

60-second takeaway

Six open-source TTS models dominate the 2026 fine-tuning landscape. They look similar on paper - most do voice cloning, most fit on 24GB, most produce good output. But they use fundamentally different architectures, and those differences determine which fine-tuning approach works, which LoRA framework you need, how long data preprocessing takes, and whether you can deploy commercially. We fine-tuned five of these models on the same single-speaker corpus ([IMDANSC FEMALE_01](#)) and analyzed the sixth from its paper and model code. This article explains what the architectures tell you about fine-tuning - before you commit GPU hours.

Who this is for

- **Engineers choosing a TTS model to fine-tune:** You have read the benchmarks. You know which models exist. You need to understand the architectural differences before committing to a fine-tuning approach - because picking the wrong approach wastes days, not hours.
- **ML engineers adding TTS to a pipeline:** You need to know which models wrap with standard PEFT, which need custom LoRA libraries, and which only support full SFT - before you design your training infrastructure.
- **Technical leads evaluating licenses:** Two of these six models have license restrictions that are not obvious from their GitHub repos. This article flags them before you build on top of them.

The six models

Model	Total params	LLM backbone	Released	License
Voxtral 4B	~4.1B	Ministral-3B	March 2026	CC BY-NC 4.0
Qwen3-TTS 1.7B	1.7B	Qwen3	January 2026	Apache-2.0
IndexTTS2	~560M	Custom GPT-2	June 2025	Apache-2.0
Chatterbox	500M	Llama 3-inspired	June 2025	MIT

Fish Speech S2 Pro	~5B	Qwen3-4B	March 2026	Fish Audio Research License
CosyVoice 3	~800M	Qwen2-0.5B	May 2025	Apache-2.0

These are the models with active fine-tuning ecosystems or clear fine-tuning potential as of March 2026. Models where upstream already provides complete fine-tuning tooling (VoxCPM 1.5, GPT-SoVITS v4) are excluded - you do not need a guide for those.

Three generation paradigms

Every model in this list generates speech through a multi-stage pipeline. The stages differ, and that difference determines your fine-tuning strategy. Three paradigms cover all six models.

Paradigm A: Autoregressive backbone → flow-matching decoder → vocoder

Models: Voxtral 4B, Chatterbox, CosyVoice 3

Text → LLM predicts speech tokens (autoregressive)
 → Flow-matching model converts tokens to mel/acoustic representation
 → Vocoder synthesizes waveform

The LLM backbone is the primary fine-tuning target. The flow-matching decoder and vocoder are typically frozen - they handle acoustic rendering, not linguistic content or speaker identity.

Why this matters for fine-tuning: You only need to LoRA or SFT the backbone. The flow-matching decoder (100–390M parameters depending on the model) and vocoder are left untouched. This makes fine-tuning cheaper than the total parameter count suggests - Voxtral is 4.1B total, but you only fine-tune the 3.4B backbone.

Voxtral specifics: The backbone predicts one semantic token per frame from an 8,192-entry codebook. The 390M flow-matching transformer then runs 8 Euler steps per frame to produce 36 acoustic values. The paper explicitly states: freeze text-embedding layers during fine-tuning to prevent overfitting on rare vocabulary. The 300M Voxtral Codec is both the tokenizer and the vocoder - it encodes reference audio into tokens and decodes generated tokens back to 24 kHz waveforms.

Chatterbox specifics: The T3 component (text-to-speech-token, Llama-based) is the fine-tuning target. S3Gen (flow-matching) and the HiFT-GAN vocoder are frozen.

The S3Tokenizer is shared with CosyVoice 2 - data prepared for one codec family partially transfers.

CosyVoice 3 specifics: The Qwen2 backbone predicts speech tokens from a 6,561-entry vocabulary. The DiT flow-matching decoder $depth = 22$, $dim = 1024$, $300Mparams$ converts tokens to 80-band mel spectrograms. A causal HiFiGAN variant produces 24 kHz audio.

Paradigm B: Multi-codebook autoregressive → codec decoder (no separate flow stage)

Models: Qwen3-TTS, Fish Speech S2 Pro

Text → AR model predicts all codebook layers (semantic + residual)
→ Codec decoder reconstructs waveform directly from tokens

There is no separate flow-matching or diffusion stage. The autoregressive model handles both semantic and acoustic token prediction. A lightweight codec decoder (not a full vocoder) converts tokens to audio.

Why this matters for fine-tuning: One LoRA pass covers the entire generation pipeline. You do not need to decide whether to also fine-tune a separate flow model. But the model is doing more work per forward pass (predicting multiple codebook layers), which affects training dynamics.

Qwen3-TTS specifics: A dual-track architecture - the talker predicts codebook-0 (semantic layer) autoregressively, then an MTP (Multi-Token Prediction) sub-talker predicts 15 residual codebooks simultaneously at each position. Total: 16 codebook layers at 12.5 Hz. The model registers as `Qwen3TTSForConditionalGeneration` - standard HuggingFace PEFT wrapping works natively. This is the simplest LoRA integration of any model in this list.

Fish Speech S2 Pro specifics: DualAR architecture - a 4B Slow AR (built on Qwen3-4B) predicts codebook-0 at each timestep, then a 400M Fast AR (4-layer shallow transformer) predicts codebooks 1–9 depth-wise. The Slow AR hidden state conditions the Fast AR. This is structurally isomorphic to a standard causal LM, so it inherits LLM serving optimizations (KV-cache, speculative decoding). However, LoRA uses a custom `loralib` implementation, not HuggingFace PEFT.

Paradigm C: Autoregressive backbone → DiT mel predictor → vocoder

Models: IndexTTS2

- Text → GPT predicts semantic codes (autoregressive)
 - DiT converts semantic codes to mel spectrogram (non-autoregressive)
 - BigVGAN synthesizes waveform

The GPT backbone is the fine-tuning target. The S2M DiT (Semantic-to-Mel Diffusion Transformer) and BigVGAN vocoder are frozen.

Why this matters for fine-tuning: This is the only model that predicts semantic codes and then uses a separate non-autoregressive model to produce mel spectrograms. The DiT adds quality but also adds a stage that cannot be LoRA'd with standard tools - the GPT backbone is a custom GPT-2, not a HuggingFace PreTrainedModel. Full SFT is the only fine-tuning path.

IndexTTS2 specifics: The GPT backbone is ~560M parameters (24 layers, dim 1280, 20 heads) - significantly smaller than often cited. It uses a MaskGCT semantic codec (SeamlessM4T → Wav2Vec2Bert → RepCodec) with an 8,192-entry codebook. Training loss is weighted: 0.2 text cross-entropy + 0.8 mel cross-entropy.

Paradigm summary

Paradigm	Models	Fine-tuning target	What's frozen	LoRA possible?
A: AR + flow-matching + vocoder	Voxtral, Chatterbox, CosyVoice 3	LLM backbone	Flow model + vocoder	Yes (backbone only)
B: Multi-codebook AR + codec	Qwen3-TTS, Fish Speech S2 Pro	Full AR model	Codec encoder/decoder	Yes (single LoRA pass)
C: AR + DiT + vocoder	IndexTTS2	GPT backbone	DiT + BigVGAN	No (full SFT only)

Codec and tokenizer comparison

The codec determines what the model is actually learning to predict - and misunderstanding this is the fastest way to waste a fine-tuning run.

Model	Codec	Token rate	Semantic vocab	Tokens per frame	Output sample rate
-------	-------	------------	----------------	------------------	--------------------

Voxtral 4B	Voxtral Codec (VQ + FSQ hybrid)	12.5 Hz	8,192	37 (1 semantic + 36 acoustic)	24 kHz
Qwen3-TTS	Custom RVQ (16 codebooks)	12.5 Hz	2,048 per layer	16	24 kHz
IndexTTS2	MaskGCT semantic (RepCodec)	~25 Hz	8,192	1	22 kHz
Chatterbox	S3Tokenizer (FSQ)	25 Hz	6,561	1	24 kHz
Fish Speech S2 Pro	Modified DAC (RVQ)	~21 Hz	4,096	10 (1 slow + 9 fast)	44.1 kHz
CosyVoice 3	FSQ-MinMo (multi-task)	25 Hz	6,561	1	24 kHz

What the token rate tells you

Lower token rate (12.5 Hz) means fewer tokens per second of audio, which means shorter training sequences and faster training per sample. Higher token rate (25 Hz) means more temporal resolution but longer sequences.

Voxtral and Qwen3-TTS at 12.5 Hz produce half as many tokens per second as IndexTTS2 or Chatterbox at 25 Hz. For a 10-second audio clip, that is 125 tokens vs 250 tokens - a 2x difference in sequence length that directly affects training memory and speed.

What the tokens-per-frame count tells you

Voxtral predicts 37 tokens per frame (1 semantic + 36 acoustic). Qwen3-TTS predicts 16 tokens per frame. Chatterbox and CosyVoice 3 predict just 1 token per frame.

More tokens per frame means more information is being generated per timestep, which generally means higher audio quality but also more complex training dynamics. Models with 1 token per frame rely on a separate flow-matching or DiT stage to reconstruct the acoustic details.

The sample rate trap

Five of six models operate at 22–24 kHz. Fish Speech S2 Pro operates at **44.1 kHz**. If you switch between models without checking sample rate, training may appear to work - loss decreases normally - but output quality degrades silently. This was the #1

silent failure across our fine-tuning runs. Always verify sample rate before every training run:

```
# Check before training - every time
ffprobe -v error -show_entries stream=sample_rate -of csv=p=0 your_audio.wav
# Resample if needed
ffmpeg -i input.wav -ar 24000 output_24k.wav # For most models
ffmpeg -i input.wav -ar 44100 output_44k.wav # For Fish Speech
```

Codec family relationships

Chatterbox's S3Tokenizer is derived from the same family as CosyVoice 2's S3TokenizerV2. CosyVoice 3 upgraded to a new FSQ-MinMo tokenizer trained on multi-task objectives (ASR, speech emotion recognition, language ID, audio event detection). This means CosyVoice 3 tokens carry richer information than CosyVoice 2 tokens - prosody, emotion, and speaker style are encoded, not just phonemic content.

LoRA compatibility: why some models wrap with PEFT and others don't

Not all LoRA is the same. The architecture of the backbone determines which LoRA framework works - and getting this wrong means your training script crashes before the first step.

Model	LoRA framework	Why	PEFT get_peft_model() works?
Qwen3-TTS	HuggingFace PEFT	Talker is a standard Qwen3 transformer decoder	Yes - native
Voxtral 4B	HuggingFace PEFT (expected)	Ministral backbone is HF-compatible	Likely yes - untested
CosyVoice 3	HuggingFace PEFT (on Qwen2)	Qwen2 backbone is HF-compatible	Yes for backbone, not for DiT
Chatterbox	HuggingFace PEFT or custom	Llama-based T3 component	Partial - community repos exist
Fish Speech S2 Pro	Custom loralib	Not PEFT - uses setup_lora() that replaces nn.Linear layers	No

IndexTTS2	None	Custom GPT-2, not a PreTrainedModel	No - full SFT only
------------------	------	--	---------------------------

The PEFT compatibility rule

If the backbone inherits from a HuggingFace PreTrainedModel (Qwen2, Qwen3, Llama, Ministral), standard PEFT LoRA wrapping works. If the backbone is custom (IndexTTS2's GPT-2, Fish Speech's DualAR), you either need the upstream's custom LoRA implementation or you are limited to full SFT.

This is an architectural constraint, not a library limitation. Fish Speech's DualAR uses `loralib` (a separate library from PEFT) with a custom `setup_lora()` function that walks the model and replaces `nn.Linear` layers. You cannot swap in PEFT without rewriting the injection logic.

LoRA target modules

For models where LoRA works, the target modules follow the backbone architecture:

Model	Target modules	Rank	Alpha
Qwen3-TTS	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj	16	32
Voxtral 4B	qkv_proj, o_proj, gate_up_proj, down_proj (from Voxtral Mini ASR)	32	32
Fish Speech S2 Pro	attention, mlp, embeddings, output (both Slow + Fast AR)	8	16
CosyVoice 3	Qwen2 attention + MLP layers	TBD	TBD

Qwen3-TTS and Voxtral target the standard attention projections and MLP gate/up/down projections - the same pattern as any Qwen or Mistral language model. Fish Speech targets broader module categories because its custom `loralib` uses string matching on module names rather than explicit layer references.

Data pipeline complexity

This is where the models diverge most dramatically. Preprocessing time ranges from minutes to over 30 minutes for the same dataset - and skipping a step does not produce an error, it produces bad output.

Rank	Model	Preprocessing steps	Time for ~7h audio	Complexity
1 (simplest)	Chatterbox	wav + txt → auto-resamples, caches embeddings	Minutes	Low
2	CosyVoice 3	wav → CAMPPlus embedding + FSQ tokenization → Parquet	~10 min	Medium
3	Qwen3-TTS	wav → 16-layer RVQ codec extraction → JSONL	~15 min	Medium
4	Voxtral 4B	wav → Voxtral Codec encoding (semantic + acoustic)	TBD	Medium (estimated)
5	Fish Speech S2 Pro	wav → DAC VQ extraction → protobuf	22 min (VQ bottleneck)	High
6 (most complex)	IndexTTS2	wav → SeamlessM4T → RepCodec → conditioning → prompt pairs → .npy	30+ min (5 stages)	Very high

What the preprocessing tells you about iteration speed

If you are running multiple fine-tuning experiments (different LR, different epoch counts, different dataset subsets), preprocessing time compounds. Chatterbox lets you run 30 experiments in the time IndexTTS2 takes to preprocess once. This is why Chatterbox is the recommended model for exploration and rapid prototyping, even if your final deployment target is a larger model.

Data format summary

Model	Input	Training format
Chatterbox	wav (16 kHz auto-resampled) + txt	LJSpeech format or wav+txt pairs
CosyVoice 3	wav (24 kHz) + text + instruct file	Parquet (1000 utterances per file)
Qwen3-TTS	wav (24 kHz) + transcript	JSONL with pre-extracted 16-layer codec codes
Voxtral 4B	wav (24 kHz) + transcript	TBD
Fish Speech S2 Pro	wav (44.1 kHz) + .lab transcript	Protobuf dataset

IndexTTS2 wav (24 kHz resampled) + transcript JSONL manifest + .npy feature files

CosyVoice 3 uniquely requires an `instruct` file in addition to standard fields - this is new vs CosyVoice 2 and is a common source of configuration errors when adapting CosyVoice 2 recipes.

What we learned fine-tuning on the same corpus

We fine-tuned five of these six models on [IMDA NSC FEMALE_01](#) - a single-speaker dataset with natural Singaporean English accent - using an RTX 3090 Ti (24GB). Voxtral is architecture-analyzed only (the model is 4 days old as of this writing).

Model	Fine-tuning type	Steps/epochs completed	Best checkpoint	Production-ready?
Qwen3-TTS	LoRA	17 epochs (2,950 steps)	Epoch 10	Yes
IndexTTS2	Full SFT	15,949 steps	Step 14,000	Yes
CosyVoice 3	LoRA (custom scripts)	5,800+ steps	Rerun pending	Not yet
Chatterbox	SFT	512 steps (2 min 20s)	Quality eval pending	TBD
Fish Speech S2 Pro	LoRA (pilot)	64 steps	Pilot only	Not evaluated
Voxtral 4B	-	-	-	Fine-tuning run planned

Three patterns that appeared across every model

1. The best checkpoint is never the last one.

VoxCPM peaked at step 4,000 (not 8,999). IndexTTS2 peaked at step 14,000 (not 15,949). Qwen3-TTS peaked at epoch 10 (not 17). TTS models overfit to training prosody quickly - the last checkpoint has the lowest training loss but not the best perceptual quality. Keep all checkpoints. Evaluate by listening, not by loss curve.

2. Sample rate mismatches are the #1 silent failure.

Qwen3-TTS requires 24 kHz. Fish Speech requires 44.1 kHz. IndexTTS2 resamples input to 24 kHz but outputs at 22 kHz. If you switch between models without checking, training appears to work (loss decreases normally) but output quality degrades. We hit this on three separate occasions before making sample rate verification the first step of every run.

3. Default hyperparameters are wrong for single-speaker fine-tuning.

Qwen3-TTS default LR ($2e-5$) produces noise - use $2e-6$. Qwen3-TTS default LoRA scale (1.0) over-steers - use 0.3–0.35. These are not edge cases. No model's defaults are tuned for fine-tuning on a small single-speaker corpus. Always sweep before committing GPU hours.

Model-specific findings

Qwen3-TTS had a double label-shift bug in the upstream training script (`sft_12hz.py`) - the script manually shifts inputs/labels before calling the model, but `ForCausalLM` in HuggingFace Transformers also shifts internally. This causes the model to learn predictions two positions ahead, which manifests as progressively faster speech with each training epoch. We fixed this in our [companion repo](#) and submitted upstream PR #178.

IndexTTS2 required explicit checkpoint retention management - the default policy deletes older checkpoints before you can evaluate them. The best checkpoint (step 14,000) was not the final step (15,949). We wrote and open-sourced the entire fine-tuning pipeline since the official repo provides inference only: [instavar/indextts2-finetuning](#).

CosyVoice 3 had upstream configuration bugs (GitHub issue #1680) - the `speech_token_size` in the yaml was stale from CosyVoice 2 (6564 vs 6561), causing shape mismatches during LLM training. Our first run also showed checkpoint drift and long-text instability, which we attribute to training only the LLM while keeping the flow model frozen - the LLM learns new token distributions that the frozen DiT has not seen.

Chatterbox trained in 2 minutes 20 seconds (512 samples) - the fastest of any model. But a pilot with only 64 samples produced `loss=0.0`, meaning no learning occurred. The minimum viable dataset is approximately 256 samples.

Fish Speech S2 Pro required careful protobuf version management - the DAC dependency needs `protobuf<5`, which conflicts with most modern ML environments.

VQ extraction for ~7 hours of audio took 22 minutes, making it the second-slowest preprocessing pipeline after IndexTTS2.

License matrix: what you can actually deploy

Two of these six models have license restrictions that are easy to miss. Getting this wrong after building a fine-tuning pipeline is expensive.

Model	Weights license	Commercial fine-tuned weights?	Fine-tuning code
Qwen3-TTS	Apache-2.0	Yes	Apache-2.0
IndexTTS2	Apache-2.0	Yes	Apache-2.0
Chatterbox	MIT	Yes	Apache-2.0
CosyVoice 3	Apache-2.0	Yes	Apache-2.0
Voxtral 4B	CC BY-NC 4.0	No - requires separate Mistral commercial license	Apache-2.0 (ours)
Fish Speech S2 Pro	Fish Audio Research License	No - requires separate Fish Audio commercial license	Apache-2.0 (ours)

If you need commercial deployment, your options are: Qwen3-TTS, IndexTTS2, Chatterbox, or CosyVoice 3. All four are fully permissive.

If you are doing research or non-commercial work, all six models are available without restriction.

The distinction matters because fine-tuning code (which we release under Apache-2.0) is separate from model weights. You can legally use our fine-tuning tools on Voxtral or Fish Speech for research - you just cannot deploy the resulting weights commercially without a separate license from Mistral or Fish Audio respectively.

Architecture → fine-tuning strategy decision matrix

Your situation	Paradigm	Recommended model	Fine-tuning approach	Data prep time
Need LoRA with standard PEFT	B	Qwen3-TTS	<code>peft.get_peft_model()</code> on Qwen3 talker	~15 min

Need full SFT for maximum fidelity	C	IndexTTS2	Full-weight training on GPT backbone	30+ min
Need fastest possible iteration	A	Chatterbox	SFT on T3 component	Minutes
Need flow-matching quality	A	CosyVoice 3	LoRA on Qwen2 backbone (custom scripts)	~10 min
Need multilingual (EN+CN+JP)	B	Fish Speech S2 Pro	LoRA via custom <code>loralib</code>	22+ min
Want first-mover on newest model	A	Voxtral 4B	LoRA on Ministral backbone (not yet implemented)	TBD
Need commercial deployment	A/B/C	Qwen3-TTS, IndexTTS2, Chatterbox, or CosyVoice 3	Depends on model	Varies

Papers and further reading

Model	Paper	arXiv
Voxtral 4B	Voxtral TTS	2603.25551
Qwen3-TTS	Qwen3-TTS Technical Report	2601.15621
IndexTTS2	IndexTTS 2	2506.21619
Chatterbox	No paper (Resemble AI blog + benchmark) -	
Fish Speech S2 Pro	Fish Audio S2 Technical Report	2603.08823
CosyVoice 3	CosyVoice 3: Scaling-up and Post-training	2505.17589

Related guides

This article explains *why* the models differ architecturally. For *how* to fine-tune each one, see the per-model walkthroughs:

- [Which TTS Model Should You Use? A Decision Tree \(2026\)](#) - use-case routing for all 9 models

- [Best Open-Source TTS Models for Production in 2026](#) - benchmark results on IMDA NSC FEMALE_01
- [LoRA Finetuning Qwen3-TTS for Custom Voices](#) - Qwen3-TTS fine-tuning walkthrough + [companion repo](#)
- [IndexTTS2 Finetuning on IMDA NSC FEMALE_01](#) - full SFT guide + [companion repo](#)
- [CosyVoice 2 vs CosyVoice 3 on IMDA NSC FEMALE_01](#) - generation comparison
- [Voice Cloning on a 24GB GPU - What Actually Works in 2026](#) - VRAM profiles and training times

Last updated: March 30, 2026. This page is updated as we complete fine-tuning runs on additional models. Voxtral 4B and Fish Speech S2 Pro sections will be expanded with first-party data when available.