote the same setting at 6,250 steps, where it scores better (0.124, 2); Sec. 3.4 is why.

Speech encoder	Language model	Data	test-clean	test-other	Trained
Slimducer — joint network only					
AuT	Qwen3-0.6B	9.74 h	13.60 ± 0.30	18.98 ± 0.33	3.0 M
AuT	Qwen3-0.6B	100 h	6.87	13.41	3.0 M
AuT	Qwen3-0.6B	960 h	4.25	9.37	3.0 M
WavLM-Large	Qwen3-4B	960 h	3.77	6.63	3.0 M
WavLM-Large	Llama-3.2-1B	960 h	**3.71**	**6.48**	3.0 M
wav2vec 2.0 — every parameter fine-tuned Baevski et al. (2020)					
BASE		960 h	3.4	8.6	95 M
LARGE		960 h	**1.8**	**3.3**	317 M

The relevant comparison is against BASE, not LARGE: 317M parameters, all fine-tuned, is a different budget than Slimducer's 3.0M, whereas BASE's 95M is the closer match once Slimducer's frozen backbone is counted (larger in total, but none of it receives a gradient). At 9.74 h Slimducer trails BASE's own low-resource numbers; at 960 h it matches BASE on `test-clean` and beats it on `test-other` by 25 % relative, training under 0.4 % of the parameters BASE fine-tunes. Which system had to move less to get there is Sec. 4.6's question: outside LibriSpeech the ordering against BASE reverses on every set tried.

4.3 Speech Translation and Spoken Language Understanding

The indices the joint emits are the language model's own, and it advances its key/value cache while recognising. When an utterance ends the model is in the state it would be in had it read that sentence as text, and whatever it could do from a text prefix it can do from that cache with no further training.

Speech translation. We take the $960\,\mathrm{h}$ checkpoint unchanged, recognise CoVoST-2 (Wang et al., 2021) `en→de` audio, and continue generating from the cache after a two-shot `English:/German:` prompt. The joint network has seen nothing but English LibriSpeech and no translation data of any kind; the German comes entirely from the frozen backbone. Over 150 utterances this reaches 6.59 BLEU (Table 2).

Two references bracket it: the *reference* transcript through the same prompt reaches 10.52, so recognition costs 3.93 BLEU, and a cascade of `wav2vec2-base-960h` into the same backbone — the system a reader would otherwise build — reaches 3.94, worse by $67\,\%$ relative even though that wav2vec 2.0 is *ahead* of Slimducer on `test-clean`. The transcripts show why: with 26 uppercase letters and an apostrophe, wav2vec 2.0 fails into non-words the backbone cannot use, while Slimducer's errors are real words in its vocabulary. A $3.4\,\mathrm{M}$ LoRA trained on CoVoST-2's *text* pairs alone — never seeing audio, the joint untouched — raises the level but not the ordering: 31.83, 14.98 and 12.09.

Recognition and translation must run through the *same* adapted instance, since a key/value cache belongs to the weights that produced it; with one instance throughout, reading the cache (14.98) and re-encoding the transcript (15.03) agree to 0.05 BLEU, so the transcript never needs re-encoding (Appendix K).

One backbone does both jobs, so the adapter has to be charged against recognition. Changing only the backbone the joint reads from, word error rate on the same 150 utterances goes from $27.70\,\%$ to $29.84\,\%$ — 2.14 points, $7.7\,\%$ relative — against $39.2\,\%$ for the cascade's own CTC recogniser on this audio (60 utterances, measured separately).

Spoken language understanding. The same continuation answers questions about what was just said. Prompted with `The speaker is talking about` after recognising *"for instance one day the children had been playing upon the piazza…"*, the model produces *"the children's play and the way they left things on the piazza"*. Recognition of that $18.4\,\mathrm{s}$ utterance took $1.06\,\mathrm{s}$ and the answer $0.34\,\mathrm{s}$ more, at 0.049 to 0.058 times real time. Generation still begins only after the utterance does, so this is a property of the input side and not a read/write policy (Ma et al., 2019).

Quantitatively, intent classification over SLURP's 101 classes (Bastianelli et al., 2020) with two in-context examples reaches $7.0\,\%$ over 200 utterances, seven times the $1.0\,\%$ chance rate, from a checkpoint trained only to transcribe and with no adapter. Scoring the *reference* transcript through the same prompt gives $10.0\,\%$, so recognition errors cost 3.0 points and the remaining gap to a usable classifier belongs to the $1\,\mathrm{B}$-parameter backbone rather than to the coupling. Both are far below a supervised SLU system; what they establish is that the capability is reachable from the recognition cache at all.

What does not survive. No layer of the language model attends to the acoustics, so nothing absent from the transcript is recoverable: speaker identity, emotion, prosody and non-speech audio are all out of reach.

4.4 Is the Frozen `lm_head` Necessary?

Training the same AuT encoder — unfrozen — into a randomly initialised classifier over Qwen3's own $151{,}669$ tokens, with no language model anywhere, is five to eight times worse at every data scale: $40.0\,\%$ dev-clean at $960\,\mathrm{h}$ against $5.2\,\%$ for the frozen head, and a collapse into memorisation at $1\,\mathrm{h}$. Appendix G gives the measurements and the training losses that explain the ordering.

Table 2: CoVoST-2 en→de, 150 utterances, BLEU. All rows share the frozen Llama-3.2-1B backbone and the same two-shot English:/German: prompt, and differ only in where the English text comes from. The first column adds nothing to that prompt; the second adds a LoRA trained on text pairs alone, which never sees audio and leaves the joint network untouched. The reference-transcript row is the backbone's own ceiling under each condition. Slimducer's translation continues from the recognition key/value cache and the transcript is never re-encoded; in the adapted column the single adapted instance carries recognition as well, so no cache crosses an adapter boundary.

English text comes from	**2-shot only**	**2-shot + LoRA** (3.4 M)
reference transcript (ceiling)	10.52	**31.83**
Slimducer (from the recognition cache)	**6.59**	**14.98**
cascade: wav2vec2-base-960h	3.94	12.09

4.5 Chunk-Streaming Feasibility

Slimducer's decode loop never looks past the frame it is on, and the AuT encoder's attention is already chunk-bounded — attention is computed per chunk and concatenated, never across chunk boundaries, in windows of at most n_window_infer = 800 frames (8.0 s). Feeding audio in chunks and decoding incrementally on 40 dev-clean utterances reproduces the offline hypothesis token for token at 8.0 s on all 40, costs 5.1 WER points at 1.0 s — the shortest the encoder structurally allows — and under 0.2 points by 4.0 s. Real-time factor stays at 0.06 to 0.07 throughout, so the attention window and not compute sets the latency/accuracy trade-off. Appendix H has the measurements.

4.6 Behaviour Outside the Training Domain

A model whose linguistic knowledge sits in a frozen backbone and trains 3.0 M parameters on one corpus should carry less of that corpus with it than one that fine-tunes its whole encoder on the same audio. On the Open ASR Leaderboard suite the reversal is clean and holds for both Slimducer configurations: wav2vec2-base-960h leads in domain on test-clean (by 13 % relative over WavLM+Llama, 22 % over AuT+Qwen3), and loses on all five sets outside it — by 13 % to 31 % and 18 % to 39 % respectively across VoxPopuli, GigaSpeech, Common Voice, Earnings-22 and AMI. 95 M parameters free to fit 960 h of read audiobooks fit them; 3.0 M have no capacity to. Appendix I has the per-set numbers, the controls and the scoring protocol.

4.7 Latency Against a Cascade

Because recognition is frame-synchronous, everything computed while the speaker is still talking is absorbed by the utterance. What remains after the last sample is one backbone forward for Slimducer — the transcript is already in the cache — against the recogniser plus a forward over the whole prompt for a cascade. On an idle GPU at batch one that saving is 0.4 ms and worth nothing, but it grows with load: 70.4 ms (80 % of the cascade's prefill) at batch 64, and an end-to-end tail 1.5 to 2.2 times shorter on four CPU threads, widening with utterance length. Appendix L has the sweep.

4.8 Streaming Accuracy at a Latency Budget

Streaming the 960 h checkpoint end to end — a WavLM tower trained bidirectionally and never fine-tuned to stream — Slimducer translates better than the cascade at every chunk size, and the gap widens as the budget tightens: 2.8 BLEU at 1.92 s and a factor of 3.9 at 0.32 s. Streaming at 1.92 s (12.75) is still ahead of the cascade run *offline* (11.03). Both lose to chunking — Slimducer from 15.28 to 12.75 — which is the price of chunking a bidirectional tower. Appendix M gives the windowing, the equivalence checks and the per-budget numbers.

AI Use Statement

Large language models were used only for spelling and grammar correction of the authors' own prose. They were not used to generate research ideas, to design or run experiments, to analyse

results, or to write any technical content. The authors take full responsibility for all content of this paper.

Ethics Statement

This work uses only publicly available speech corpora released for research use and involves no human subjects, no personally identifying information, and no sensitive attributes. We see no ethical concerns specific to this work.

Reproducibility Statement

Section 4.1 and Appendix J give the optimiser, schedule, batching and decoding protocol for every run, and the evaluation protocol, including which text normalisation applies to which set, is stated in Appendix F and with every table. The training and evaluation code, together with the configurations for the runs reported here, will be made available.

References

C. Kim, Minkyu Shin, Abhinav Garg, and Dhananjaya Gowda. Improved vocal tract length perturbation for a state-of-the-art end-to-end speech recognition system. In *INTERSPEECH-2019*, pp. 739–743, Graz, Austria, Sept. 2019. doi: 10.21437/Interspeech.2019-3227. URL http://dx.doi.org/10.21437/Interspeech.2019-3227.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12449–12460. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. SLURP: A spoken language understanding resource package. In *Proc. EMNLP*, 2020. URL https://arxiv.org/abs/2011.13205.

Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. AISHELL-1: An open-source Mandarin speech corpus and a speech recognition baseline. In *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*, pp. 1–5. IEEE, 2017.

C. Kim, A. Misra, K.K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani. Generation of simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home. In *INTERSPEECH-2017*, pp. 379–383, Aug. 2017.

Xie Chen, Zhong Meng, Sarangarajan Parthasarathy, and Jinyu Li. Factorized neural transducer for efficient language model adaptation. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8132–8136. IEEE, 2022. URL https://arxiv.org/abs/2110.01500.

Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778, April 2018. doi: 10.1109/ICASSP.2018.8462105.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen2-Audio technical report. *arXiv preprint arXiv:2407.10759*, 2024. URL https://arxiv.org/abs/2407.10759.

Alexis Conneau, Min Ma, Simran Khanuja, Yu Zhang, Vera Axelrod, Siddharth Dalmia, Jason Riesa, Clara Rivera, and Ankur Bapna. FLEURS: Few-shot learning evaluation of universal representations of speech. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pp. 798–805. IEEE, 2023.

Keqi Deng and Philip C. Woodland. Label-synchronous neural transducer for adaptable online E2E speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32: 3507–3516, 2024. arXiv preprint version: arXiv:2311.11353.

Keqi Deng, Jinxi Guo, Yingyi Ma, Niko Moritz, Philip C. Woodland, Ozlem Kalinli, and Mike Seltzer. Transducer-llama: Integrating LLMs into streamable transducer-based speech recognition. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025. URL https://arxiv.org/abs/2412.16464.

Fixie.ai. Ultravox: a multimodal speech LLM with a frozen Llama backbone. https://huggingface.co/fixie-ai/ultravox-v0_5-llama-3_1-8b, 2025.

Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, Rodrigo Cabrera, and Eugene Weinstein. RNN-transducer with stateless prediction network. In *ICASSP*, pp. 7049–7053, 2020.

Alex Graves. Sequence transduction with recurrent neural networks. In *International Conference of Machine Learning (ICML) 2012 Workshop on Representation Learning*, 2012a. URL "http://arxiv.org/abs/1211.3711".

Alex Graves. Sequence transduction with recurrent neural networks. In *International Conference of Machine Learning (ICML) 2012 Workshop on Representation Learning*, 2012b. URL http://arxiv.org/abs/1211.3711.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 369–376, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143891. URL http://doi.acm.org/10.1145/1143844.1143891.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proc. Interspeech 2020*, pp. 5036–5040, 2020. doi: 10.21437/Interspeech.2020-3015. URL http://dx.doi.org/10.21437/Interspeech.2020-3015.

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context. In *Proc. Interspeech 2020*, pp. 3610–3614, 2020. doi: 10.21437/Interspeech.2020-2059. URL http://dx.doi.org/10.21437/Interspeech.2020-2059.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021. URL https://arxiv.org/abs/2106.07447.

E. J. Hu et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proc. ACL*, 2019. URL https://arxiv.org/abs/1810.08398.

Ziyang Ma, Guanrou Yang, Yifan Yang, Zhifu Gao, Jiaming Wang, Zhihao Du, Fan Yu, Qian Chen, Siqi Zheng, Shiliang Zhang, and Xie Chen. An embarrassingly simple approach for LLM with strong ASR capacity. *arXiv preprint arXiv:2402.08846*, 2024. URL https://arxiv.org/abs/2402.08846.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022. URL https://arxiv.org/abs/2212.04356.

Paul K. Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, Hannah Muckenhirn, Dirk Padfield, James Qin, Danny Rozenberg, Tara Sainath, Johan Schalkwyk, Matt Sharifi, Michelle Tadmor Ramanovich, Marco Tagliasacchi, Alexandru Tudor, Mihajlo Velimirović, Damien Vincent, Jiahui Yu, Yongqiang Wang, Vicky Zayats, Neil Zeghidour, Yu Zhang, Zhishuai Zhang, Lukas Zilka, and Christian Frank. AudioPaLM: A large language model that can speak and listen. *arXiv preprint arXiv:2306.12925*, 2023. URL https://arxiv.org/abs/2306.12925.

Haşim Sak, Matt Shannon, Kanishka Rao, and Françoise Beaufays. Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping. In *Proc. Interspeech*, pp. 1298–1302, 2017.

Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching-Feng Yeh, Julian Chan, Frank Zhang, Duc Le, and Mike Seltzer. Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021. URL https://arxiv.org/abs/2010.10759.

Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. SALMONN: Towards generic hearing abilities for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.13289.

Yun Tang, Shanil Puri, Shinji Watanabe, and Subhabrata Mukherjee. TRADE: Transducer-augmented decoder for speech LLM. *arXiv preprint arXiv:2606.08486*, 2026. URL https://arxiv.org/abs/2606.08486.

Anshuman Tripathi, Han Lu, Haşim Sak, and Hagen Soltau. Monotonic recurrent neural network transducer and decoding strategies. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 944–948, 2019.

Ehsan Variani, David Rybach, Cyril Allauzen, and Michael Riley. Hybrid autoregressive transducer (HAT). In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6139–6143. IEEE, 2020. URL https://arxiv.org/abs/2003.07705.

Changhan Wang, Anne Wu, and Juan Pino. CoVoST 2 and massively multilingual speech translation. In *Proc. Interspeech*, 2021. URL https://arxiv.org/abs/2007.10310.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Zengwei Yao, Liyong Guo, Xiaoyu Yang, Wei Kang, Fangjun Kuang, Yifan Yang, Zengrui Jin, Long Lin, and Daniel Povey. Zipformer: A faster and better encoder for automatic speech recognition. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.11230.

Table 3: Where each system attaches an LLM to speech, and what it changes to do so. Slimducer alone changes neither the language model's weights nor the kind of input it is given, and alone predicts over the vocabulary the LLM was pre-trained with.

System	LLM weights	What the LLM is fed	Speech encoder	Output vocabulary
Qwen2-Audio (Chu et al., 2024)	fine-tuned	audio embeddings in context	trained	LLM's
SALMONN (Tang et al., 2024)	LoRA	audio embeddings in context	trained	LLM's
AudioPaLM (Rubenstein et al., 2023)	fine-tuned	audio embeddings in context	trained	LLM's, extended
Ultravox (Fixie.ai, 2025)	frozen	audio embeddings in context	fine-tuned	LLM's
SLAM-ASR (Ma et al., 2024)	frozen	audio embeddings in context	frozen	LLM's
Transducer-Llama (Deng et al., 2025)	transformer frozen, embed. + head retrained	token indices	trained	5,000, new
TRADE (Tang et al., 2026)	LoRA	token indices	top 6 layers	∼20,000, derived
Slimducer	**frozen**	**token indices only**	**frozen**	151,936**, unchanged**

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023. URL https://arxiv.org/abs/2305.11000.

Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7829–7833, 2020. doi: 10.1109/ICASSP40776.2020.9053896.

A Where Each System Attaches an LLM to Speech

Coupling at the transducer's predictor. Two systems attach where Slimducer does. Transducer-Llama (Deng et al., 2025) puts an LLM in the vocabulary branch of a factorized transducer and reaches 2.47 % / 6.53 % on `test-clean` / `test-other`; TRADE (Tang et al., 2026) runs a transducer branch alongside a multimodal LLM over a shared Conformer, reaching 3.13 % on `test-other`. Both shrink the output space to make the transducer grid affordable. Transducer-Llama freezes the LLM's layers but *retrains its embedding and output layers* over 5,000 ASR tokens instead of Llama3's 128,000. Initialising them from the LLM's is worth 8.6 % relative on `test-clean` by that paper's own measurement. TRADE uses LoRA, unfreezes the encoder's top six layers, and derives a 20,000-token vocabulary from the LLM's. Slimducer trains neither tower and neither embedding matrix: the 151,936-way `lm_head` is used exactly as pre-training left it, and the 3.0 M-parameter joint is all that learns. Its `test-other` lands next to Transducer-Llama's (0.064, 8 against 0.065, 3) under that constraint; TRADE, which trains the most, is ahead of both.

Keeping the vocabulary whole is affordable only because the classifier is not being learned — Qwen3's `lm_head` already is one. The same encoder feeding a randomly initialised layer over Qwen3's own 151,669 tokens reaches 40.0 % dev-clean after 960 h against 5.2 % for the frozen head (Sec. 4.4), which is why a system that must train its own output layer has to shrink what it predicts over.

Slimducer's emitted indices *are* LLM token indices, fed straight back into the language model, so an utterance ends with the key/value cache already holding it. Under a reduced vocabulary the transcript has to re-enter the LLM's token space first. Slimducer spends *more* LLM compute than a cascade, running the model L times sequentially where a prefill runs it once; what differs is when that compute happens, at 0.195 times real time.

B Another Language, and Further Results

B.1 Another Language

Nothing in the construction is English-specific: the joint learns an acoustic-to-token map, not anything lexical. The same 3.0 M-parameter joint on AISHELL-1's 170 h of Mandarin, both towers frozen as in every English run, reaches 6.54 % CER — inside the range a streaming Zipformer RNN-T reports on the same split, at a fraction of the parameters.

Table 4: Slimducer on AISHELL-1 (character error rate, no LM, full dev/test splits). Decoding differs across rows — greedy and no LM here, the cited systems' own beam/LM settings elsewhere — so this is a placement, not a controlled comparison. Slimducer's dev CER is $5.59\,\%$; both are final checkpoints with no dev-based selection.

Setting	**test CER**
AISHELL-1 baseline (GMM-HMM/DNN) Bu et al. (2017)	$\sim$10.5%
Zipformer streaming RNN-T Yao et al. (2024)	6.51–6.95%
Slimducer (3.0M trainable / 780.7M frozen, no LM)	**6.54**%

B.2 Further Results

Appendix C reports three more: the full $960\,\mathrm{h}$ set exercises under $19\,\%$ of Qwen3's $151{,}936$ tokens; replacing Llama-3.2-1B with Qwen3-4B moves $960\,\mathrm{h}$ accuracy $1.6\,\%$ to $2.3\,\%$ relative in the wrong direction; and on FLEURS Mandarin's $9.1\,\mathrm{h}$ Slimducer reaches $22.8\,\%$ CER against $69.9\,\%$ for a from-scratch CTC head on the same unfrozen encoder.

C Additional Experimental Results

C.1 Frozen Language Model and Vocabulary Coverage

The frozen output projection is Qwen3-0.6B's Yang et al. (2025) (`Qwen/Qwen3-0.6B`), with $C = 151{,}936$ entries. Only tokens that occur in the fine-tuning transcripts receive a gradient; the remaining rows keep whatever direction they had at initialization. We lower-cased and tokenized every transcript in the 10-hour, 100-hour and *train-all* sets with the Qwen3 tokenizer and counted distinct token IDs (Table 5).

Table 5: Coverage of the frozen Qwen3-0.6B vocabulary ($C = 151{,}936$ tokens) by the transcripts of each LibriSpeech-derived fine-tuning set. "Unique tokens" is the number of distinct Qwen3 token IDs observed at least once among the lower-cased transcripts.

Fine-tuning set	**Utterances**	**Total tokens**	**Unique tokens**	**Coverage (%)**
LibriLight 10 h	2,763	107,010	9,664	6.36
LibriSpeech `train-clean-100` (100 h)	28,539	1,087,490	21,005	13.82
LibriSpeech train-all ($\sim$960 h)	281,241	10,330,942	28,718	18.90

Coverage grows far more slowly than data. A $96\times$ increase in transcribed hours ($10\,\mathrm{h} \rightarrow 960\,\mathrm{h}$) exercises only $3\times$ more distinct tokens ($9{,}664 \rightarrow 28{,}718$) — Zipf's law within one read-speech domain. Even *train-all* touches under 19% of the vocabulary, so the usefulness of the remaining rows rests entirely on Qwen3's own text pre-training, which Slimducer leaves untouched.

C.2 A Larger Language Model Buys Nothing

If the frozen backbone supplied the accuracy, a larger one should raise it. We trained the same joint, on the same $960\,\mathrm{h}$ and schedule, against Qwen3-4B in place of Llama-3.2-1B, leaving the WavLM-Large tower, the $3.0\,\mathrm{M}$ trainable parameters and the decoding alone. The 4-billion backbone reaches $3.77\,\%$/$6.63\,\%$ against $3.71\,\%$/$6.48\,\%$ for the 1-billion one: $1.6\,\%$ to $2.3\,\%$ relative in the wrong direction from 3.3 times the parameters (4.02 against $1.24\,\mathrm{B}$). Both are single seed, so the reading is not that the smaller model wins but that language-model size is not the binding constraint. The error ratio agrees. At $960\,\mathrm{h}$ both WavLM runs sit at `test-other`/`test-clean` $= 1.75$ to 1.76 while AuT sits at 2.20, so the ratio moves with the encoder and a $3.3\times$ change in language model leaves it where it was.

C.3 What the Frozen Backbone Buys at Very Little Data

AISHELL-1's $170\,\mathrm{h}$ shows the coupling works on Mandarin, but not that the frozen backbone's Mandarin knowledge is what got it there — $170\,\mathrm{h}$ suffices for a from-scratch system regardless. FLEURS's Mandarin split (Conneau et al., 2023) removes the data: $9.1\,\mathrm{h}$ of training audio, an order of magnitude below AISHELL-1.

Two systems were fit to it, sharing the AuT tower and the 9.74 h/6,250-step recipe of Sec. 4.1 unchanged: Slimducer (AuT and the 0.6 B-parameter Qwen3 backbone both frozen, 3.0 M trainable) against a plain CTC head on an unfrozen AuT with a 32-symbol grapheme output fit from scratch and no language model at all. The from-scratch system does not reach usable accuracy at this scale — 69.9 % CER — while Slimducer reaches 22.8 % at the same step budget. Slimducer's 3.0 M parameters only have to learn where a token boundary falls, never what the token is; that remains the frozen backbone's own distribution.

Table 6: FLEURS Mandarin, 9.1 h train, test CER (944 utterances, the full split), same 6,250-step schedule. Slimducer trains 3.0 M parameters (both towers frozen); the CTC system trains AuT's full 184.7 M plus a from-scratch output layer, with no language model of any kind.

System	Trainable params	test CER
Slimducer (AuT + Qwen3-0.6B, both frozen)	3.0 M	**22.8**%
Plain CTC (AuT unfrozen, grapheme head)	184.7 M	69.9%

D Component Substitution and Matched-Data Comparison

D.1 Accuracy against a transducer at matched data

Table 7 places SlimDucer beside a conventional RNN-T trained on the same audio. The transducer is a wav2vec 2.0 base encoder with an LSTM predictor and a 320-dimensional joint, 91.5 M trainable parameters, decoding a 32-symbol character vocabulary; SlimDucer trains only its joint and blank head — 3.0 M parameters against 780.7 M frozen — and decodes the frozen backbone's own 151,936-token vocabulary. The comparison is of systems, not of parameter counts, and the two differ in what they are allowed to learn: the transducer fits an output layer to the corpus, SlimDucer inherits one.

The transducer is the more accurate system on `dev-clean` at both scales, by 35 % relative at 9.74 h and 17 % at 100 h. On `dev-other` the ordering reverses, by 16 % and 37 % relative. The last column states the effect compactly: reading noisy speech costs the transducer a factor of 2.0 to 2.9 in error rate and costs SlimDucer a factor of 1.25 to 1.59. This is the behaviour the coupling predicts. A transducer's predictor is a small network trained on the transcripts of the same corpus, so when the acoustic evidence degrades there is little to fall back on; SlimDucer's fallback is a language model trained on a corpus larger than any speech set, and it is present unchanged at every step of the beam.

The gap on `dev-clean` narrows as data is added — 35 % to 17 % relative from 9.74 h to 100 h — which is consistent with the joint network, not the backbone, being the component that is short of data. At 960 h SlimDucer reaches 0.045, 5/0.068, 7 while keeping the ratio at 1.51; the robustness does not decay as the clean number improves.

D.2 Substituting the backbone

The language model is never updated, so replacing it is a matter of re-fitting the joint network and nothing else: no vocabulary is rebuilt, no output layer is discarded, and the acoustic side is untouched. Table 8 does that across an order of magnitude in backbone size.

All three work, and the spread across them is smaller than the spread across training budgets in Table 18. It is also not monotone in backbone size: the 0.6 B model is the most accurate of the three on `test-clean` at both scales, the 1.2 B model is the most accurate on `test-other` at 100 h by 4 % relative while being indistinguishable from it on `test-clean` (0.2 %), and the 0.36 B model — the only one of the three whose tokeniser is not built for this vocabulary size — gives up 14.8 % relative at 9.74 h and 7.0 % at 100 h. Within this range, then, what the joint needs from the backbone is a usable next-token state and a well-conditioned unembedding, not parameter count. The method is not a property of one backbone, and the choice of backbone is a deployment decision rather than an accuracy decision — which is the practical form of the claim that the language model is left intact. A 4 B-parameter backbone was trained separately at 960 h and is reported in Table 1 and Sec. C; it does not improve on the 1 B one.

Table 7: Word error rate on `dev-clean` and `dev-other`. The RNN-T rows are seed means (five seeds at 9.74 h, four at 100 h); the SlimDucer rows are single runs at the budget Section O.1 selects. The last column is the ratio between the two error rates, which is the quantity the architecture is meant to move.

System	Train	dev-clean	dev-other	other/clean
RNN-T	9.74 h	**0.0941**	0.1883	2.00
SlimDucer	9.74 h	0.1271	**0.1590**	**1.25**
RNN-T	100 h	**0.0580**	0.1705	2.94
SlimDucer	100 h	0.0676	**0.1073**	**1.59**
SlimDucer	960 h	0.0455	0.0687	1.51

Table 8: Backbone substitution. The speech encoder, the joint network, the schedule and the seed are held fixed; only the frozen language model changes. Budgets are 6,250 steps at 9.74 h and 50,000 at 100 h. Greedy float32 decoding at batch size one on the full test splits ($n = 2{,}611/2{,}932$).

		9.74 h		100 h	
Frozen backbone	**Params**	**test-cl.**	**test-oth.**	**test-cl.**	**test-oth.**
SmolLM2-360M	0.36 B	0.1426	0.1948	0.0735	0.1387
Qwen3-0.6B	0.6 B	**0.1242**	**0.1815**	**0.0687**	0.1341
Llama-3.2-1B	1.2 B	0.1305	0.1846	0.0688	**0.1287**

Table 9: Encoder substitution at 9.74 h, 6,250 steps, Qwen3-0.6B frozen throughout. The wav2vec 2.0 row is the best of twenty configurations spanning output stride 1 to 16, joint dropout 0.1 to 0.5, staged encoder warmup, and masking disabled. Decoded greedily in float32 at batch size one on the full test splits ($n = 2{,}611/2{,}932$), the same protocol as Table 10, so the two may be read together.

Speech encoder	Rate	Params	test-clean	test-other
AuT (Qwen3-ASR)	13 Hz	185 M	**0.1242**	**0.1815**
Whisper-small, stride 4	12.5 Hz	88 M	0.1458	0.2122
wav2vec 2.0 base	12.5 Hz	95 M	0.9790	0.9798

D.3 Substituting the speech encoder

The other half of the coupling is less forgiving, and the honest statement of the result is that one encoder family works and another does not.

Whisper's encoder transfers: it is half the size of AuT and reaches 17 % relative worse on `test-clean`, the kind of gap one expects between two encoders of different capacity and pre-training data. wav2vec 2.0 does not transfer at all — it does not leave the plateau — and no configuration we tried recovered it.

Our first explanation was wrong, and we record it because the way it fails is what identifies the real cause. Both encoders that work are self-supervised or weakly-supervised and all three are given the same frame rate, so the frame rate is not the explanation. We conjectured instead that an encoder transfers only if it was pre-trained beneath a text-predicting decoder, which would leave wav2vec 2.0's representation carrying phonetic content in a geometry the frozen unembedding cannot read. That conjecture is tested and rejected immediately below. What the table establishes on its own is narrower and survives: the method is not tied to one encoder, and encoder choice matters far more than backbone choice.

Table 10: Encoder substitution across training size, evaluated on the full LibriSpeech test splits with greedy decoding, batch size 1, fp32, and $n = 2611/2932$ after the 30 s cap the encoders impose. The 9.74 h and 100 h rows hold Qwen3-0.6B fixed and change only the encoder. The 960 h rows change encoder *and* backbone, so the last line is not a controlled encoder swap; the 100 h backbone comparison puts Llama-3.2-1B at 1.9 % ahead of Qwen3-0.6B, which bounds how much of the 960 h gap that substitution can explain.

Data	Encoder + backbone	test-clean	test-other	other/clean
9.74 h	AuT + Qwen3-0.6B	**0.1242**	**0.1815**	1.46
	WavLM-Large + Qwen3-0.6B	0.1703	0.2001	1.17
	Whisper-small + Qwen3-0.6B	0.1458	0.2122	1.46
100 h	AuT + Qwen3-0.6B	**0.0687**	0.1341	1.95
	WavLM-Large + Qwen3-0.6B	0.0761	**0.1184**	1.56
960 h	AuT + Qwen3-0.6B, 375k	0.0434	0.0948	2.18
	AuT + Qwen3-0.6B, 500k	0.0425	0.0937	2.20
	WavLM-Large + Llama-3.2-1B, 375k	**0.0371**	**0.0648**	1.75

The text-decoder conjecture is false. The conjecture above – that an encoder transfers only if it was pre-trained beneath a text-predicting decoder – does not survive a direct test. WavLM-Large is self-supervised with no decoder of any kind above it, and it transfers: at 100 h it beats AuT on `test-other` by 11.7 % relative. A second conjecture we briefly held, that pre-training volume orders the encoders, does not survive either – Whisper-small sees 680,000 hours against WavLM's 94,000 hours and loses to it.

What separates wav2vec 2.0 from the other three is mechanical. `facebook/wav2vec2-base` sets `feat_extract_norm="group"`, so the group normalization inside the convolutional feature extractor takes its statistics over the whole time axis. Zero padding enters those statistics and changes the outputs at the *valid* frames; the attention mask acts only inside the transformer and cannot undo it. Under dynamic batching each utterance is padded differently at every step, so the joint network never sees a stable representation to fit. WavLM-Large sets `feat_extract_norm="layer"`, which normalizes per position and is immune; AuT and Whisper have no convolutional front end of this kind at all. The three observations – `group` fails, `layer` converges, no front end converges – follow from one cause.

The same mechanism is measurable at decode time. On a matched checkpoint, decoding at batch 8 rather than batch 1 – padding being the only difference – moves WER by 5.7 % relative for a `group`-normalized CTC model and 27.7 % at batch 32 for the corresponding transducer. All numbers in this paper are therefore decoded at batch 1, where the padding is zero bytes wide and the score does not depend on which utterances happen to share a batch. The encoders in Table 10 are unaffected: WavLM-Large normalizes per position, and AuT and Whisper pad every utterance to a fixed 30 s window regardless of batch.

Where the gain sits. Reporting a single geometric mean hides what the encoder swap does. Table 10 gives both splits at three training sizes. AuT is ahead on `test-clean` until the data is large; WavLM is ahead on `test-other` from 100 h onward, and the margin widens. At 100 h the two move in opposite directions by roughly equal amounts – `test-clean` 10.8 %, `test-other` −11.7 % – so their geometric mean, −1.1 %, reports almost nothing. The crossover often described as one encoder overtaking another is instead two splits improving at different rates: WavLM's `test-clean` deficit falls monotonically with data while its `test-other` deficit turns into an advantage.

At 960 h and a matched 375,000-step budget the substitution gives −14.6 % on `test-clean` and −31.6 % on `test-other`, a geometric mean of 4.90 % against 6.41 %. Against the best AuT run we have at any budget (500,000 steps) the margins are −12.8 % and −30.8 %. The `other/clean` ratio falls from 2.18–2.20 to 1.75: whatever WavLM supplies, it supplies most of it on the harder split.

Two caveats attach to the $960\,\mathrm{h}$ line. It changes two things at once, as the caption says. And it is a single seed – the $9.74\,\mathrm{h}$ rows are four seeds, the rest are one, and we have not measured run-to-run spread at this scale.

E SLIMDUCER AS A MONOTONIC TRANSDUCER

Slimducer is a monotonic transducer Sak et al. (2017); Tripathi et al. (2019) whose prediction network is a frozen language model and whose joint output layer is that model's frozen unembedding. Three differences follow. The prediction network never sees the paired transcripts, so it cannot memorize them. The label distribution is the backbone's own, not an approximation learned from speech transcripts. And the alignment is chosen rather than summed over, which removes the joint tensor dominating transducer memory — a tensor linear in $|\mathcal{V}|$ that a transducer must keep for its backward pass, and that Slimducer never retains at all. The second point meets a line of work arguing that a transducer's prediction network does not behave as a language model (Ghodsi et al., 2020) and architectures that restructure it so that it does (Variani et al., 2020; Deng & Woodland, 2024); Slimducer instead supplies a language model that was never asked to do anything else.

Structurally Slimducer matches a transducer, except that the label index is no longer a free axis. An alignment is a map $a : \{0..T{-}1\} \to \{\varnothing\} \cup \{0..L{-}1\}$ emitting each reference token exactly once and in order. It fixes how many tokens precede each frame,

$$n(t) = \big|\{\, t' < t \;:\; a(t') \neq \varnothing \,\}\big|, \tag{7}$$

so the joint is evaluated at $(t, l) = (t, n(t))$ — once per frame, along the path, rather than over a grid:

$$\mathbf{h}^{(\mathrm{slim})}_{t,l} = \tanh\Big(W^{(\mathrm{slim})}_{p}\mathbf{h}^{(\mathrm{pred})}_{l} + W^{(\mathrm{slim})}_{e}\mathbf{h}^{(\mathrm{enc})}_{t} + \mathbf{b}^{(\mathrm{slim})}\Big), \tag{8a}$$

$$\mathbf{h}^{(\mathrm{slim})}_{s} = W^{(\mathrm{logit})}\mathbf{h}^{(\mathrm{slim})}_{t,l} + \mathbf{b}^{(\mathrm{logit})}, \text{ and} \tag{8b}$$

$$\hat{\mathbf{z}}^{(\mathrm{slim})}_{s} = \mathrm{Softmax}\left(\mathbf{h}^{(\mathrm{slim})}_{s}\right). \tag{8c}$$

Two moves are available from state (t, l), and *both* advance t:

$$\text{blank:}\quad (t,l) \to (t{+}1,\, l), \qquad \text{score } \log \beta_{t,l}, \tag{9a}$$

$$\text{emit } c_l:\quad (t,l) \to (t{+}1,\, l{+}1), \qquad \text{score } \log(1-\beta_{t,l}) + \log \hat{z}_{t,l}(c_l), \tag{9b}$$

where $\beta_{t,l} = \sigma\Big(\mathbf{w}^{\top}\mathbf{h}^{(\mathrm{slim})}_{t,l} + b\Big)$ is the blank head and $\hat{z}_{t,l}$ is equation 8c. Because neither move leaves t where it was, a path visits exactly T states, a frame emits at most one token, and an alignment exists only when $T \geq L$. What Slimducer adds is the transducer's conditioning: $\mathbf{h}^{(\mathrm{pred})}_{l}$ depends on the tokens emitted so far.

The alignment is the Viterbi path under equation 9a–equation 9b, taken once per utterance under `no_grad`. Training reads it as a fixed target, so the loss is an ordinary cross-entropy over the T states on that path and nothing wider (Sec. 3.4).

F TEXT NORMALISATION

Text normalisation differs by evaluation set and is stated with every table. On LibriSpeech we apply *none*, the protocol every published number on these splits uses. Out of domain we apply the Whisper English normaliser to hypothesis and reference alike, as the Open ASR Leaderboard does: `wav2vec2-base-960h` predicts over 26 upper-case letters and an apostrophe, so scoring it unnormalised against punctuated references would measure its output vocabulary rather than its acoustics. The two are never mixed within a table and are not interchangeable — on the same checkpoint and audio they differ by up to $5.5\,\%$ relative.

G THE PLAIN-CTC ABLATION

Reusing Qwen3's frozen output projection saves gradients and memory, but a large enough labelled set could in principle learn the same mapping. We test this directly. The AuT encoder of Sec. 3.1 — here trained in full rather than frozen — feeds a randomly initialised linear layer over Qwen3-0.6B's own 151,669 tokens plus one CTC blank, trained end to end with CTC Graves et al. (2006). That width is the tokenizer's actual inventory; the frozen head Slimducer writes into is 151,936 wide because the embedding matrix is padded, and the extra 267 rows correspond to no token. There is no joint network and no language model; the condition shares only the encoder and the vocabulary with Slimducer.

The classifier improves steadily with data — 89.9%, 71.3%, 35.3% dev-clean at 1 h, 10 h and 100 h — but does not catch up: 35.3% at 100 h against Slimducer's 13.6 to 13.9% at 10 h. At 1 h it collapses outright, reaching 0.004 training loss while held-out WER stays at 89.9%/93.5%. That is memorization: with a few hundred utterances and 151,670 output classes carrying no prior structure the classifier can fit the training set exactly, because nothing in a random initialization relates one token's row to another's the way Qwen3's pre-trained geometry does.

The 960 h run reaches 40.0%/50.6% after 500,000 steps and 4.5 epochs — *worse* than 100 h. The training losses explain the ordering: 0.004, 0.29, 1.36, 1.86 at 1 h, 10 h, 100 h and 960 h. The 1 h condition fits its training set outright, the 100 h condition at 1.7 epochs still partly does, and only the 960 h condition cannot; its loss is flat over the final 10,000 steps, so that row records underfitting, not overfitting. At every scale the from-scratch classifier is five to eight times worse than the same encoder writing into a frozen `lm_head` — 5.2× at 10 h, 4.8× at 100 h and 7.7× at 960 h, all on dev-clean.

Table 11: Plain CTC over the same 151,669-token Qwen3 vocabulary as Slimducer, but with a classification head trained from random initialization rather than reused frozen. dev-clean/dev-other WER on LibriSpeech, against Slimducer's own numbers on the same data (Sec. 4).

Fine-tuning set	**dev-clean WER**	**dev-other WER**
Plain CTC, random-init classifier (this ablation)		
LibriLight 1 h	89.9%	93.5%
LibriLight 10 h	71.3%	79.4%
LibriSpeech `train-clean-100` (100 h)	35.3%	48.3%
LibriSpeech train-all (960 h)‡	40.0%	50.6%
Slimducer, frozen `lm_head` (Sec. 4)		
LibriLight 10 h	13.6–13.9%	17.7–18.0%
LibriSpeech `train-clean-100`†	7.3%	11.0%

† trained under this profile's prior schedule (5,000/70,000/25,000 rather than the 1,000/79,000/20,000 warmup/stable/decay used elsewhere). ‡ 500,000 steps, 4.5 epochs, decay carried to $\mathrm{lr} = 0$. Not on the same part of the curve as the 100 h row — see the text.

H CHUNK STREAMING

Slimducer's decoding loop never looks past the frame it is on: `decode` is frame-synchronous, and the Qwen3 prediction network advances only on emission — L times, not T — keeping its own key/value cache. The only non-incremental piece is the AuT encoder call, and whether that is a limit depends on the encoder alone. It is not: `Qwen3ASRAudioAttention.forward` computes attention separately per chunk and concatenates, never across chunk boundaries. The base chunk is `n_window` $\times 2 = 100$ mel frames (1.0 s), and consecutive base chunks are grouped into windows of up to `n_window_infer` $= 800$ frames (8.0 s).

We fed AuT audio in chunks of 1, 2, 4 and 8 of those 1.0 s units, ran the existing decode loop incrementally on each chunk's new frames, and compared against offline whole-utterance `decode` on the same 40 dev-clean utterances (10 h checkpoint). At 8.0 s, matching `n_window_infer`, streaming reproduces the offline hypothesis token for token on all 40 — a correctness check, and independent confirmation that 8.0 s is the encoder's actual receptive field rather than a configured default. Below it the error degrades gracefully: 1.0 s chunks, the shortest the encoder structurally

allows, cost 5.1 WER points against offline, and by $4.0\,\mathrm{s}$ the gap is under 0.2. Compute is not the binding constraint — mean real-time factor stays at 0.06 to 0.07 and the worst observed chunk ($1.0\,\mathrm{s}$, RTF 0.81) still finishes inside its own duration — the attention window is. A live system can run near $1\,\mathrm{s}$ of algorithmic latency, or trade up to $4\,\mathrm{s}$ to $8\,\mathrm{s}$ for offline-equivalent accuracy, by choosing how much audio to hand the encoder per step.

Table 12: Chunk-streaming AuT against offline, whole-utterance decoding: dev- clean WER, exact agreement with the offline hypothesis, per-chunk wall time, and the per-chunk real-time factor (processing time divided by the chunk's own audio duration; below 1.0 means the chunk finishes before the next chunk of live audio would arrive).

Chunk	dev-clean WER	Exact match	Wall/chunk	RTF (mean/max)
1.0 s	20.1%	3/40	55.6 ms	0.063 / 0.814
2.0 s	16.3%	5/40	105.5 ms	0.065 / 0.847
4.0 s	14.8%	12/40	175.3 ms	0.057 / 0.408
8.0 s	**14.96%**	**40/40**	337.8 ms	0.057 / 0.088
offline (whole utterance)	14.96%	–	–	–

I Behaviour Outside the Training Domain

A model that keeps its linguistic knowledge in a frozen backbone and trains $3.0\,\mathrm{M}$ parameters on one corpus should carry less of that corpus with it than one that fine-tunes its whole encoder on the same audio. We test this on the Open ASR Leaderboard suite, decoded under the protocol of Sec. 4.1 with the Whisper English normaliser applied to hypothesis and reference alike.

The systems start level in domain. On the same $2{,}611$ utterances of `test-clean`, `wav2vec2-base-960h` reaches $0.033, 9$ against $0.038, 9$ for WavLM+Llama and $0.043, 5$ for AuT+Qwen3 — the closest of the wav2vec 2.0 family, which is why we use it. `wav2vec2-large-960h-lv60-self` is at $0.018, 3$, far enough ahead that comparing against it would confound a domain effect with an in-domain one. Whisper-small is a ceiling, not a control: trained on $680{,}000$ hours spanning these domains.

Table 13 shows the reversal, and it holds for both Slimducer configurations. In domain, wav2vec 2.0 leads on `test-clean` by $13\,\%$ relative over WavLM+Llama and $22\,\%$ over AuT+Qwen3, and the two straddle it on `test-other` ($0.068, 4$ and $0.093, 7$ against $0.085, 0$). Outside the domain both are ahead on all five sets: by $13\,\%$ to $31\,\%$ for WavLM+Llama and $18\,\%$ to $39\,\%$ for AuT+Qwen3. $95\,\mathrm{M}$ parameters free to fit $960\,\mathrm{h}$ of read audiobooks fit them; $3.0\,\mathrm{M}$ have no capacity to, and the knowledge they decode with was never conditioned on that corpus.

J Optimiser and Schedule

AdamW at PyTorch defaults ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$), weight decay 0.01, gradients clipped to global norm 1.0, joint learning rate 5×10^{-4}. A second group at 2×10^{-5} exists for the audio tower and is unused here, the tower being frozen. The schedule is warmup–stable–decay, linear in each phase, decaying to zero. Steps are $1{,}000/4{,}000/1{,}250$ and $1{,}000/19{,}000/5{,}000$ for the two $9.74\,\mathrm{h}$ settings ($6{,}250$ and $25{,}000$ steps), $1{,}000/79{,}000/20{,}000$ at $100\,\mathrm{h}$, and $1{,}000/399{,}000/100{,}000$ at $960\,\mathrm{h}$. The first $1{,}000$ steps of every run train against uniform rather than model-derived targets (Sec. 3.4). Batches are formed by duration, not utterance count: $48\,\mathrm{s}$ of audio per step with no gradient accumulation, $4{,}800$ mel frames at the $100\,\mathrm{Hz}$ rate used throughout. At $960\,\mathrm{h}$ that is $81{,}831$ steps per epoch, so $500{,}000$ steps is 6.1 epochs.

K The Translation Adapter

The adapter is trained with AdamW at a peak learning rate of 2×10^{-4} under a one-cycle schedule with $5\,\%$ warmup, batch size 16, and a single pass over CoVoST-2's `en`→`de` training text. It has rank 16 and $\alpha = 32$ with dropout 0.05 on the attention projections `q_proj`, `k_proj`, `v_proj` and `o_proj`; pairs are truncated to 192 tokens and the prompt is masked out of the loss. Nothing

Table 13: Word error rate outside the training domain. The Whisper English normaliser is applied to hypothesis and reference alike (Sec. 4.1), so these numbers are not on the scale of Table 1. All rows but the first two are domains LibriSpeech does not cover. Slimducer and Whisper cap audio at 30 s, so their n is smaller on splits with longer utterances; both counts are given. `common_voice` and `earnings22` were resampled to 16 kHz, `earnings22` also downmixed to mono and not an official split.

	Slimducer, 3.0 M trained		**Fully fine-tuned**		n	
Evaluation set	**AuT+Qwen3**	**WavLM+Llama**	**wav2vec2-base**	**whisper-small**†	**Slim./Whis.**	**w2v2**
Training domain						
`librispeech test-clean`	0.0435	0.0389	**0.0339**	0.0349	2,611	2,620
`librispeech test-other`	0.0937	**0.0684**	0.0850	0.0760	2,932	2,939
Outside it						
`voxpopuli`	**0.1751**	0.1863	0.2133	*0.0871*	1,830	1,842
`gigaspeech`	**0.1576**	0.1910	0.2422	*0.1116*	19,898	19,898
`common_voice`	**0.2430**	0.2734	0.3954	*0.1564*	16,329	16,334
`earnings22`	**0.2574**	0.3144	0.4247	*0.1320*	2,727	2,737
`ami`	**0.2632**	0.3380	0.4025	*0.2036*	11,653	11,653

† ceiling, not a control: trained on 680,000 hours across these domains rather than on LibriSpeech.

in the speech path moves: the joint network, the blank head, the audio tower and `lm_head` are the weights the 960 h recognition checkpoint ended with.

One adapter, used throughout. A key/value cache belongs to the weights that produced it. Recognition writes the cache and translation continues from it, so both must run through the *same* adapted backbone — and this does not depend on which projections the adapter touches. Adapting any module changes the *input* to every `k_proj` and `v_proj` above it, so keys and values written by the base model are not the ones the adapted model would have written, at any layer. Recognising with one instance and generating with another leaves the generator reading a cache its own weights never produced. Measured, that costs 2.71 BLEU — 12.27 against 14.98 — which is enough to invert the ordering this table reports. It also manufactures a misleading consolation: under the same mistake, re-encoding the transcript as text scores 16.03, which makes re-encoding look necessary when it is only repairing the boundary the second instance introduced. With one instance throughout, reading the cache (14.98) and re-encoding the transcript (15.03) agree to 0.05 BLEU. The cache carries what the text carries, so the transcript need never be re-encoded.

L Latency Against a Cascade

Work done while the speaker is still talking is absorbed by the utterance and is not latency; emission is frame-synchronous, so what a listener waits through is only what remains after the last sample. For Slimducer that is the final encoder window and *one* backbone forward carrying the last token with `\nGerman:`, the transcript being already in the cache. For the cascade it is the recogniser over the whole utterance followed by a forward over the entire prompt, none of which can start earlier.

What that saving is worth depends on the compute regime, and is easy to measure wrongly. On an *idle* GPU at batch 1 a 1 B-parameter forward is bound by kernel launch, not arithmetic — one token already costs 10.9 ms — so 17 prompt tokens cost what 3 cost and the saving is 0.4 ms. Table 14 sweeps out of that one regime in both directions. Under concurrency the arithmetic is paid: the cascade's prefill grows from 11.1 ms at batch 1 to 88.5 ms at batch 64 while Slimducer's suffix forward stays at 11 ms to 22 ms, a saving of 16.0 ms (58 %) at batch 16 and 70.4 ms (80 %) at 64. On four CPU threads — one request, no spare capacity — the cascade pays 224 ms to read a 17-token transcript and 412 ms for a 64-token one, against 175 ms for Slimducer whatever was said. End to end the tail is 278 ms against 414 ms and 602 ms: 1.5 to 2.2 times, widening with utterance length because the quantity Slimducer removes is the one that scales. This assumes a chunk-causal encoder; retrofitting a bidirectional tower by re-encoding its left context consumes the saving.

Table 14: Tail latency: what remains after the last audio sample, so work that overlapped the speech is excluded. 5.63 s of audio, a 17-token transcript, medians. **LM side** is the quantity the coupling controls — a whole-prompt prefill for the cascade against a suffix-only forward for Slimducer. The batch-1 idle GPU row is the only regime in which reusing the cache saves nothing, and it is the regime a naive measurement lands in.

	LM side			
Regime	**Slimducer suffix**	**cascade prompt**	**saved**	**saved %**
GPU, batch 1, idle	11.2 ms	11.1 ms	−0.1 ms	—
GPU, batch 4	11.0 ms	12.8 ms	1.7 ms	13.6%
GPU, batch 8	13.2 ms	17.2 ms	4.1 ms	23.7%
GPU, batch 16	11.6 ms	27.5 ms	16.0 ms	58.0%
GPU, batch 32	12.4 ms	47.3 ms	34.9 ms	73.7%
GPU, batch 64	18.2 ms	88.5 ms	70.4 ms	79.5%
on-device, CPU ×4, 17-token transcript	175 ms	224 ms	49 ms	21.9%
on-device, CPU ×4, 64-token transcript	175 ms	412 ms	237 ms	57.5%

M Streaming Accuracy at a Latency Budget

Section 4.7 isolates the language-model side. This is the end-to-end measurement behind it, and it also settles what chunked streaming costs the 960 h checkpoint, whose WavLM tower was trained bidirectionally and never fine-tuned to stream.

We stream it by handing the tower a bounded left context plus the new chunk and keeping only the new chunk's frames. With the conv front end's stride 320 and receptive field 400 a window starting on the frame grid maps onto stream frames exactly, and because the kept frames sit at the window's end nothing can reach audio that has not arrived — verified bit-for-bit against the offline encoder. The cascade's recogniser is streamed identically.

The frame loop is not the constraint — median real-time factor 0.067 at a 0.32 s chunk, worst chunk 24 % of its own duration — and bounding the left context is free, with 3.2 s scoring the same as 8.0 s and as unbounded. What the chunk size buys is lookahead: the damage is missing *right* context, not missing left. At every chunk size Slimducer translates better than the cascade and the gap widens as the budget tightens — 2.8 BLEU at 1.92 s, a factor of 3.9 at 0.32 s — with streaming Slimducer at 1.92 s (12.75) ahead of the cascade run *offline* (11.03). The mechanism is the one the offline transcripts already showed: as the chunk shrinks, wav2vec 2.0's letter-level output degrades into non-words the backbone cannot use, while Slimducer's errors stay inside the backbone's vocabulary. Both systems lose to chunking — Slimducer from 15.28 to 12.75 BLEU — which is the price of chunking a bidirectionally trained tower; an encoder trained to stream would narrow it and remove the re-encoding noted in Section 4.7.

Table 15: CoVoST-2 `en→de`, 60 utterances (median 5.63 s), one job per GPU. Both systems stream through the same chunking with a 3.2 s left context and generate from the same adapted backbone. **RTF** is the streaming recogniser's processing time over the audio's own duration, median and worst single chunk; below 1 it keeps up with live audio. WER is of the English each system recognises.

	Slimducer		**cascade: `wav2vec2-base-960h`**		
Chunk	**WER**	**BLEU**	**WER**	**BLEU**	**RTF (med/max)**
0.32 s	53.7%	**6.65**	75.8%	1.69	0.067 / 0.240
0.64 s	46.6%	**9.85**	64.3%	5.37	0.127 / 0.306
0.96 s	41.3%	**10.83**	58.1%	5.24	0.097 / 0.199
1.92 s	31.0%	**12.75**	48.8%	9.91	0.073 / 0.128
offline (whole utterance)	27.0%	**15.28**	39.2%	11.03	—

Table 16: Forced alignment against marginalisation, LibriLight $9.74\,\mathrm{h}$, $6{,}250$ steps, four seeds each, identical in every other respect. dev-clean WER, seconds per step and peak memory are all measured on the same card. The memory column is a recompute-versus-retain choice, not a property of the two objectives: both score the same $(T, L{+}1)$ grid, but the marginal must differentiate through it, so its chunk scores are wrapped in gradient checkpointing and the chunk $\times\ (L{+}1) \times |\mathcal{V}|$ log-softmax is rebuilt during the backward pass instead of being kept. That is what buys the lower peak, and what costs the extra pass through the joint and `lm_head`. The forced path needs neither: its grid is traversed once under `no_grad` and only the T cells on the chosen path enter the backward graph.

Alignment	**dev-clean WER**	**s / step**	**peak GiB**
Forced (Viterbi), ordinary CE	$\mathbf{12.71 \pm 0.22}\%$	$\mathbf{0.345}$	9.10
Marginal (forward–backward)	$14.58 \pm 0.39\%$	1.05	$\mathbf{2.97}$

N Forced Alignment against Marginalisation

The marginal is available as an option — the forward variable can be accumulated with `logaddexp` so that autograd differentiates the recursion directly, which avoids writing the backward by hand but not the cost of *differentiating* through the whole $T \times (L{+}1)$ grid: the grid is scored either way, but the marginal must carry it into the backward pass, and recomputing it there under gradient checkpointing is what makes the step three times slower (Table 16). We measured both on the same four seeds (Table 16). Marginalising is three times slower per step and reaches a worse error rate, so the hard alignment is the default and the reason the loss stays an ordinary cross-entropy.

O Ablations

O.1 The alignment schedule

Line 2 of Algorithm 2 decides what the first alignment is computed from, and the training loss falls either way, so a bad choice is silent. Aligning against the initialized model is not arbitrary — Viterbi maximizes rather than samples — and across three seeds it gave 0.336, 0.276 and 0.958, the last never leaving its plateau. Spacing the L tokens evenly over the T frames removes the plateau and, over five seeds, beats all three in a quarter of the epochs. Given that start, refreshing the alignment every minibatch is both the most accurate cadence and the fastest. Appendix O.1 gives the measurements, the windowed cadence for longer refresh intervals, and the two diagnostics that make a collapse visible.

Round zero (line 2 of Algorithm 2) decides what the first alignment is computed from. The training loss falls either way, so a bad choice is silent.

Aligning against the initialized model is not arbitrary: Viterbi maximizes rather than samples, so the path reflects whatever the random joint prefers and the next epoch confirms it. Three seeds differing in nothing else gave 0.336, 0.276 and 0.958, the last never leaving the plateau. Spacing the L tokens evenly over the T frames removes it, and over five seeds beats all three in a quarter of the epochs; it also starts where healthy runs converge, at emission spread $1/\sqrt{12} \approx 0.289$ against 0.283 to 0.285.

From that start, refreshing every minibatch is both the most accurate cadence and the fastest (Table 17): the ordering is monotone in freshness, and aligning a batch immediately before fitting it shares one frozen-tower encoding, $71.6\,\mathrm{ms}$ against $86.3\,\mathrm{ms}$ per utterance-visit. From an initialized round zero the same cadence collapses — token loss 0.42, held-out error rising 0.976 to 1.158, emission 0.31 tokens per reference token. Appendix O.1 has the windowed cadence and the two diagnostics that make either collapse visible.

The windowed cadence remains, for corpora that need it. It is what makes any cadence longer than a batch affordable. Aligning the *whole corpus* every W steps costs (steps per epoch)$/W$ passes, which becomes ruinous as W shrinks; aligning only the minibatches about to be consumed costs one alignment per utterance-visit for *any* W, the same as per-batch and the same as one pass per epoch. Drawing the window from the shuffled stream before training on it is what makes that possible. It matters because staleness otherwise scales with the corpus: one epoch is 833 optimiser steps at $10\,\mathrm{h}$

Table 17: Refresh cadence on the 10 h set, 30 epochs, round zero uniform, otherwise identical. Every cadence performs one alignment per utterance-visit, so the three cost the same in alignment work; per-batch is faster only because it shares one encoding of the frozen towers between the alignment and the loss.

Refresh cadence	dev-clean	dev-other	s/step
every batch	**0.1373**	**0.1698**	**0.32**
rolling, 500 steps	0.1506	0.1900	0.51
rolling, 1000 steps	0.1640	0.2026	0.47
rolling, 2000 steps	0.2340	0.2783	0.45

Table 18: Step budget on the 10 h set at a fixed 20% decay fraction, round zero uniform and per-batch refresh throughout. Warmup is 1,000 steps in every row, so the rows differ only in how long the rate is held flat before it is annealed. One epoch is 833 steps.

Steps	Epochs	Decay from	dev-clean	dev-other
6,250	**7.5**	epoch 6.0	**0.1271**	**0.1590**
12,500	15	epoch 12.0	0.1290	0.1637
25,000	30	epoch 24.0	0.1378	0.1698
50,000	60	epoch 48.0	stopped at epoch 9; identical to the above until then	

but 8,577 at $100\,\mathrm{h}$ and 81,831 at $960\,\mathrm{h}$, and with W set to one epoch $100\,\mathrm{h}$ sits at 0.963 after that epoch where $W = 833$ on the same data and code reaches 0.126, and 0.073 at the end.

Two diagnostics make the failures visible. Neither collapse can be read from the loss, so both of the following are logged at every refresh. The *emission spread*, the standard deviation of emission positions divided by T, has the known reference above: healthy runs sit at 0.283–0.285 against the uniform 0.289, and a collapsing run shrinks toward 0.253 as the emissions bunch. The *agreement* between consecutive alignments should climb and then hold below saturation — healthy runs sit at 96–99% — whereas a run that has settled into a degenerate assignment freezes at 99.5% on the wrong fixed point. High agreement is evidence of convergence only when read together with the spread.

Table 17 is the ablation the method section argues for. All rows share the data, the code, the step budget and the seed; they differ only in when the frame targets are recomputed. The failures are silent in the training loss, which is why the table reports it alongside the error rate.

Table 17 is the ordering *after* round zero is fixed. Before it, with the initialized model driving the first alignment, the same per-batch cadence diverges — held-out error climbing from 0.976 to 1.158 while the training token loss falls to 0.42 — and a windowed cadence is what rescues it. That is the confound worth stating plainly: a cadence ablation run on top of a bad initialiser measures the initialiser.

All four rows of Table 18 share a seed, so their trajectories are identical up to the point where the decay begins: 0.9672 at epoch one, 0.1393 by epoch five, 0.1554 at epoch six. Only the anneal differs, and quadrupling the budget costs 8% relative error. The comparison is therefore within-seed; run-to-run spread is not measured here.

The optimum moves with the data, and more slowly than the data. The hard assignment reaches its fixed point early, and the schedule should be built around that rather than around a step count carried over from another model. On $10\,\mathrm{h}$ the error is at 0.139 by the fifth epoch and does not improve again until the learning rate decays: epochs 6 to 23 of a thirty-epoch run wander between 0.142 and 0.157 without trend.

Those epochs are not merely wasted. Table 18 quadruples the budget at a fixed 20% decay fraction and the error gets steadily *worse*, because a longer stable phase enters the decay from a worse point than a shorter one leaves it. Annealing immediately after the alignment settles is what produces the best result, at a quarter of the compute.

Repeating the sweep at the larger scales gives $0.071,7$ at $25{,}000$ steps, $0.067,6$ at $50{,}000$ and $0.072,9$ at $100{,}000$ on the $100\,\mathrm{h}$ set, and on $960\,\mathrm{h}$ a run branched at $250{,}000$ steps and annealed over a further $50{,}000$ reaches $0.045,5$/$0.068,7$ — better than the same run carried to $434{,}000$ steps on its original schedule, which stands at $0.051,9$/$0.080,1$. The best budget is thus $6{,}250$ steps at $9.74\,\mathrm{h}$, $50{,}000$ at $100\,\mathrm{h}$ and roughly $300{,}000$ at $960\,\mathrm{h}$: a tenfold increase in data buys an eightfold increase in useful steps at the first step and a sixfold increase at the second, so the number of epochs falls as the corpus grows — 7.5, 5.8 and 3.7 respectively (one epoch is 833, $8{,}576$ and $81{,}831$ steps at the three scales). The curve is flat enough on either side of the optimum that this is a guideline rather than a tuned quantity, but it is flat in only one direction: overshooting costs little, and the runs that were stopped early were still improving.

Branch scale. The first two ablations are motivated by a measurement rather than by symmetry. On a trained checkpoint without per-branch normalization, the language-model state arrives with an L_2 norm of 99.0 against the acoustic state's 17.2; training partially compensated by growing W_{aud} beyond W_{lm}, but the language-model term still entered the sum 3.15 times larger. Since the nonlinearity is close to linear at that magnitude, the larger term sets the output and the smaller reads as a perturbation. Equation equation 1 therefore normalizes each branch.