Devstral Audio

Abstract

We present Devstral Audio, a 24B-parameter multimodal model that accepts coding instructions in either speech or text and executes agentic software engineering tasks. Devstral Audio is constructed by merging the language decoder weights of two sibling models that share a common architecture but specialize in complementary domains: Voxtral Small, an audio-language model with strong speech understanding, and Devstral Small, a coding-agent model with state-of-the-art performance on agentic benchmarks. Because both models descend from the same base (Mistral Small), their decoder weights occupy similar regions of parameter space, making weight-space combination feasible. We systematically evaluate three merge strategies (linear interpolation, SLERP, and DELLA) across a grid of hyperparameters, selecting the configuration that best balances audio comprehension and coding capability. The resulting merged checkpoint is then refined through supervised fine-tuning on a carefully constructed mixture of text-based coding trajectories and synthetically generated audio coding data. Our best model recovers the full audio understanding performance of Voxtral (74.0% vs 74.3% MMLU Audio) while matching Devstral's agentic coding score (67% SWE-Bench Verified), with certain configurations even exceeding the individual parent models on specific benchmarks. We release the model weights and describe a general, reproducible recipe for cross-modal model merging that preserves parent capabilities with minimal regression.

Webpage:	TODO:Sid
Model weights:	TODO:Sid
	TODO:Sid
Evals:	TODO:Sid

[NOTE TO EVERYONE:] DEVSTRAL SMALL SCORES ARE REPORTED 67% AT SOME PLACES AND 68% AT OTHERS. WITH SID's EXPERIMENT, WE GOT 67% BUT BLOG MENTIONS 68%. OUR BEST CKPT IS ALSO AT 67%

1 Introduction

Large language models are increasingly deployed as autonomous coding agents that can navigate codebases, edit files, run tests, and resolve real-world software issues **?**. Separately, audio-language models have matured to the point where they can follow spoken instructions, answer questions about audio content, and transcribe speech with high fidelity **?**. Combining these capabilities into a single model, one that a developer could speak to naturally and receive fully autonomous code changes in return, is a compelling but underexplored direction.

Training such a model from scratch would require joint pretraining across audio, text, and code corpora at enormous computational cost. An attractive alternative is *model merging*: combining

the weights of two existing specialist models into a single checkpoint that inherits capabilities from both. Recent work has demonstrated that merging can be effective when the constituent models share architectural structure and descend from a common base **????**. Techniques range from simple linear interpolation to more sophisticated approaches such as SLERP (spherical interpolation) and DELLA (drop-elect-fuse on weight deltas), each offering different trade-offs between fidelity to the parent models and robustness to destructive interference.

In this work, we apply model merging to unify **Voxtral Small ?**, a 24B-parameter audio-language model, with **Devstral Small ?**, a coding-agent model of identical decoder architecture. Both models are 24B-parameter decoder-only transformers derived from the Mistral Small family, sharing the same backbone but differing in post-training: Voxtral adds a Whisper-based audio encoder and is instruction-tuned for audio tasks, while Devstral undergoes extensive SFT and policy optimization for agentic code generation. This shared lineage makes their decoder weights amenable to interpolation. We describe the shared architecture in detail in Section 2.

However, merging alone is insufficient. While the best merge configuration preserves most of Devstral's coding performance (61.4% on SWE-Bench Verified), it severely degrades audio understanding (from 74.3% to 38% on MMLU Audio). To recover the lost audio capability without sacrificing coding, we perform targeted supervised fine-tuning on a mixture of Devstral's original text-based coding trajectories and a new synthetic dataset of audio-augmented coding instructions that we construct via a judge-then-synthesize pipeline.

Our primary contributions are:

- **Devstral Audio**, an open-weights 24B audio-language model with both agentic coding and audio instruction-following capabilities, achieving 67% on SWE-Bench Verified and 74.0% on MMLU Audio Understanding simultaneously.
- A **comprehensive analysis of cross-modal merging strategies** (linear, SLERP, DELLA) for combining an audio model with a coding model, including tokenizer alignment, evaluation across multiple benchmarks, and practical guidelines for hyperparameter selection.
- A **merge-then-SFT recipe** that recovers parent-model performance on both modalities with minimal compute, demonstrating that audio understanding can be restored with as little as 5% audio data in the fine-tuning mixture. Notably, text-only SFT alone recovers a substantial portion of audio capability, a finding with implications for understanding how merged weights retain latent modality-specific knowledge.

[TODO:Avi] Add bibtex entries for: devstral, voxtral, wortsman2022model, yadav2023tiesmerging, dare_ties, della_merging.

2 Modeling

Both parent models, Voxtral Small and Devstral Small, are 24B-parameter decoder-only transformers derived from the Mistral Small family. They share an identical decoder architecture (layer count, hidden dimension, attention heads, and vocabulary size), differing only in their post-training specialization: Devstral is fine-tuned for agentic code generation, while Voxtral undergoes multimodal pretraining and instruction tuning for audio tasks.

The key architectural distinction is Voxtral's audio front-end: a Whisper-Large-V3 encoder that produces frame-level audio representations, followed by an MLP projection layer that downsamples these representations to keep sequence lengths manageable. This projection layer maps the Whisper embedding space into the decoder's input space, after which audio tokens are processed identically to text tokens by the shared decoder.

To construct Devstral Audio, we affix Voxtral's complete audio apparatus (the Whisper encoder and MLP projection layer) to the merged decoder. No merging or training is performed on these audio components; they are preserved exactly as in the original Voxtral checkpoint. The merged decoder then processes both text tokens (from the tokenizer) and audio tokens (from the Whisper/MLP pipeline) in a unified sequence, enabling the model to accept instructions in either modality. Figure 1 illustrates this architecture.

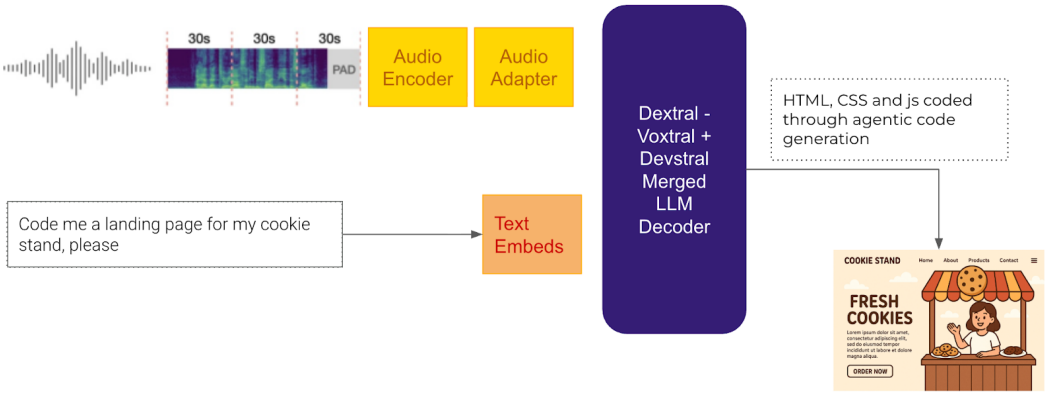

Figure 1: Dextral architecture. The Whisper encoder and MLP projection layer are taken verbatim from Voxtral (frozen, shown in blue). The language decoder is obtained by merging Devstral and Voxtral decoder weights (shown in purple). Audio and text inputs are processed through separate front-ends but share the same decoder for generation.

3 Methodology

We train the model in two phases: Model Merging and supervised finetuning. Each phase is described separately below. Finally, we describe our evaluation protocol for speech understanding tasks.

4 Model Merging

As described in Section 2, Devstral Audio preserves Voxtral's audio front-end (Whisper encoder and MLP projection) verbatim and merges only the shared language decoder weights between Voxtral and Devstral. This ensures the audio processing pipeline remains intact regardless of the merge strategy applied to the decoder.

4.1 Tokenizer Alignment

Before any weight fusion, we perform tokenizer-aware alignment to prevent special-token collisions. Both models share the same base vocabulary, but differ in their special-token assignments (e.g., audio control tokens present only in Voxtral). We identify three categories of tokens: (i) common specials present in both tokenizers, (ii) Voxtral-only specials (audio control tokens), and (iii) placeholder slots. Starting from Devstral's special-token list, we insert Voxtral-only specials into the next available placeholder slots, producing a merged tokenizer and an assignment map (old Voxtral rank $\rightarrow$ new merged rank). This map is applied consistently to both the token embedding matrix and the LM head to maintain weight/tokenizer correspondence throughout.

4.2 Merge Strategies

We evaluate three merge algorithms, each applied only to the language decoder tensors (the audio stack is never modified):

Linear Interpolation. The simplest approach averages matching decoder tensors between the two models using a mixing coefficient α: $\theta_{\text{merged}} = (1-\alpha) \cdot \theta_{\text{Devstral}} + \alpha \cdot \theta_{\text{Voxtral}}$. For token embeddings and the LM head, we apply a special-aware routine: common specials are averaged when both rows are non-trivial, Voxtral-only specials are copied to their reassigned slots, and non-special tokens are averaged where both exist.

SLERP (Spherical Linear Interpolation). SLERP replaces arithmetic averaging with spherical interpolation to better preserve vector geometry and angular relationships between weight vectors. Decoder weights are SLERPed where shapes match, with fallback to linear interpolation when angles or norms are degenerate. For embeddings and the LM head, audio transcription control tokens

Table 1: Merge evaluation results across all configurations. WER metrics are lower-is-better (↓); accuracy metrics are higher-is-better (↑). SWE-Bench is evaluated on the 500-instance Verified set.

Checkpoint	Devstral		Voxtral			
	SWE-Bench (%) ↑	MMLU Acc. ↑	FLEURS WER ↓	LibriSpeech WER ↓	LLaMA Q. Acc. ↑	MMLU Audio Acc. ↑
Devstral 2 Small	**68**	0.798	–	–	–	–
Voxtral 24B	–	0.74	0.620	**0.015**	0.713	0.743
DELLA (d=0.35, ε=0.12, λ=1.00)	TODO	0.749	1.166	0.879	0.693	0.661
DELLA (d=0.35, ε=0.12, λ=1.05)	TODO	0.736	0.654	0.139	0.633	0.636
DELLA (d=0.35, ε=0.15, λ=1.00)	TODO	0.769	0.780	0.753	0.677	0.673
DELLA (d=0.35, ε=0.15, λ=1.05)	TODO	0.744	0.691	0.138	**0.743**	0.593
DELLA (d=0.50, ε=0.12, λ=1.00)	TODO	0.780	**0.595**	0.079	0.687	0.697
DELLA (d=0.50, ε=0.12, λ=1.05)	TODO	0.777	0.630	0.022	0.690	0.680
DELLA (d=0.50, ε=0.15, λ=1.00)	TODO	0.781	0.609	0.154	0.660	0.692
DELLA (d=0.50, ε=0.15, λ=1.05)	TODO	0.781	0.630	0.552	0.713	0.678
Linear (α=0.10)	61.7	0.806	1.186	1.795	0.670	0.383
Linear (α=0.20)	TODO	0.807	1.100	2.001	0.680	0.457
Linear (α=0.30)	TODO	0.809	0.871	2.700	0.700	0.617
Linear (α=0.40)	TODO	0.812	0.908	3.132	0.717	0.719
Linear (α=0.50)	TODO	**0.813**	0.905	3.297	0.730	**0.763**
SLERP (α=0.10)	TODO	0.805	1.192	2.126	0.663	0.408
SLERP (α=0.20)	TODO	0.807	1.152	2.418	0.680	0.459
SLERP (α=0.30)	TODO	0.808	0.946	2.721	0.690	0.529
SLERP (α=0.40)	TODO	0.810	0.868	3.075	0.700	0.643
SLERP (α=0.50)	TODO	0.810	0.882	3.345	0.713	0.726

are copied verbatim from Voxtral, and Voxtral-only specials are reassigned per the tokenizer map. Non-special rows use SLERP when both sides exist.

DELLA (Drop–Elect–Fuse). DELLA **?** operates on the weight deltas ($\theta_{\text{Voxtral}} - \theta_{\text{Devstral}}$) through three steps: (i) *MagPrune*: magnitude-aware probabilistic pruning controlled by a density parameter (keep-rate) and ϵ (probability window); (ii) *Election*: sign-based conflict resolution to avoid destructive interference; and (iii) *Fusion*: weighted combination with a global λ scale. DELLA is applied to decoder tensors and to embeddings/LM head using the same tokenizer-aware rules. An advanced schedule optionally applies denser retention to MLP layers and more conservative merging in early decoder blocks.

4.3 Merge Evaluation Grid

We construct a grid of merge configurations sweeping the key hyperparameters for each strategy:

- **Linear**: $\alpha \in \{0.1, 0.5\}$
- **SLERP**: $\alpha \in \{0.1, 0.5\}$
- **DELLA**: density, ϵ, λ, and rescale norm (L1/L2/L_∞/per-weight)

Every candidate checkpoint is evaluated on four axes: (i) **MMLU Audio Understanding** (audio comprehension), (ii) **LLaMA Questions S2T** (speech-to-text question answering), (iii) **SWE-Bench Verified** (agentic coding), and (iv) **MMLU Text** (general knowledge preservation). ASR benchmarks (FLEURS, LibriSpeech) are run as diagnostics to gauge transcription preservation but are not primary selection criteria.

4.4 Merge Results and Selection

Table 1 summarizes evaluation results across all merge configurations. Metrics are grouped by origin model: Devstral benchmarks (SWE-Bench Verified and MMLU text) assess coding and general knowledge preservation, while Voxtral benchmarks (FLEURS WER, LibriSpeech WER, LLaMA Questions, MMLU Audio) assess audio capability retention.

Several patterns emerge from the grid search:

Low α is essential for coding preservation. Both linear and SLERP merges with $\alpha = 0.5$ (equal weighting) cause severe degradation in SWE-Bench performance. At $\alpha = 0.1$ (90% Devstral, 10% Voxtral), coding performance is largely preserved. This confirms that the coding capability is sensitive to weight perturbation and requires the merged model to remain close to the Devstral weight manifold.

DELLA underperforms for this setting. Despite being designed to reduce destructive interference through delta sparsification, DELLA configurations consistently fail to produce competitive SWE-Bench scores. The delta-pruning approach appears too aggressive for the cross-modal setting where the two models have diverged substantially from their shared base.

Linear and SLERP perform similarly at low α. At $\alpha = 0.1$, both linear interpolation and SLERP produce comparable results. Given the simplicity of linear interpolation and the marginal difference from SLERP, we select the **linear merge with** $\alpha = 0.1$ as the base checkpoint for subsequent SFT.

4.5 Safeguards and Implementation Details

Across all merge strategies, several safeguards ensure robustness: the audio stack (Whisper encoder and MLP projection) is isolated and never altered; LM head rows are handled identically to token embeddings to maintain tokenizer/weight consistency; existence thresholds prevent averaging with near-zero rows; and shape mismatches are logged and skipped rather than forced. Merged embeddings are written under the Voxtral-compatible key for loader compatibility.

5 Supervised Finetuning

5.1 Motivation

While the linear merge with $\alpha = 0.1$ (90% Devstral, 10% Voxtral) successfully preserved the coding capabilities of Devstral, achieving 307/500 (61.4%) on SWE-Bench Verified, the audio understanding capabilities were substantially degraded. The merged checkpoint scored only 38% on MMLU Audio Understanding, compared to 74.3% for the original Voxtral-24B-Audio. This degradation is expected: the low merge coefficient, while necessary to protect coding performance, dilutes the learned audio representations in the decoder weights.

The goal of the SFT stage was therefore twofold: (1) recover the audio understanding performance of the merged checkpoint toward the Voxtral baseline of $\sim$74% MMLU Audio, and (2) do so without degrading the SWE-Bench coding performance below the merged checkpoint's 61.4%. This required careful construction of audio-augmented coding data and extensive ablation over data mixtures and training hyperparameters.

[TODO:Sid] Add bar chart comparing Devstral SWE-Bench (67%), Voxtral MMLU Audio (74.3%), Merged SWE-Bench (61.4%), Merged MMLU Audio (38%).

5.2 Synthetic Audio Coding Data Curation

No large-scale dataset of audio-input coding instructions exists, so we constructed one synthetically by converting the text-based user turns of Devstral's instruction-tuning trajectories into spoken audio. The pipeline operates on the original multi-turn coding trajectories and proceeds in three stages, illustrated in Figure **??**.

[TODO:Sid] Add data curation pipeline architecture diagram (fig:data_pipeline). Flow: Coding Trajectories $\rightarrow$ LLM Judge $\rightarrow$ Rewrite/TTS $\rightarrow$ Audio+Text Sample.

Stage 1: Verbalizability Judgment. Not all user messages in coding trajectories are suitable for conversion to speech. Messages containing directory trees, file structure listings, long file paths, HTML/markup-heavy content, multi-line code blocks, diffs, stack traces, or non-English text are inherently difficult to verbalize faithfully. We employ an LLM as a judge to classify each user message into one of three categories:

- **DIRECT**: The message is already naturally speakable without modification.

- **REWRITE**: The message can be rewritten into a spoken-friendly version while preserving all substantive details including requirements, constraints, file paths, error messages, and numerical values.
- **NO**: The message is not verbalizable and should remain as text.

The judge is instructed to preserve all semantically important content: code tokens, file paths, and parameter values must remain exactly as written. Only purely syntactic formatting (markdown syntax, bullet-point structure, extraneous whitespace, and emojis) may be removed. Enumerated lists are converted into natural paragraph form. Each judgment is logged as a structured JSON record containing the input text, the decision, and the rewritten spoken text, enabling post-hoc quality auditing of the entire dataset.

[TODO:Sid] Add judge statistics – fraction of messages classified DIRECT / REWRITE / NO from judge logs.

Stage 2: Text-to-Speech Synthesis with Voice Cloning. For messages classified as verbalizable (DIRECT or REWRITE), we synthesize speech using a TTS model with voice-cloning capabilities. A reference audio clip is sampled from a pool of reference recordings to provide the target voice characteristics. The TTS model generates a waveform conditioned on the spoken text and the reference clip. The resulting audio is encoded as base64 WAV and paired with a transcription field in the training sample, replacing the original text content of that user turn.

Stage 3: Assembly and Quality Control. The pipeline supports row-wise sharding across multiple GPU workers for parallel processing, with atomic file-path reservation to prevent overwrites in concurrent execution. A resume mechanism tracks the last successfully processed row per shard, enabling fault-tolerant restarts without reprocessing. Messages that fail TTS synthesis are retained best-effort with their original text, ensuring no training samples are silently dropped. The final output preserves the original conversation structure (system prompts, assistant responses with tool calls, and available tool definitions) while selectively replacing user text turns with audio.

The resulting dataset, referred to as *Devstral Audio Instruct*, contains audio-augmented coding trajectories derived from the same SWE-agent rollout distributions used in the original Devstral instruction tuning.

6 Results

6.1 Training Configuration

All SFT experiments were conducted on 128 H100 GPUs with the following base configuration: AdamW optimizer with learning rate 1×10^{-6}, weight decay 0.8, 100 warmup steps, minimum learning rate ratio 0.1, gradient clipping at max norm 20, sequence length 131,072, and global batch size of $\sim$4M tokens. The Whisper-Large-V3 audio encoder was frozen throughout training, with audio embeddings projected via an MLP adapter.

A critical constraint of the multimodal architecture is the parallelism configuration. The Whisper encoder's 20 attention heads require model parallelism to be set to a divisor of 20; we use model parallel $= 4$. Context parallelism, which Devstral's text-only training leverages, cannot be used because the Whisper encoder does not support it, so context parallel is set to 1.

[TODO:Sid] Add hyperparameter table (tab:hyperparams).

Hyperparameter Discovery. Initial SFT experiments used hyperparameters that were a hybrid of the best configurations from Devstral and Voxtral training. These experiments consistently degraded SWE-Bench performance, in some cases reducing it to $\sim 50\%$. Through systematic ablation, we identified two root causes:

1. **Missing YaRN configuration.** The merged checkpoint's metadata did not include YaRN (Yet another RoPE extensioN) parameters, leading us to initially train without positional extrapolation. However, Devstral was in fact trained with YaRN (factor $= 48$, $\beta = 32$, $\alpha = 1$, $\theta_{\text{rope}} = 10^8$). Re-enabling YaRN in the training configuration produced immediate improvements in downstream performance.

Table 2: SFT data mixture ablations. SWE-Bench is evaluated on the 500-instance Verified set unless marked with † (50-instance proxy). MMLU Audio is the audio understanding match score (%). The base merged checkpoint scores 61.4% SWE-Bench and 38% MMLU Audio.

Data Mixture	Text : Audio	SWE-Bench (%)	MMLU Audio (%)
Devstral Text Traj. only	100 : 0	**63.4**	66.0
Devstral Text Traj. + Devstral Audio Traj. (high)	67 : 33	$\sim 50^{\dagger}$	74–75
Devstral Text Traj. + Voxtral Audio Instruct + Devstral Audio Traj.	–	diverged	–
Devstral Audio Traj. only	0 : 100	$2^{\dagger}$	–
Devstral Text Traj. + Devstral Audio Traj. (moderate)	80 : 20	60.0	73.9
Devstral Text Traj. + Devstral Audio Traj. (low)	90 : 10	–	**75.5**
Devstral Text Traj. + Devstral Audio Traj. (minimal)	95 : 5	**67.0**	75.0

2. **Hyperparameter alignment with Devstral.** Configurations using Voxtral-style hyperparameters led to significant SWE-Bench degradation. Aligning the training recipe closely with the original Devstral SFT configuration, particularly the optimizer settings, learning rate, and weight decay, was essential for preserving coding capabilities.

6.2 Data Mixture Ablations

We conducted extensive ablations over data composition to find the optimal trade-off between audio recovery and coding preservation. Table 2 summarizes the key experiments.

Text-Only Replay. As a control experiment, we first fine-tuned the merged checkpoint using only Devstral Text Trajectories to verify that the training recipe does not degrade coding performance. This configuration achieved 317/500 (63.4%) on SWE-Bench, a 2-point improvement over the base merged checkpoint. Remarkably, MMLU Audio also improved from 38% to $\sim 66\%$ *without any audio instruct data*. This suggests that text-only SFT partially re-activates the audio pathways disrupted during weight merging, a finding we discuss further in Section **??**.

Audio-Only. Training exclusively on Devstral Audio Trajectories resulted in catastrophic forgetting of coding capabilities: SWE-Bench collapsed to $\sim 2\%$ on the proxy evaluation set. This confirms that text instruct data is essential for maintaining coding performance. We additionally verified that finite versus infinite dataloader configurations have no measurable effect on this outcome.

High Audio Ratio. With $\sim 33\%$ audio data, MMLU Audio recovered to 74–75%, closely matching or exceeding the Voxtral baseline. However, SWE-Bench dropped to $\sim 50\%$. A configuration that combined the full Devstral Text Trajectories with all Voxtral Audio Instruct data and Devstral Audio Trajectories failed to converge entirely: the training loss never stabilized and SWE-Bench evaluation entered infinite generation loops, indicating severe model degradation.

Moderate Audio Ratio. Reducing the audio proportion to $\sim 20\%$ produced intermediate results. We evaluated multiple intermediate checkpoints from this run and observed a clear trade-off pattern:

- At early checkpoints: SWE-Bench 300/500 (60.0%), MMLU Audio 73.9%
- At mid-training: SWE-Bench 288/500 (57.6%), MMLU Audio 75.0%
- At final checkpoint: SWE-Bench 284/500 (56.8%), MMLU Audio 74.0%

MMLU Audio recovers rapidly within the first few dozen training steps and then plateaus, while SWE-Bench progressively degrades with continued training. This indicates that the model requires very few audio gradient updates to recover audio understanding, but extended exposure to audio data erodes coding capabilities.

Table 3: Best scores across all checkpoints for each SFT configuration. All configurations use the Devstral Text + Audio data mixture. Best per column in **bold**.

α	Audio Epochs	MMLU Audio (%) ↑
0.1	1.0	75.93
0.1	0.5	75.52
0.1	0.25	75.00
0.5	0	**78.72**
0.5	1.0	75.72

[TODO:Sid] Add training curves figure for 80:20 config (MMLU Audio + LLaMA S2T vs steps). Use dextral_sweep_linear_20_try2.png.

Low and Minimal Audio Ratio: Best Results. Based on the observation that audio recovery saturates quickly while coding degradation is cumulative, we reduced the audio fraction further. With 10% audio data, MMLU Audio reached 75.5%, *surpassing* the original Voxtral baseline of 74.3%. With only 5% audio data, MMLU Audio still reached 75.0%, and the best intermediate checkpoint achieved **67.0%** on SWE-Bench Verified, matching the original Devstral-2 baseline.

[TODO:Sid] Add side-by-side training curves for 90:10 and 95:5 configs. Use dextral_sweep_linear_21.png and dextral_sweep_linear_22.png.

6.3 SFT Hyperparameter Experiments

To further optimize the SFT recipe beyond data mixture ratios, we conducted systematic experiments across two axes: (1) the merge coefficient α of the base checkpoint (0.1 vs 0.5) and (2) audio data epoch length (0.25, 0.5, and 1.0 epochs). An additional configuration tested the effect of training schedule refinements (increased buffer size and warmup steps). Each configuration was evaluated at 10 intermediate checkpoints. Table 3 reports the best score achieved at any checkpoint within each configuration.

Merge coefficient dominates the coding–audio tradeoff. The α=0.5 text-only baseline reaches the highest MMLU Audio at 78.72%, demonstrating that the higher Voxtral weight in the merge carries significant audio capability that SFT can surface even without audio training data. This confirms that α is the primary lever controlling the coding–audio balance, with SFT providing fine-grained recovery on top.

Audio epoch length has diminishing returns. Within α=0.1 configurations, reducing the audio data from 1.0 to 0.25 epochs has minimal impact on MMLU Audio (75.93% vs 75.00%). This is consistent with the observation from data mixture ablations: audio understanding recovers rapidly within the first few gradient updates, and further audio exposure primarily erodes coding performance.

Training schedule refinements. The final configuration (α=0.1, buffer $=$ 16, warmup $=$ 10) maintained competitive MMLU Audio (75.21%). The increased buffer size and warmup steps appear to stabilize early training and improve downstream performance.

[TODO:Sid] Add Pareto frontier figure: SWE-Bench vs MMLU Audio, one point per configuration's best checkpoint, colored by α. This would visually summarize the coding–audio tradeoff across all configurations.

6.4 Final Model Performance

Table 4 compares the best SFT checkpoint (95:5 text-to-audio ratio, early checkpoint) against both parent models and the base merged checkpoint.

[TODO:Sid] Fill in MMLU Text and LLaMA Q. S2T for SFT best ckpt.

The final model recovers SWE-Bench to 67% (matching Devstral-2) and MMLU Audio to 74.0%, nearly matching Voxtral-24B-Audio's 74.3%. This demonstrates that the merge-then-SFT pipeline can unify complementary modality-specific capabilities with minimal regression on either.

Table 4: Comparison of the final SFT model against parent models and the base merged checkpoint. ↑: higher is better; ↓: lower is better. Dashes indicate the metric is not applicable to that model.

Model	SWE-Bench Verified (%)↑	MMLU Audio (%)↑	MMLU Text (%)↑	LLaMA Q. S2T↑
Devstral 2 (text only)	**68.0**	–	**79.8**	–
Voxtral 24B	–	74.3	74.3	**0.713**
Merged (α=0.1)	61.4	38.0	–	0.670
+ SFT (best ckpt)	67.0	**74.0**	–	0.660

7 Analysis

Several noteworthy findings emerged from our experiments:

Audio recovery without audio data. The text-only replay experiment improved MMLU Audio from 38% to 66% despite containing no audio training samples whatsoever. This suggests that audio understanding capability is latently preserved in the merged weights but is "misaligned" by the merging process. Standard text SFT partially restores this alignment, possibly by recovering attention patterns in the shared decoder layers that properly route audio encoder outputs. This finding implies that merging preserves more of the constituent models' capabilities than raw evaluation numbers initially suggest.

Super-parent performance. Certain merged and fine-tuned checkpoints exceeded the performance of *both* parent models on individual benchmarks. Some configurations achieved MMLU Text scores above Devstral's 79.8%, and SFT experiments on the α=0.5 base checkpoint reached MMLU Audio scores of up to 78.72%, substantially surpassing Voxtral's baseline of 74.3%. Even on the α=0.1 base checkpoint, multiple SFT runs achieved MMLU Audio scores of 75–76%, exceeding the parent Voxtral model. These improvements are consistent across multiple runs and suggest that model merging can produce complementary feature interactions that benefit both modalities, consistent with prior observations of "synergistic" effects in the model merging literature **??**.

[TODO:Sid] Add exact MMLU Text numbers for merged ckpts that exceeded 79.8% from xlsx data.

Rapid audio saturation and the data efficiency of recovery. Across all configurations containing audio data, MMLU Audio recovers to $\geq$73% within the first $\sim$50 training steps and plateaus thereafter, regardless of the audio data fraction (5%, 10%, or 20%). This rapid saturation implies that very few gradient updates on audio-augmented data suffice to re-activate the dormant audio pathways. The practical implication is that lightweight adaptation strategies such as LoRA applied only to the audio projection layers may be sufficient for audio capability recovery, offering a more parameter-efficient alternative to full SFT.

[TODO:Sid] Add composite overlay plot of MMLU Audio recovery curves across text-only, 80:20, 90:10, 95:5 on same axes. Generate new plot.

Catastrophic forgetting is asymmetric. Coding performance degrades gradually and monotonically with increased audio training exposure, while audio performance recovers almost instantaneously. This asymmetry suggests that the coding capability is distributed across many decoder parameters and is sensitive to weight perturbation, whereas the audio capability is more localized (likely concentrated in the Whisper adapter MLP and a small subset of decoder attention heads) and can be recovered with minimal parameter updates.

[TODO:Sid] Add SWE-Bench vs MMLU Audio Pareto scatter plot across all checkpoints. Generate new plot.

[TODO:Sid] Consider adding a WandB training loss curve for one representative run.

8 Conclusion

We presented Dextral, a 24B-parameter model that unifies audio understanding and agentic coding within a single checkpoint. By merging the decoder weights of Voxtral Small and Devstral Small via linear interpolation ($\alpha = 0.1$) and performing targeted SFT with a 95:5 text-to-audio data ratio, we recover both parent models' capabilities: 67% on SWE-Bench Verified (matching Devstral) and 74.0% on MMLU Audio (nearly matching Voxtral's 74.3%). SFT experiments on higher-α merges achieved up to 78.72% MMLU Audio, demonstrating super-parent performance on audio understanding.

Our experiments reveal several practical insights for cross-modal model merging. First, coding capabilities are sensitive to weight perturbation and require the merged model to remain close to the coding-specialist manifold. Second, audio understanding recovers rapidly with minimal data, suggesting that merged weights retain latent modality-specific knowledge that can be re-activated through lightweight fine-tuning. Third, text-only SFT alone recovers a substantial portion of audio capability (38% $\rightarrow$ 66%), implying that the merging process preserves more information than immediate post-merge evaluations suggest.

These findings point to model merging as a viable, low-compute strategy for constructing multimodal systems from specialist models, particularly when the constituents share a common architectural backbone. Future work includes exploring parameter-efficient adaptation (e.g., LoRA) as an alternative to full SFT for audio recovery, extending the approach to additional modalities, and investigating the mechanisms by which merged weights retain and recover latent capabilities.

Core contributors

Contributors

Abhinav Rastogi, Adam Yang, Albert Q. Jiang, Alexandre Sablayrolles, Amélie Héliou, Amélie Martin, Anmol Agarwal, Antoine Roux, Arthur Darcet, Arthur Mensch, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Chris Bamford, Christian Wallenwein, Christophe Renaudin, Clémence Lanfranchi, Darius Dabert, Devendra Singh Chaplot, Devon Mizelle, Diego de las Casas, Elliot Chane-Sane, Emilien Fugier, Emma Bou Hanna, Gabrielle Berrada, Gauthier Delerce, Gauthier Guinet, Georgii Novikov, Guillaume Martin, Himanshu Jaju, Jan Ludziejewski, Jason Rute, Jean-Hadrien Chabran, Jessica Chudnovsky, Joachim Studnia, Joep Barmentlo, Jonas Amar, Josselin Somerville Roberts, Julien Denize, Karan Saxena, Karmesh Yadav, Kartik Khandelwal, Kush Jain, Lélio Renard Lavaud, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Marie Pellat, Mathilde Guillaumin, Mathis Felardos, Matthieu Dinot, Maxime Darrin, Maximilian Augustin, Mickaël Seznec, Neha Gupta, Nikhil Raghuraman, Olivier Duchenne, Patricia Wang, Patryk Saffer, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Rémi Delacourt, Romain Sauvestre, Roman Soletskyi, Sagar Vaze, Sandeep Subramanian, Saurabh Garg, Shashwat Dalal, Siddharth Gandhi, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Thibault Schueller, Thibaut Lavril, Thomas Robert, Thomas Wang, Timothée Lacroix, Tom Bewley, Valeriia Nemychnikova, Victor Paltz, Virgile Richard, Wen-Ding Li, William Marshall, Xuanyu Zhang, Yihan Wan, Yunhao Tang

References