TRAINING IS SERVING: UNIFYING INFERENCE AND ADAPTATION FOR MULTIMODAL EXTENSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Extending large language models (LLMs) to multiple modalities is a key step toward general-purpose multimodal intelligence. As foundation models scale, conventional fine-tuning increasingly assumes homogeneous clusters and pays model-scale communication across ranks. We present Training is Serving (**TiS**), an API-driven compute-sharing framework that decouples the heavy computation of a frozen LLM from lightweight multimodal adaptation. Specifically, we enable the LLM to support training and return gradients in the form of an API. An edge client trains only its modality encoder and adapter, sends embeddings to the service, and receives gradients; raw data and trainable parameters remain at the edge, while the LLM weights remain at the cloud server. TiS decouples edge-side data parallel training from provider-side model parallel serving, which has four core advantages: (1) compute sharing, where edge clients can perform training by accessing the API without acquiring server control; (2) hardware decoupling, where heterogeneous accelerators are unified behind a standard API backend; (3) low-bandwidth communication, enabling multi-node MLLM training over only 1 Gbps links; and (4) data locality, where raw training data remains on the edge while the service receives only embeddings and labels. Using shared compute, we extend Gemma-4-31B-it to the speech modality and support 70 languages.

1 INTRODUCTION

Extending large language models (LLMs) to additional modalities is a central step toward general-purpose multimodal intelligence. However, as foundation models grow to hundreds of billions of parameters, conventional adaptation remains tied to a homogeneous high-performance cluster. Large-scale models are typically sharded with tensor parallelism (TP) (Shoeybi et al., 2019), which requires model shards to be co-located, coordinated by a common collective library, and connected by a high-bandwidth fabric. These assumptions are difficult to satisfy when compute is distributed across teams and clusters.

Enterprise environments expose a different bottleneck. Accelerators are often idle on independently managed machines, but their owners do not want to grant SSH access, administrative control. The machines may use different accelerator classes and collective libraries, such as A100 GPUs with NCCL, and 910C NPUs with HCCL, and they may be connected only through 1 Gbps LAN links. The problem is therefore not simply insufficient aggregate compute; it is the fragmentation of ownership and communication interfaces. Existing mechanisms address only parts of this setting. Cluster schedulers such as Ray (Moritz et al., 2018) assume unified administrative control and resource ownership. Inference APIs provide LLM generation but not model training.

We present Training is Serving (TiS), an API-driven compute-sharing framework. TiS applies split learning (Vepakomma et al., 2018) at the boundary of a frozen cloud-side LLM, while a lightweight modality encoder and adapter train at the edge. The edge sends embeddings and labels; the service returns the input-embedding gradient. Service owners expose idle capacity through a Docker worker without granting SSH access or server control, and clients route compatible requests to that endpoint. Thus, idle accelerators enter the LAN pool as stored credits for large-batch inference when demand arises. Participants may serve as edge clients or cloud providers; contribution earns credits and requests spend them.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

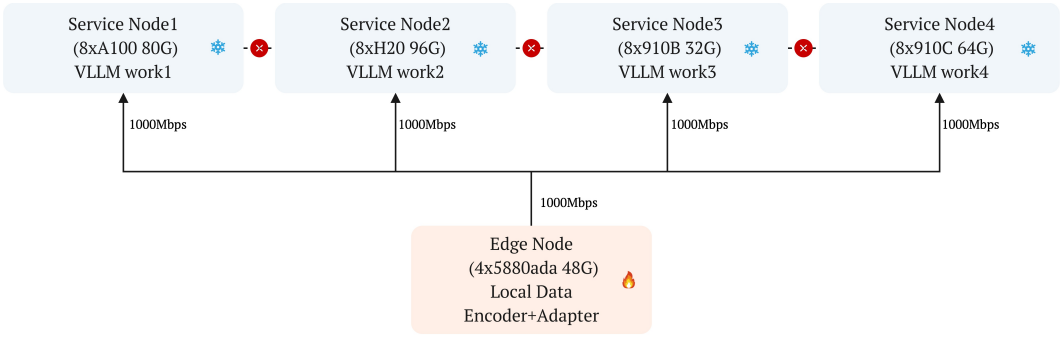

Figure 1: A hypothetical TiS deployment scenario. The edge node keeps raw data and trainable encoder and adapter parameters local. Four independent vLLM workers serve the same frozen checkpoint over 1 Gbps access links on four types of accelerators. Each worker exposes standard vLLM inference and a TiS gradient-return path. Red markers indicate that no communication between workers is required during training.

Our framework has four core advantages: (1) **compute sharing**, where edge clients can perform training by accessing the API without acquiring server control; (2) **hardware decoupling**, where heterogeneous accelerators are unified behind a standard API backend; (3) **low-bandwidth communication**, enabling multi-node MLLM training over only 1 Gbps links; and (4) **data locality**, where raw training data remains on the edge while the service receives only embeddings and labels.

Figure 1 illustrates a hypothetical deployment scenario. An edge node holds raw data and the trainable speech encoder and adapter parameters. Four independently managed vLLM (Kwon et al., 2023) workers, deployed on four types of accelerators, serve replicas of the same frozen LLM over 1 Gbps access links. They do not form one spanning collective and do not exchange per-layer training traffic. Each worker exposes two request paths: standard vLLM inference and a TiS gradient-return path. Only fixed-width embeddings, labels, and input-embedding gradients cross the service boundary. TiS therefore lets **researchers maintain only simple DDP code on the edge** while invoking multi-node heterogeneous compute through the API.

We evaluate the framework by extending the Gemma-4-31B-it (Team, 2026) vision–text backbone with speech. The resulting system is trained through the TiS boundary and evaluated on FLEURS (Conneau et al., 2023) over 70 languages and all 70x69 translation directions. Our model achieves performance comparable to Gemini-3.5-Flash-Lite. We also report service-level metrics and throughput scaling across pool nodes. We will release the code and models to support multimodal research.

The contributions of this paper are as follows.

1. **API-driven compute sharing.** TiS lets independently managed services expose idle capacity through authenticated endpoints without acquiring server control. Admission remains under the service owner's policy. Providing cloud capacity earns credits, while requesting cloud services consumes them.

2. **Hardware decoupling and low-bandwidth communication.** Accelerator classes and tensor-parallel layouts remain behind one API contract; only embeddings and gradients cross a boundary designed for 1 Gbps LAN links and independent of model-scale synchronization.

3. **31B multimodal-extension evaluation.** We extend a 31B vision–text backbone with speech, evaluate all 70×69 many-to-many speech-to-text translation directions on FLEURS, and report the framework's service throughput.

2 Background and Motivation

2.1 Fragmented vLLM Services in an Enterprise LAN

An enterprise may own thousands of accelerators while making only a fraction available to any one team. Let $\mathcal{V} = \{0, 1, \ldots, N\}$ denote the participants in one LAN: a training client owns the modality data $\mathcal{D}$ and the trainable encoder/adapter, while services $j \in \{1, \ldots, N\}$ expose independent vLLM deployments. Each service is characterized by

$$s_j = (a_j, d_j, q_j, \kappa_j, u_j), \tag{1}$$

where a_j is the accelerator class, d_j the frozen checkpoint identifier, q_j the local tensor-parallel layout, κ_j the vendor collective library, and u_j the currently available service capacity. In the current deployment, a_j includes A100 GPUs and 910C NPUs, with $\kappa_j \in \{\mathrm{NCCL}, \mathrm{HCCL}\}$. Services with different d_j form different pools; only replicas with the same checkpoint and tokenizer are interchangeable.

Two properties define the pool. (C1) No spanning collective. NCCL and HCCL cannot form one communicator over the set of heterogeneous services, so the pool cannot rely on a shared rank group. (C2) Ownership and locality. Machines, model versions, and local workloads remain under their owners' control. The pool may route authenticated requests to a service, but it cannot assume SSH access, replace the service's parallel layout, or move its model state. TiS therefore treats each service as an independent replica and changes what crosses the service boundary rather than coupling all replicas into one training job.

2.2 Roles and Training Objective

Participants may act as edge clients or cloud providers. Providing inference or gradient capacity earns credits, while requesting compatible cloud services consumes them, enabling idle compute to be reused. Admission never exceeds advertised spare capacity, and local workloads retain priority.

Let x denote a speech waveform and y a task target. A training client maps x to an embedding sequence $\boldsymbol{h} = F_\theta(x) \in \mathbb{R}^{K \times D}$, where K is the number of adapter tokens and D is the language-model embedding dimension. A service with the matching frozen checkpoint computes

$$z = G_\phi(\boldsymbol{h}, p), \qquad L = \ell(z, y), \tag{2}$$

and returns $g_{\boldsymbol{h}} = \partial L / \partial \boldsymbol{h}$. The client minimizes $\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(\theta, \psi; x, y)]$ over its local parameters while keeping raw data, trainable parameters, and optimizer state local. Labels are represented only by token IDs at the service boundary. The service parameters ϕ remain under the model owner's policy and are frozen in the current system.

2.3 Communication Cost

We now estimate communication for updating the local encoder and adapter. Suppose the edge encoder and adapter contain 1B trainable parameters, while the cloud-side 31B LLM is frozen. As a reference, suppose these 1B edge parameters were synchronized with data parallelism (DP) across four nodes in BF16 ($b = 2$ bytes). Ring all-reduce moves

$$C_{\mathrm{DP}} = \frac{2(N-1)}{N} \Theta b \tag{3}$$

per node per step, where $N = 4$ and $\Theta = 1\mathrm{B}$. Each node therefore exchanges about 3 GB, with about 12 GB aggregate traffic. DP communication grows linearly with the number of trainable parameters.

TiS does not synchronize these parameters: the encoder and adapter are updated at the edge, and the service returns input-embedding gradients. For a micro-batch of $B = 256$ samples, $K = 80$ adapter tokens, hidden dimension $D = 5376$, and BF16, the bidirectional payload per step is

$$C_{\mathrm{TiS}} = 2BKDb \approx 0.44\,\mathrm{GB}. \tag{4}$$

With four workers, each handles about 0.11 GB; labels are negligible. Unlike DP, TiS communication depends only on batch size, adapter length, hidden dimension, and precision, so it does not grow with the number of trainable parameters.

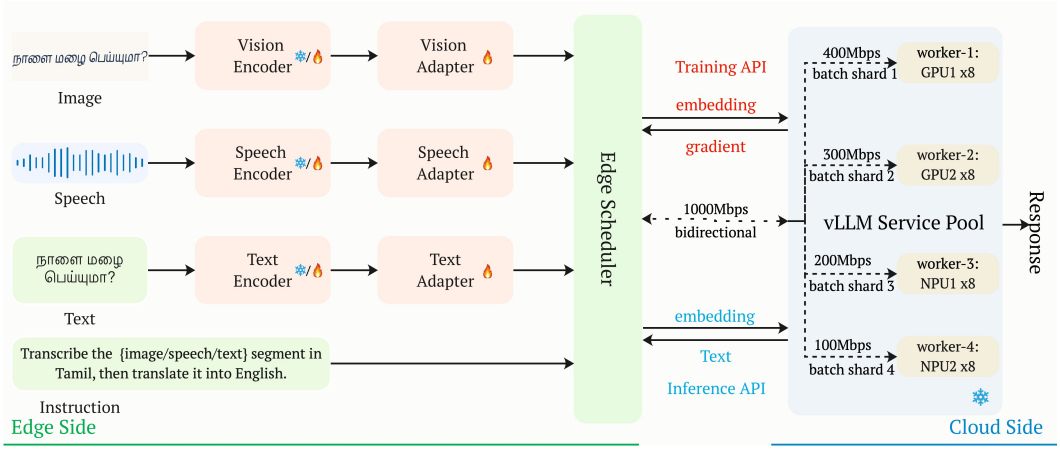

Figure 2: The TiS framework, divided into edge and cloud sides. The edge keeps raw data, trainable parameters, and the local optimizer; the cloud serves the same frozen checkpoint through independent workers that retain their own accelerators and tensor-parallel layouts. Embeddings and labels travel to the cloud, input-embedding gradients return to the edge, and standard inference remains available. Workers receive data-plane requests directly; link rates illustrate different compute capabilities and loads.

3 API-Driven TiS Framework

3.1 Multimodal LLM Adaptation

A multimodal large language model (MLLM) contains a modality encoder, an adapter, and an LLM. The encoder maps raw inputs such as images or waveforms into continuous representations; the adapter converts them into a fixed number of tokens or embeddings that the LLM can consume. This decomposition keeps the LLM reusable and confines adaptation to small input-side modules. TiS places the training boundary at the adapter output, as summarized in Figure 2.

Visual MLLMs. Visual MLLMs use a vision encoder such as CLIP (Radford et al., 2021) followed by a query-based adapter. BLIP-2 (Li et al., 2023) connects a frozen image encoder to a frozen LLM through a Q-Former and trains only the adapter. The adapter is small enough to train locally, and the LLM can remain a remote frozen component.

Speech MLLMs. Speech MLLMs replace the vision encoder with an acoustic encoder such as Whisper (Radford et al., 2023), then map its representations through a Q-Former or MLP. LLM-SRT (Du et al., 2025) and MCAT (Du et al., 2026) adapt an LLM for many-to-many speech translation by training the speech-side interface while leaving the language model largely fixed. In TiS, the edge trains this interface and the frozen LLM computes the loss and returns gradients at the adapter output.

Multilingual MLLMs. Multilingual MLLMs extend the same decomposition across languages. NLLB (Costa-jussà et al., 2022) provides broad coverage as a multilingual encoder, while language conditioning and curriculum learning remain important for low-resource languages. LinguaLift (Zhang et al., 2026) shows that two-stage instruction tuning can improve low-resource reasoning by adapting the input-side modules without changing the full backbone. These components fit the same TiS boundary.

Training with TiS. TiS turns this decomposition into a remote training contract: the encoder, adapter, data, and optimizer stay at the edge, while the service returns the embedding gradient. The same deployment exposes inference and gradient return without exposing model weights or parallel layout.

3.2 TiS System Design

TiS separates the training problem into four components: a heterogeneous scheduler, a remote gradient path, a star-shaped communication contract, and an elastic service pool. The edge node owns the raw data and the trainable modality module; each service node owns a replica of the frozen LLM. The components are exposed through one API, so the local training loop does not need to know where or how the LLM is executed.

Dynamic heterogeneous scheduler. The scheduler is the control plane. It measures only two inputs from a service node: token throughput r_i and link rate. These two values determine the computation budget and transfer time for that node. An owner starts a Docker worker; the scheduler probes these two values and adds the worker to the scheduling pool.

For each optimizer step, the scheduler assigns a subset S_i of the global batch to node i. Training length-sorts samples so batches are homogeneous; the scheduler balances token time, not sample count. Let T_i^{comp} and T_i^{comm} be the estimated computation and communication times of node i. The assignment solves

$$\min_{\{S_i\}} \max_i \left| T_i^{\text{comp}} + T_i^{\text{comm}} - \bar{T} \right|, \qquad \sum_i |S_i| = B, \quad S_i \cap S_j = \varnothing, \tag{5}$$

where B is the global batch size and $\bar{T}$ is the target completion time. The scheduler estimates T_i^{comp} from measured tokens per second and the token lengths in S_i, and estimates T_i^{comm} from the embedding payload and observed link rate. A fast node receives more or longer samples; a slow node receives fewer. The objective is equal-time completion, not equal sample counts, so the synchronous step does not idle on a straggler.

The scheduler also divides each assignment into bounded chunks and streams them through the forward and backward paths instead of sending the whole step at once. This flattens bandwidth spikes and overlaps data movement with remote computation. Smaller chunks reduce burst bandwidth but add round trips; the chunk size is therefore selected jointly with the per-node token budget.

Gradient synchronization and communication topology. TiS uses an edge-to-multi-cloud star topology. For a sample chunk, the edge computes $\boldsymbol{h} = A_\psi(E_\omega(x)) \in \mathbb{R}^{K \times D}$ with $K = 80$ and sends $\boldsymbol{h}$ plus labels y to one selected worker. The worker treats $\boldsymbol{h}$ as a leaf tensor, runs the frozen LLM forward pass, computes $L = \ell(G_\phi([\boldsymbol{h}; P]), y)$, and returns $g_{\boldsymbol{h}} = \partial L / \partial \boldsymbol{h}$. The edge then backpropagates $g_{\boldsymbol{h}}$ through its local encoder and adapter graph and updates ω and ψ with its local optimizer. Model weights, vocabulary logits, output-side gradients, and per-layer activations remain on the service; raw data and trainable parameters remain on the edge.

We implement the gradient path as a sidecar around vLLM. The engine enters gradient mode, disables the inference KV-cache path, recomputes decoder activations during backward, and chunks the vocabulary projection so that full $[B, T, V]$ logits are never retained. All local tensor-parallel ranks agree on the loss, token count, and returned gradient. Let B_i be the number of samples assigned to worker i and b the bytes per element. The bidirectional payloads are

$$C_i = 2B_i K D b, \qquad C_{\text{edge}} = \sum_{i=1}^{R} C_i = 2BKDb. \tag{6}$$

The edge is the fan-out and fan-in point: its aggregate traffic scales with the global batch, while each worker carries only its assigned shard. Workers do not exchange parameters, activations, or gradients with one another, so the topology is a star rather than a ring or a cross-worker collective.

With four workers, the edge exchanges about $0.44\,\text{GB}$ per step, but each worker exchanges only about $0.11\,\text{GB}$; with eight workers, the per-worker payload falls to about $0.055\,\text{GB}$. The edge link is the busiest link, while each remote worker sees a small point-to-point payload.

Geo-distributed and cross-region training. The star topology removes the need for a collective between service workers. Each worker communicates only with the edge through an independent request-response session, and its payload is fixed by B_i, K, D, and b. Cross-region deployment therefore places independent sessions in different regions rather than constructing one wide-area NCCL or HCCL collective.

Table 1: TiS communication for one optimizer step with $B = 256$, $K = 80$, $D = 5376$, and BF16. Values are bidirectional payloads; labels are negligible and transfer times exclude round-trip latency.

Scope	Payload per step	Example	1 Gbps time
Edge aggregate	$2BKDb$	0.44 GB	3.5 s
Each of 4 workers	$2(B/4)KDb$	0.11 GB	0.9 s
Each of 8 workers	$2(B/8)KDb$	0.055 GB	0.4 s

Workers do not synchronize at every layer or share an activation buffer. A slow region affects only the samples assigned to it; the scheduler can reduce its token budget and give the remaining samples to other regions. Because per-worker payloads are small, multiple regional transfers can proceed in parallel and overlap with local encoder computation. A synchronous step still waits for all gradients, but the critical path is the slowest edge-worker session, not a cross-region collective ring.

Elastic expansion and fault-tolerant re-dispatch. A new service node can join at a step boundary. The scheduler measures its token throughput, adds it to the routing pool, and assigns it a share of the next step's token budget. No other worker restarts or changes its parallel layout, because workers do not share a collective and the frozen model is stateless between optimizer steps. If a worker leaves, the scheduler removes it from future assignments and redistributes its remaining chunks.

The scheduler monitors request timeouts. If a worker fails during a forward or backward call, the corresponding sample IDs return to the pending queue and are re-dispatched to surviving workers. The edge does not commit a partial optimizer step; it waits for the complete global batch or retries or skips the missing samples according to policy. Duplicate retries can be removed by sample ID before the optimizer update. This keeps data completeness without restarting the full job when one node disappears.

Researcher usability. The researcher keeps a standard local DDP program, including the data loader, encoder, adapter, optimizer, gradient accumulation, and checkpointing. Only the frozen LLM computation is remote. A custom autograd function exposes the service boundary: its forward receives the embedding and labels and returns the loss, and its backward requests the input-embedding gradient. The rest of the local graph remains ordinary PyTorch.

Conceptually, the training loop remains

```
h = encoder_adapter(x); L = tis.forward(h, y);
g = tis.backward(L); h.backward(g); optimizer.step();
```

The researcher does not select endpoints, assign samples, choose accelerators, configure tensor or pipeline parallelism, or implement retries; those belong to the scheduler and service workers. Worker joins or leaves affect only the scheduler, leaving the model, loss, optimizer, and DDP wrapper unchanged. The same endpoint also serves inference when no adaptation call is active.

Compute sharing. TiS does not target maximum per-device efficiency. API-mediated execution and Docker entry add overhead, so an individual accelerator can be less utilized than in a tightly coupled cluster. The benefit is wall-clock time: drawing on a larger pool of otherwise idle, independently managed machines lets a job finish sooner even when that pool is not the fastest way to run each step in isolation.

Credits give idle capacity a durable unit of account: a provider that contributes spare GPU or NPU capacity earns them, a client that requests service spends them, and the balance can be redeemed later as a large batch of training or inference. Admission is bounded by advertised spare capacity, and local workloads retain priority. When demand exceeds available capacity, compatible work is queued while each provider's existing jobs remain untouched; when capacity returns, the same worker can rejoin without changing the edge-side training graph. This turns fragmented, temporarily unused resources into a shared pool without a shared administrator.

4 EXPERIMENTAL SETTINGS

Datasets. We evaluate TiS on many-to-many S2TT over FLEURS (Conneau et al., 2023), covering 70 source languages and 69 non-identity targets ($70 \times 69 = 4{,}830$ directions). The appendix provides all 70 source languages. The speech model is pretrained on Common Voice 24 (Ardila et al., 2020), while FLEURS is used for downstream evaluation. Raw speech and trainable parameters remain at the edge; only embeddings, labels, and embedding gradients cross the TiS boundary.

Training Details. We extend Gemma-4-31B-it (Team, 2026) with speech using a frozen Whisper encoder (Radford et al., 2023) and a trainable 80-token Q-Former/MLP adapter; the 31B LLM remains frozen behind TiS. The edge sends BF16 embeddings of shape 80×5376 over a 1 Gbps LAN and receives the embedding gradient. We evaluate two hardware settings: (i) gradient precision, comparing $2\times$A100 (TP=1) against $4\times$910C (TP=2) on identical embeddings, and (ii) speed scaling from one to four $8\times$910C service nodes. With an $8\times$910C edge and total batch $8\times128 = 1{,}024$, the full 31B adaptation run completes in three days on four nodes.

Baselines. We compare against Gemini-3.5-Flash-Lite (Google DeepMind, 2026), SeamlessM4T-V2-Large (Seamless Communication, 2025), Voxtral-24B (Liu et al., 2025), and MCAT-27B (Du et al., 2026) on the same FLEURS directions. The quality comparison and hardware studies use the same frozen checkpoint and service contract. All measurements use the same model checkpoint, tokenizer, and BF16 service contract.

Evaluation Metrics. We report spBLEU with the FLORES-200 tokenizer and COMET, using beam size 1. The appendix provides the per-target and full per-language scores; identity directions are excluded from the aggregate grid, and all quality results are averaged over the same non-identity direction set. Quality scores are computed after decoding, whereas the precision test compares gradient tensors directly and the speed test measures completed optimizer steps. The precision runs keep embeddings identical, and the throughput runs keep the global batch fixed.

Table 3: Language support.

Resource	Languages (ISO Code)
High	Arabic (ara), Belarusian (bel), Bengali (ben), Catalan (cat), Czech (ces), Mandarin Chinese (cmn), German (deu), English (eng), Finnish (fin), French (fra), Italian (ita), Japanese (jpn), Dutch (nld), Polish (pol), Romanian (ron), Spanish (spa)
Medium	Welsh (cym), Danish (dan), Greek (ell), Estonian (est), Hindi (hin), Croatian (hrv), Hungarian (hun), Indonesian (ind), Javanese (jav), Kazakh (kaz), Korean (kor), Portuguese (por), Russian (rus), Slovak (slk), Swahili (swh), Tamil (tam), Telugu (tel), Tagalog (tgl), Thai (tha), Turkish (tur), Ukrainian (ukr), Urdu (urd), Uzbek (uzb), Vietnamese (vie)
Low	Afrikaans (afr), Amharic (amh), Assamese (asm), North Azerbaijani (azj), Bosnian (bos), Bulgarian (bul), Persian (fas), Galician (glg), Gujarati (guj), Hebrew (heb), Armenian (hye), Icelandic (isl), Kannada (kan), Georgian (kat), Khmer (khm), Kyrgyz (kir), Lao (lao), Latvian (lav), Lithuanian (lit), Malayalam (mal), Macedonian (mkd), Malay (msa), Burmese (mya), Norwegian Bokmål (nob), Nepali (npi), Punjabi (pan), Slovenian (slv), Serbian (srp), Swedish (swe), Cantonese (yue)

Table 2: Hardware, throughput, and precision comparisons.

Nodes	Workers	Samples/s	Tokens/s	Relative
1	4	16.2	2242.7	1.00×
2	8	32.2	4457.6	1.99×
4	16	60.7	8397.9	3.75×

Device	Cosine median	Cosine range
910C	0.945	0.802–0.967
A100	0.966	0.894–0.974

Each node uses $8\times$910C, TP=2, and four vLLM workers.

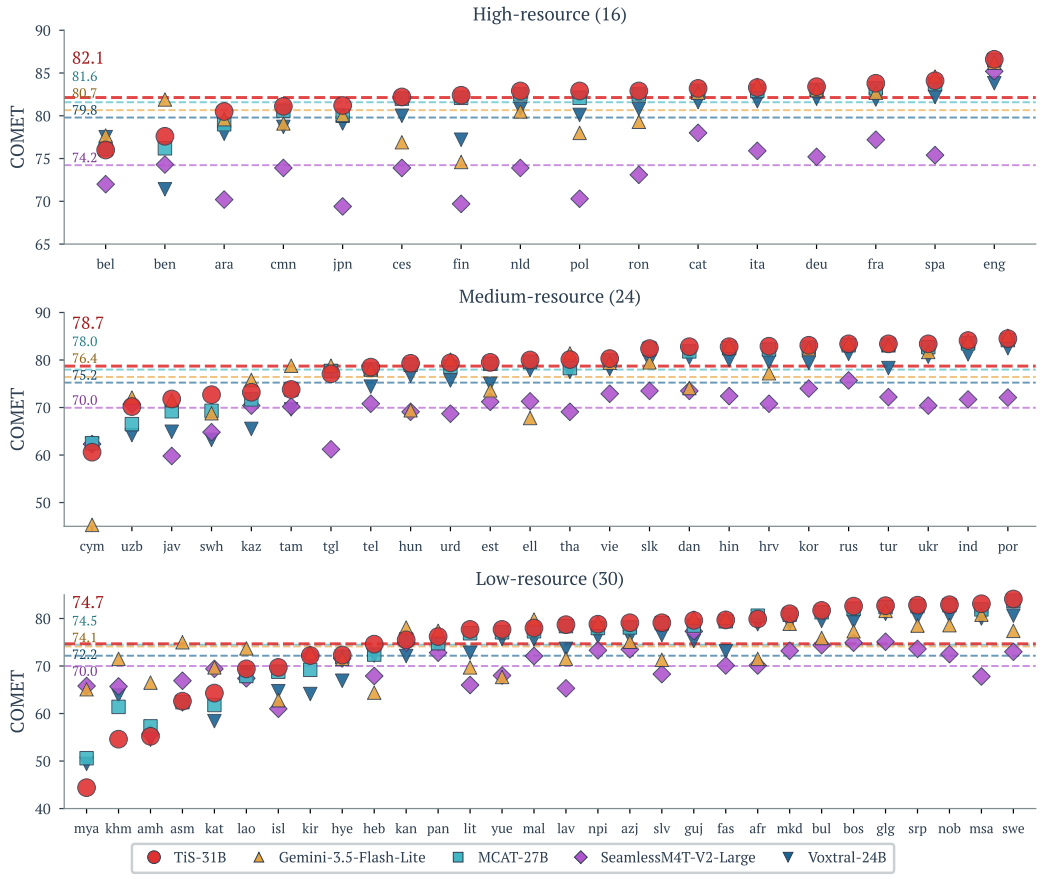

Figure 3: FLEURS COMET for 70 source languages, averaged over 69 non-identity targets and split by resource tier; Tables 4 and 5 report the corresponding results.

5 Multilingual Speech Adaptation

5.1 Main Results

Model performance. TiS-31B is evaluated on FLEURS across the full multilingual setting. Its performance is close to Gemini-3.5-Flash-Lite and outperforms the open-source baselines MCAT-27B, SeamlessM4T-V2-Large, and Voxtral-24B. Figure 3 shows that this result holds across the resource tiers. The evaluation covers 16 high-, 24 medium-, and 30 low-resource languages, and the appendix reports all per-target and per-language scores. The distributed service only executes the frozen LLM, so the quality claim concerns the resulting model.

Throughput. Table 2 reports throughput for one, two, and four 8×910C nodes, with TP=2 and four vLLM workers per node. The runs use 4, 8, and 16 workers and reach 16.2, 32.2, and 60.7 samples/s, or 2242.7, 4457.6, and 8397.9 tokens/s. Relative to one node, these values are 1.00×, 1.99×, and 3.75×, and the two- and four-node runs retain 99.5% and 93.8% of linear scaling. The four-node configuration therefore provides the highest measured pooled throughput.

Precision and convergence. Table 2 also reports same-machine gradient consistency. A100 is slightly better, with median cosine 0.966 vs. 0.945 for 910C and a narrower range (0.894–0.974 vs. 0.802–0.967). Both 910C-only and mixed 910C+A100 training converge, showing that heterogeneous training works without convergence degradation across six paired measurements. Both configurations therefore reach the same qualitative conclusion despite the small numerical gap.

6 Related Work

Speech and multimodal models. Multimodal LLMs pair a modality encoder with a lightweight adapter and a language model. Visual systems such as BLIP-2 connect a frozen image encoder to a frozen LLM through a Q-Former (Li et al., 2023), while speech systems use Whisper-style acoustic encoders with projection or query adapters (Radford et al., 2023). Qwen-Audio and Qwen2-Audio extend this interface to broader audio understanding (Chu et al., 2023; 2024), and many-to-many systems such as SeamlessM4T, Qwen3-Omni, Voxtral, and MCAT scale the same encoder–adapter–LLM pattern across languages (Seamless Communication, 2025; Xu et al., 2025; Liu et al., 2025; Du et al., 2026). In these systems, the adapter defines the modality interface, while the LLM supplies the target-conditioned representation and decoding capacity. The encoder and adapter are therefore natural local training modules. The LLM, however, remains part of the same coordinated training job. Consequently, an existing serving deployment cannot serve as the training backend unless its serving path can also return the required gradient. TiS keeps the module decomposition but places the frozen LLM behind a remote gradient API.

Split learning and serving. Split learning partitions a network and exchanges activations and gradients across the boundary (Vepakomma et al., 2018); Auto-Split automates split-point selection (Banitalebi-Dehkordi et al., 2021), and Splitwise separates prefill and decode phases across machines (Patel et al., 2024). These systems move the model boundary for collaborative training or serving, but assume coordinated peers or model fragments held by the participants. Cluster schedulers such as Ray (Moritz et al., 2018) instead assume a shared resource domain, and inference APIs expose generation rather than training gradients. TiS uses an already deployed frozen LLM: the edge keeps raw data and trainable adapters, the service returns input-embedding gradients, and the client needs no model weights, optimizer state, or parallel layout. Compatible replicas can therefore serve the same local training graph without coupling participants into one collective. The same deployment can continue to handle inference requests while the endpoint is used for gradient return.

7 Conclusion

TiS changes the deployment question from which site can hold the model to which computation must cross the service boundary. The edge owns the data and trainable input-side modules, while the provider exposes a frozen LLM through a fixed-width API that returns input-embedding gradients. Workers remain independent: they do not form a collective or exchange model states. The scheduler balances token budgets using measured throughput and re-dispatches failed chunks without restarting the job. On FLEURS, TiS-31B is comparable to the strongest multilingual baselines, and the four-node service retains 93.8% of linear scaling. The present contract trains input-side modules; extending it to parameter-efficient service-side updates is the next step. Both 910C-only and mixed 910C+A100 training converge under the same API contract.

Reproducibility Statement

Section 3 specifies the API contract, scheduler, and gradient path, and Section 4 gives the datasets, training configuration, baselines, and metrics. The main model uses Gemma-4-31B-it with a frozen Whisper encoder and an 80-token Q-Former/MLP adapter, trained with total batch 1,024 over 1 Gbps links. The precision study uses 2×A100 (TP=1) and 4×910C (TP=2) with BF16 embeddings; the throughput study uses one, two, and four 8×910C nodes with TP=2 and four vLLM workers per node. Evaluation covers FLEURS across 70 source languages and 69 non-identity targets (4,830 directions), with spBLEU/COMET, beam size 1; per-target and per-language scores are reported in the appendix. We will release the implementation, service configuration, adapter checkpoints, and prompts.

AI Use Statement

Generative AI tools were used only for grammar checking. The authors are responsible for all technical claims, citations, and results.

REFERENCES

R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber. Common voice: A massively-multilingual speech corpus. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pp. 4211–4215, 2020.

Amin Banitalebi-Dehkordi, Naveen Vedula, Jian Pei, Fei Xia, Lanjun Wang, and Yong Zhang. Auto-split: A general framework of collaborative edge-cloud ai. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 2543–2553, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi:10.1145/3447548.3467078. URL https://doi.org/10.1145/3447548.3467078.

Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. Qwen-audio: Advancing universal audio understanding via unified large-scale audio-language models. *arXiv preprint arXiv:2311.07919*, 2023.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024.

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.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.

Yexing Du, Youcheng Pan, Ziyang Ma, Bo Yang, Yifan Yang, Keqi Deng, Xie Chen, Yang Xiang, Ming Liu, and Bing Qin. Making llms better many-to-many speech-to-text translators with curriculum learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12466–12478, 2025.

Yexing Du, Kaiyuan Liu, Youcheng Pan, Bo Yang, Keqi Deng, Xie Chen, Yang Xiang, Ming Liu, Bing Qin, and YaoWei Wang. Mcat: Scaling many-to-many speech-to-text translation with mllms to 70 languages. *IEEE Transactions on Audio, Speech and Language Processing*, 2026.

Google DeepMind. Gemini 3.5 Flash Lite. Gemini API model documentation, 2026. URL https://ai.google.dev/gemini-api/docs/models. Accessed: 2026-09-25.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PMLR, 2023.

Alexander H Liu, Andy Ehrenberg, Andy Lo, Clément Denoix, Corentin Barreau, Guillaume Lample, Jean-Malo Delignon, Khyathi Raghavi Chandu, Patrick von Platen, Pavankumar Reddy Muddireddy, et al. Voxtral. *arXiv preprint arXiv:2507.13264*, 2025.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 561–577, 2018.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132, 2024. doi:10.1109/ISCA59077.2024.00019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pp. 28492–28518. PMLR, 2023.

Seamless Communication. Joint speech and text machine translation for up to 100 languages. *Nature*, 637(8046):587–593, 2025.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, et al. Qwen3-omni technical report. *arXiv preprint arXiv:2509.17765*, 2025.

Hongbin Zhang, Kehai Chen, Xuefeng Bai, Yang Xiang, and Min Zhang. Lingualift: An effective two-stage instruction tuning framework for low-resource language reasoning. *IEEE Transactions on Audio, Speech and Language Processing*, 2026.

Table 4: S2TT quality (spBLEU / COMET) for English-to-69 FLEURS directions.

eng→	Gemini-3.5 -Flash-Lite	SeamlessM4T -V2-Large	Voxtral -24B	MCAT -27B	TiS -31B
afr	38.5 / 82.5	40.1 / 85.4	37.9 / 85.7	40.7 / 85.1	39.7 / 86.0
amh	26.8 / 85.3	27.7 / 86.9	2.0 / 41.8	16.8 / 82.8	25.4 / 86.7
ara	34.4 / 83.8	35.1 / 84.5	34.9 / 85.9	35.7 / 86.1	38.8 / 86.8
asm	20.6 / 80.6	20.1 / 81.4	5.0 / 67.2	13.7 / 79.3	18.7 / 81.4
azj	22.0 / 85.9	20.7 / 83.8	17.8 / 85.3	17.8 / 84.8	21.3 / 86.8
bel	24.7 / 83.9	23.0 / 83.5	23.0 / 85.1	22.3 / 85.8	20.1 / 82.6
ben	28.7 / 85.1	30.0 / 84.6	26.2 / 84.2	26.9 / 85.2	29.4 / 86.6
bos	37.5 / 88.9	34.6 / 88.7	33.9 / 89.9	33.6 / 89.7	36.3 / 89.8
bul	43.7 / 88.9	43.5 / 88.8	40.3 / 89.6	42.2 / 90.0	43.1 / 89.9
cat	41.5 / 85.3	40.5 / 85.0	41.5 / 86.8	39.6 / 85.9	38.7 / 85.1
ces	37.2 / 89.1	34.8 / 88.0	35.1 / 90.6	33.8 / 90.1	34.2 / 90.0
cmn	33.4 / 86.5	23.0 / 79.7	36.6 / 88.3	33.0 / 87.2	35.5 / 87.8
cym	46.3 / 83.5	44.6 / 84.3	33.7 / 81.3	32.9 / 81.3	24.5 / 73.2
dan	43.4 / 88.7	43.5 / 88.7	43.7 / 89.6	41.6 / 89.5	44.9 / 90.2
deu	41.8 / 86.2	39.2 / 84.9	45.8 / 88.1	39.2 / 86.5	41.6 / 87.0
ell	34.2 / 87.2	33.3 / 87.5	33.1 / 88.3	33.4 / 88.7	32.9 / 87.8
est	34.6 / 89.8	29.8 / 87.7	25.6 / 87.7	27.9 / 89.5	22.7 / 84.3
fas	33.2 / 86.5	31.6 / 84.6	29.9 / 86.1	32.3 / 86.9	32.4 / 87.2
fin	35.4 / 91.0	29.0 / 88.5	29.7 / 91.0	30.6 / 91.3	33.9 / 91.7
fra	48.4 / 85.8	48.2 / 85.3	53.0 / 88.0	46.1 / 86.3	48.6 / 86.2
glg	35.9 / 84.7	32.9 / 84.5	36.2 / 86.8	32.8 / 85.5	33.0 / 85.2
guj	31.1 / 86.7	32.4 / 86.5	23.9 / 84.9	28.2 / 87.1	29.5 / 87.4
heb	39.7 / 85.2	37.9 / 84.5	35.7 / 86.6	37.8 / 87.2	40.6 / 87.6
hin	32.0 / 79.0	33.8 / 78.1	30.1 / 78.2	30.5 / 78.9	33.1 / 80.2
hrv	35.8 / 89.0	32.3 / 87.8	31.6 / 88.9	31.1 / 89.2	32.9 / 89.3
hun	33.7 / 88.1	31.0 / 86.0	28.7 / 87.2	27.8 / 87.6	27.8 / 86.2
hye	36.7 / 88.5	33.0 / 87.7	30.9 / 87.9	29.6 / 87.7	32.6 / 86.9
ind	39.9 / 89.9	38.2 / 89.0	39.5 / 90.4	39.1 / 90.2	41.6 / 90.6
isl	32.4 / 83.6	28.4 / 82.2	20.6 / 78.6	22.3 / 82.3	21.0 / 78.7
ita	32.7 / 86.3	30.7 / 85.1	35.3 / 87.7	31.1 / 86.8	33.5 / 87.3
jav	22.9 / 85.2	23.4 / 85.4	6.0 / 71.4	19.8 / 85.7	23.4 / 86.4
jpn	29.2 / 89.7	15.9 / 84.7	30.4 / 90.7	28.8 / 90.4	30.6 / 90.9
kan	32.8 / 84.4	32.3 / 84.0	24.0 / 82.3	25.5 / 83.8	29.9 / 85.3
kat	34.4 / 86.0	31.3 / 85.4	25.0 / 81.8	25.9 / 84.5	30.1 / 85.8
kaz	32.3 / 88.7	28.9 / 87.9	22.7 / 85.5	26.0 / 88.2	29.0 / 88.9
khm	23.5 / 81.4	16.7 / 79.9	1.0 / 48.6	16.2 / 79.7	24.1 / 83.1
kir	25.9 / 86.1	24.3 / 86.1	14.6 / 79.3	19.4 / 84.9	22.0 / 85.2
kor	27.0 / 88.4	21.5 / 85.1	25.6 / 89.2	24.6 / 88.5	27.0 / 89.7
lao	27.5 / 81.4	24.8 / 81.4	0.8 / 64.1	27.1 / 83.1	28.9 / 84.1
lav	37.5 / 88.3	30.0 / 85.9	26.9 / 85.7	29.5 / 88.5	25.8 / 84.1
lit	33.0 / 88.1	27.7 / 86.4	24.6 / 85.8	26.7 / 88.5	23.8 / 84.7
mal	35.8 / 86.8	34.2 / 85.7	24.2 / 82.4	27.2 / 85.4	33.3 / 87.7
mkd	38.8 / 86.4	37.0 / 87.2	34.8 / 88.1	33.9 / 87.7	36.1 / 87.6
msa	35.9 / 87.6	31.7 / 86.6	34.6 / 88.0	34.0 / 87.5	38.0 / 88.5
mya	21.9 / 87.2	17.2 / 85.7	8.1 / 72.7	15.2 / 85.0	25.0 / 89.1
nld	31.5 / 85.6	31.0 / 85.1	33.4 / 87.1	29.7 / 86.6	32.3 / 86.9
nob	33.8 / 88.0	33.7 / 86.8	34.9 / 89.3	31.5 / 88.7	35.3 / 89.3
npi	29.2 / 81.5	28.6 / 81.1	21.4 / 79.3	24.7 / 81.1	28.5 / 83.2
pan	32.2 / 82.1	32.7 / 82.4	26.4 / 80.8	27.2 / 81.8	31.3 / 83.5
pol	30.0 / 87.1	27.9 / 85.7	30.1 / 88.4	27.3 / 88.1	31.0 / 88.9
por	45.9 / 87.3	47.5 / 86.6	52.2 / 89.1	45.3 / 87.9	47.7 / 88.2
ron	42.0 / 88.5	39.9 / 87.8	40.0 / 89.3	38.9 / 89.1	40.7 / 88.7
rus	37.2 / 87.8	34.3 / 86.3	38.8 / 89.1	34.8 / 88.8	37.3 / 89.4
slk	36.4 / 88.5	35.3 / 87.3	30.4 / 87.6	31.4 / 88.8	30.5 / 87.7
slv	33.7 / 87.8	29.2 / 86.8	28.2 / 87.3	28.6 / 88.5	27.2 / 85.8
spa	29.6 / 84.2	28.1 / 83.2	33.6 / 86.0	29.7 / 85.2	30.7 / 84.8
srp	38.2 / 86.3	37.3 / 87.2	2.5 / 77.6	34.6 / 88.0	35.3 / 87.0
swe	44.2 / 88.7	45.1 / 88.4	46.1 / 89.9	43.3 / 89.3	45.4 / 90.1
swh	35.1 / 83.0	32.6 / 81.7	16.8 / 74.8	28.3 / 81.7	32.2 / 83.2
tam	31.5 / 87.7	32.4 / 87.3	25.4 / 86.0	29.3 / 88.3	30.1 / 88.9
tel	35.1 / 85.0	36.4 / 84.9	26.8 / 82.2	32.3 / 85.1	34.4 / 86.3
tgl	31.8 / 83.0	30.2 / 82.1	25.8 / 81.8	31.1 / 83.3	31.7 / 83.8
tha	40.7 / 87.1	30.6 / 83.3	33.9 / 86.7	37.6 / 87.7	41.9 / 88.9
tur	35.0 / 88.3	32.0 / 86.7	30.9 / 87.9	31.5 / 88.2	35.5 / 89.3
ukr	35.5 / 88.1	33.1 / 87.0	35.8 / 89.5	33.4 / 88.9	36.4 / 89.7
urd	26.4 / 81.3	25.9 / 79.4	22.0 / 78.0	22.8 / 80.9	26.8 / 82.4
uzb	31.0 / 88.2	28.6 / 87.7	21.3 / 87.3	25.6 / 88.2	29.8 / 89.2
vie	37.8 / 87.2	36.7 / 85.6	37.9 / 88.3	35.9 / 87.8	39.4 / 88.4
yue	16.4 / 81.7	15.7 / 79.8	27.7 / 88.0	27.7 / 88.0	27.8 / 87.9
Avg.	33.8 / 86.2	31.7 / 85.2	28.5 / 83.8	30.3 / 86.5	32.4 / 86.6

A Full Results

A.1 Per-Target and Per-Language Results

Table 5: S2TT quality (spBLEU / COMET) for all 70 FLEURS source languages.

Src	Gemini-3.5 -Flash-Lite	SeamlessM4T -V2-Large	Voxtral -24B	MCAT -27B	TiS -31B
afr→69	18.7 / 71.5	17.7 / 70.1	22.8 / 78.8	**23.4 / 80.6**	23.2 / 79.9
amh→69	**10.5 / 66.5**	6.9 / 55.7	3.8 / 54.5	5.0 / 57.3	4.1 / 55.2
ara→69	**22.2** / 79.6	15.9 / 70.2	18.2 / 77.9	19.0 / 79.0	21.9 / **80.5**
asm→69	**15.3 / 75.0**	11.7 / 66.9	6.2 / 61.9	6.6 / 62.4	6.7 / 62.6
azj→69	14.7 / 75.1	13.8 / 73.5	13.9 / 76.1	14.9 / 78.1	**16.5 / 79.1**
bel→69	**17.3 / 77.7**	13.4 / 72.0	16.3 / 77.5	14.1 / 76.1	13.4 / 76.0
ben→69	**20.7 / 81.9**	14.9 / 74.3	10.8 / 71.4	14.3 / 76.2	15.8 / 77.6
bos→69	22.6 / 77.3	20.0 / 74.9	20.6 / 79.4	22.6 / 82.0	**24.5 / 82.6**
bul→69	20.4 / 75.9	19.0 / 74.4	20.9 / 79.5	21.9 / 81.3	**23.4 / 81.7**
cat→69	**27.3** / 82.7	22.4 / 78.0	23.5 / 81.6	23.6 / 82.9	25.4 / **83.2**
ces→69	20.3 / 76.9	18.3 / 73.9	20.1 / 80.0	21.5 / 82.0	**23.1 / 82.2**
cmn→69	16.9 / 79.1	13.5 / 73.9	15.1 / 78.7	16.8 / 80.6	**18.4 / 81.1**
cym→69	4.4 / 45.3	**14.4** / 62.3	8.8 / 62.3	8.8 / **62.5**	6.9 / 60.6
dan→69	19.2 / 74.1	17.8 / 73.5	21.9 / 80.5	22.1 / 81.8	**24.6 / 82.8**
deu→69	**27.2** / 83.0	19.7 / 75.2	24.2 / 82.0	23.9 / 83.0	25.7 / **83.4**
ell→69	13.7 / 67.8	14.6 / 71.3	18.1 / 77.9	19.1 / 79.6	**20.2 / 80.0**
eng→69	**33.8** / 86.2	31.7 / 85.2	28.5 / 83.8	30.3 / 86.5	32.4 / **86.6**
est→69	18.9 / 73.6	16.6 / 71.2	15.3 / 75.0	18.9 / 79.2	**19.3 / 79.5**
fas→69	**22.0 / 80.1**	15.0 / 70.1	13.8 / 73.1	18.0 / 79.4	19.2 / 79.7
fin→69	16.4 / 74.6	14.0 / 69.7	16.1 / 77.2	20.2 / 82.1	**21.5 / 82.4**
fra→69	**26.3** / 82.7	19.5 / 77.2	23.3 / 81.9	23.2 / 83.2	25.4 / **83.8**
glg→69	**25.6** / 81.6	19.8 / 75.1	21.9 / 81.0	22.2 / 82.4	23.9 / **82.7**
guj→69	**19.7 / 80.2**	17.4 / 77.3	12.4 / 75.3	16.1 / 78.5	18.3 / 79.6
heb→69	13.1 / 64.4	15.8 / 67.9	14.2 / 72.2	14.6 / 72.4	**17.6 / 74.6**
hin→69	**23.3 / 83.0**	13.8 / 72.4	17.7 / 80.0	20.9 / 82.5	21.7 / 82.8
hrv→69	20.2 / 77.2	17.1 / 70.8	19.7 / 79.6	22.0 / 82.2	**23.9 / 82.9**
hun→69	14.7 / 69.4	15.5 / 69.1	16.6 / 76.7	18.6 / 79.0	**19.8 / 79.3**
hye→69	15.4 / 71.5	**15.9** / 71.6	9.5 / 66.9	13.1 / 71.9	13.7 / **72.3**
ind→69	**26.7** / 84.0	15.4 / 71.7	21.1 / 81.2	23.0 / 83.5	24.9 / **84.1**
isl→69	**12.3** / 62.8	12.1 / 61.0	8.9 / 64.7	10.1 / 68.8	11.0 / **69.7**
ita→69	**25.0 / 83.6**	17.3 / 75.9	21.7 / 81.7	21.3 / 82.9	23.0 / 83.3
jav→69	**16.2** / 71.5	10.6 / 59.8	9.1 / 64.9	11.3 / 69.2	14.2 / **71.8**
jpn→69	18.1 / 80.1	11.9 / 69.4	16.2 / 79.1	17.1 / 80.0	**19.0 / 81.2**
kan→69	**17.9 / 78.1**	15.7 / 74.9	10.9 / 72.1	14.2 / 75.6	14.0 / 75.5
kat→69	13.0 / **69.7**	**13.4** / 69.4	4.9 / 58.4	6.9 / 61.8	7.2 / 64.3
kaz→69	**16.2 / 75.8**	12.9 / 70.4	8.4 / 65.5	11.9 / 71.7	13.0 / 73.2
khm→69	**11.4 / 71.5**	8.8 / 65.7	5.8 / 63.8	4.6 / 61.4	3.4 / 54.6
kir→69	12.4 / **72.7**	**13.6** / 72.2	7.3 / 64.1	9.5 / 69.2	11.0 / 72.2
kor→69	20.5 / 82.4	14.3 / 74.0	16.2 / 79.4	19.7 / 82.6	**21.2 / 83.1**
lao→69	**14.1 / 73.7**	10.5 / 67.4	8.8 / 68.7	8.0 / 67.9	10.2 / 69.4
lav→69	17.4 / 71.5	14.1 / 65.3	14.4 / 73.6	18.4 / 78.4	**19.2 / 78.7**
lit→69	15.2 / 69.7	13.9 / 66.0	13.4 / 72.8	16.5 / 76.9	**17.6 / 77.7**
mal→69	**18.9 / 79.8**	14.0 / 72.1	13.8 / 75.6	14.7 / 77.3	16.1 / 78.0
mkd→69	**24.0** / 78.9	18.7 / 73.2	21.3 / 78.9	22.1 / 80.8	23.5 / **81.0**
msa→69	23.3 / 80.8	13.1 / 67.8	19.9 / 80.0	21.7 / 81.9	**24.1 / 83.1**
mya→69	6.2 / 65.1	**9.0 / 65.8**	1.5 / 49.4	1.4 / 50.6	1.0 / 44.4
nld→69	22.2 / 80.5	16.7 / 73.9	20.3 / 80.8	21.0 / 82.5	**22.5 / 82.9**
nob→69	22.6 / 78.6	16.6 / 72.5	21.4 / 80.5	22.8 / 82.8	**24.1 / 82.9**
npi→69	**17.8 / 79.1**	14.2 / 73.3	12.6 / 76.2	14.9 / 78.0	16.8 / 78.8
pan→69	**18.3 / 77.4**	14.8 / 72.8	13.0 / 74.4	13.8 / 74.8	16.4 / 76.2
pol→69	19.6 / 78.0	14.3 / 70.3	19.5 / 80.1	20.6 / 82.1	**22.5 / 82.9**
por→69	**30.3 / 85.1**	16.7 / 72.1	24.9 / 82.5	25.1 / 84.2	26.9 / 84.5
ron→69	23.7 / 79.3	19.0 / 73.1	22.5 / 80.8	23.6 / 82.5	**25.0 / 82.9**
rus→69	**26.5 / 83.5**	18.7 / 75.7	22.5 / 81.3	22.9 / 83.1	24.4 / 83.4
slk→69	22.3 / 79.5	18.2 / 73.5	20.2 / 79.7	22.2 / **82.4**	**23.5 / 82.4**
slv→69	15.6 / 71.3	15.0 / 68.3	16.6 / 76.5	18.7 / 79.0	**19.7 / 79.1**
spa→69	**25.6 / 84.6**	16.3 / 75.4	21.8 / 82.2	21.9 / 83.7	23.3 / 84.1
srp→69	24.4 / 78.5	19.2 / 73.6	22.2 / 80.3	24.4 / 82.6	**25.8 / 82.8**
swe→69	21.4 / 77.4	17.7 / 73.0	21.5 / 80.6	24.0 / 83.7	**25.9 / 84.1**
swh→69	**14.9** / 68.8	13.5 / 64.8	8.2 / 63.2	12.3 / 69.3	**14.9 / 72.7**
tam→69	**17.3 / 78.8**	12.8 / 70.2	9.3 / 69.6	12.5 / 73.8	12.3 / 73.8
tel→69	**18.0** / 78.0	13.6 / 70.8	12.8 / 74.4	16.2 / 78.0	17.2 / **78.5**
tgl→69	**22.0 / 78.8**	11.9 / 61.2	19.4 / 77.6	17.7 / 77.7	17.7 / 77.1
tha→69	**19.9 / 81.4**	11.7 / 69.1	14.6 / 77.5	14.8 / 78.3	17.6 / 80.1
tur→69	**25.1** / 83.1	16.0 / 72.2	18.2 / 78.3	22.8 / 83.2	23.9 / **83.4**
ukr→69	**26.0** / 81.7	16.8 / 70.4	21.8 / 80.7	23.4 / 82.7	24.9 / **83.4**
urd→69	**19.6 / 80.1**	11.7 / 68.7	14.3 / 75.7	16.5 / 78.6	18.3 / 79.4
uzb→69	14.1 / **72.1**	**14.9** / 70.7	7.9 / 64.2	9.7 / 66.5	11.5 / 70.2
vie→69	**19.4** / 79.4	14.3 / 72.9	15.9 / 78.1	17.1 / 79.9	18.8 / **80.3**
yue→69	7.9 / 67.7	10.8 / 68.0	12.3 / 75.7	13.3 / 77.1	**15.3 / 77.7**
Avg.	**19.2** / 76.4	15.3 / 70.9	15.9 / 75.0	17.4 / 77.3	18.8 / **77.8**