Voxtral TTS

Abstract

We introduce Voxtral TTS, an expressive multilingual text-to-speech model that generates natural speech from as little as 3 seconds of reference audio. Voxtral TTS adopts a hybrid architecture that combines auto-regressive generation of semantic speech tokens with flow-matching for acoustic tokens. These tokens are encoded and decoded with Voxtral Codec, a speech tokenizer trained from scratch with a hybrid VQ-FSQ quantization scheme. In human evaluations conducted by native speakers, Voxtral TTS is preferred for multilingual voice cloning due to its naturalness and expressivity, achieving a 68.4% win rate over ElevenLabs Flash v2.5. We release the model weights under a CC BY-NC license.

Webpage: https://mistral.ai/news/voxtral-tts
Model weights: https://huggingface.co/mistralai/Voxtral-4B-TTS-2603

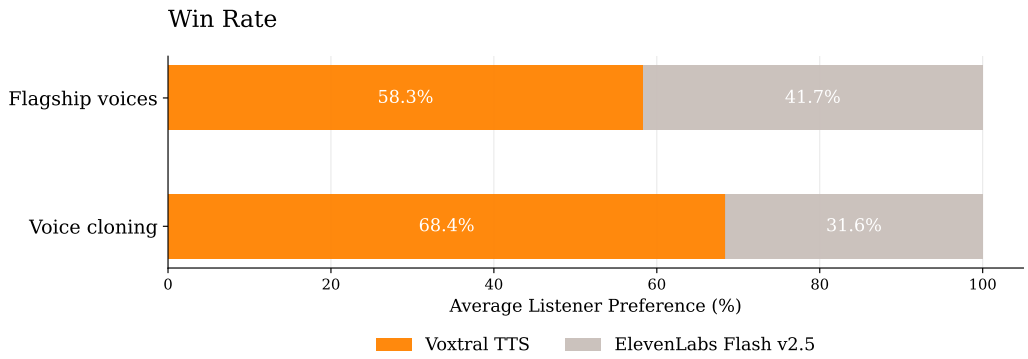

Figure 1: Voxtral TTS is preferred to ElevenLabs Flash v2.5 in human evaluations. We plot the win rate for Voxtral TTS against ElevenLabs Flash v2.5 in human evaluations across two categories. For flagship voices, we use the default voices for each model and 77 unique text examples. In the voice cloning set-up, we provide a short audio reference clip and 60 text prompts. In both categories, human annotators blindly rate which audio is better between the two models. Voxtral TTS is preferred in 58.3 and 68.4% of instances.

1 Introduction

Natural and expressive text-to-speech (TTS) remains a cornerstone of flexible human-computer interactions, with applications spanning virtual assistants, audiobooks, and accessibility tools. While recent neural TTS models achieve strong intelligibility, capturing the nuances and expressivity of human speech remains an open challenge, particularly in the zero-shot voice setting.

Recent zero-shot TTS systems typically condition generation on discrete speech tokens extracted from a short voice prompt, enabling generalization to unseen speakers and natural synthesis across long sequences [Borsos et al., 2023, Wang et al., 2023]. In parallel, diffusion and flow-based models are effective for modeling rich acoustic variation in speech generation [Popov et al., 2021, Le et al., 2023]. Recent speech codecs demonstrate that speech can be factorized into a low-rate semantic stream and a higher-rate acoustic stream [Défossez et al., 2024]. Hierarchical generators such as Moshi already exploit this structure using a temporal transformer over timesteps and a depth transformer over codec levels. However, acoustic generation in these systems remains depth-wise autoregressive. For TTS, this raises the question whether the dense acoustic component must be modeled auto-regressively at all, or whether it can instead be generated more effectively with a conditional continuous model.

In this work, we introduce Voxtral TTS, a multilingual zero-shot TTS system built around a representation-aware hybrid architecture. A voice prompt is tokenized through Voxtral Codec, a low-bitrate speech tokenizer with an ASR-distilled semantic token and finite scalar quantized (FSQ) acoustic tokens [Mentzer et al., 2023]. Given this factorized representation, a decoder-only transformer auto-regressively predicts the semantic token sequence, while a lightweight flow-matching model predicts the acoustic tokens conditioned on the decoder states. This design combines the strengths of auto-regressive modeling for long-range consistency with continuous flow-matching for rich acoustic detail. We adapt Direct Preference Optimization (DPO) [Rafailov et al., 2023] to this hybrid discrete-continuous setting by combining a standard preference objective over semantic token generation with a flow-based preference objective for acoustic prediction [Ziv et al., 2025].

Voxtral TTS supports 9 languages, supports voice prompts as short as 3 seconds, and is designed for low-latency streaming inference. Across automatic evaluations on SEED-TTS [Anastassiou et al., 2024] and MiniMax-TTS [Zhang et al., 2025], it achieves strong intelligibility and naturalness, beating ElevenLabs v3 on speaker similarity scores. In human evaluation for multilingual zero-shot voice cloning, it is preferred over ElevenLabs Flash v2.5 with a 68.4% win rate, while remaining competitive with strong proprietary systems on expressive flagship-voice evaluations.

2 Modeling

Figure 2 highlights the architecture of Voxtral TTS. It consists of a novel audio codec—Voxtral Codec—which encodes a reference voice sample into audio tokens consisting of semantic and acoustic tokens. The audio tokens are combined with text tokens to form the input to the LM decoder backbone. To generate speech, decoder backbone auto-regressively generates semantic token outputs. A flow-matching transformer generates the acoustic tokens. The codec decoder maps the output tokens to the corresponding audio waveform.

2.1 Voxtral Codec

Voxtral Codec is a convolutional–transformer autoencoder [Défossez et al., 2022] that compresses raw 24 kHz mono waveforms into 12.5 Hz frames of 37 discrete tokens (1 semantic + 36 acoustic), achieving a total bitrate of 2.14 kbps. These tokens serve as the input audio representation to Voxtral TTS. Through a novel combination of architectural and training objective improvements, Voxtral Codec outperforms existing baselines such as Mimi [Défossez et al., 2024], with results presented in Section 4.1.

Waveform Autoencoder. Inspired by prior works on transformer-based audio codecs [Parker et al., 2024, Wu et al., 2024], our audio tokenizer operates on "patchified" waveforms. A 24 kHz mono input waveform is chunked into non-overlapping patches of 240 samples, yielding a 100 Hz input to the encoder. The 100 Hz input frames are first projected to 1024-dimensional embeddings via a causal convolution with kernel size 7. The embeddings are then forwarded through 4 encoder blocks, each comprising:

- A 2-layer causal self-attention transformer with sliding window attention (window sizes $16 \rightarrow 8 \rightarrow 4 \rightarrow 2$, halved at each downsampling stage), ALiBi positional bias [Press et al., 2021], QK-norm, and LayerScale [Touvron et al., 2021] initialized at 0.01.
- A causal CNN layer. In the first three blocks, the CNN downsamples by $2\times$ (stride 2), yielding a cumulative $8\times$ reduction from 100 Hz to 12.5 Hz. In the fourth block, the CNN has stride 1 and projects the 1024-dimensional representation to a 292-dimensional latent space.

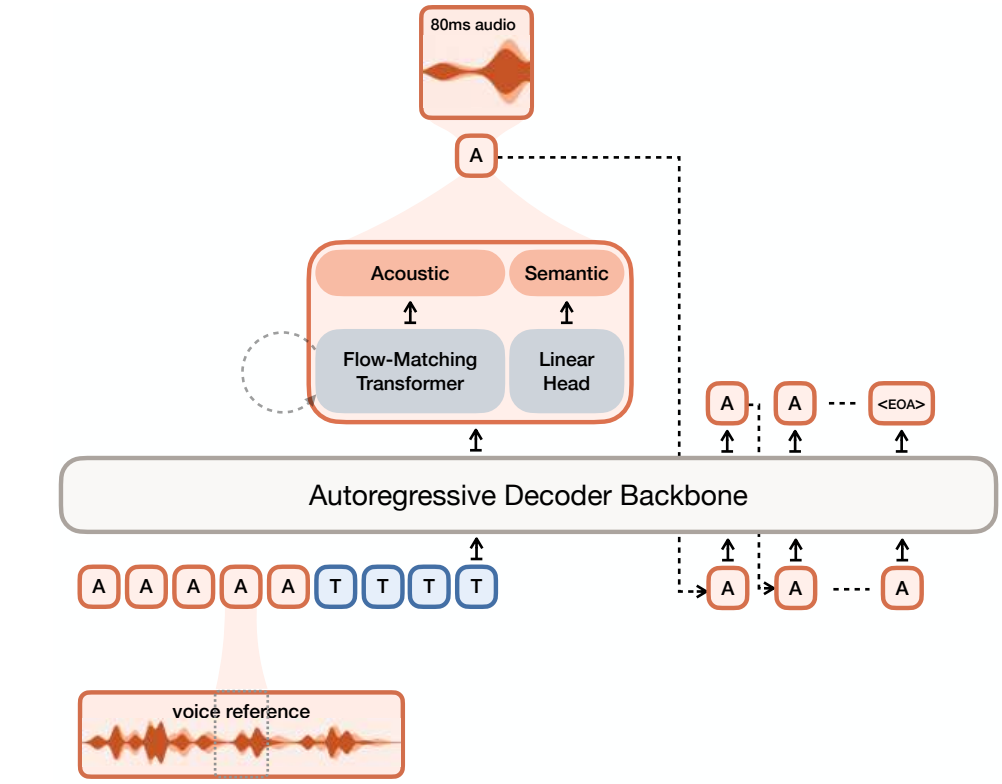

Figure 2: Architecture overview of Voxtral TTS. A voice reference ranging from 3s-30s is fed to the Voxtral Codec encoder to obtain audio tokens at a frame rate of 12.5 Hz. Each audio frame (labeled **A**) consists of a semantic token and acoustic tokens. The voice reference audio tokens along with the text prompt tokens (labeled **T**) are fed to the decoder backbone. The decoder auto-regressively generates a sequence of semantic tokens until it reaches a special End of Audio token (**<EOA>**). At each timestep, the semantic token from the decoder backbone is fed to a flow-matching transformer, which is run multiple times to predict the acoustic tokens. The semantic and acoustic tokens are fed to the Voxtral Codec decoder to obtain the generated waveform.

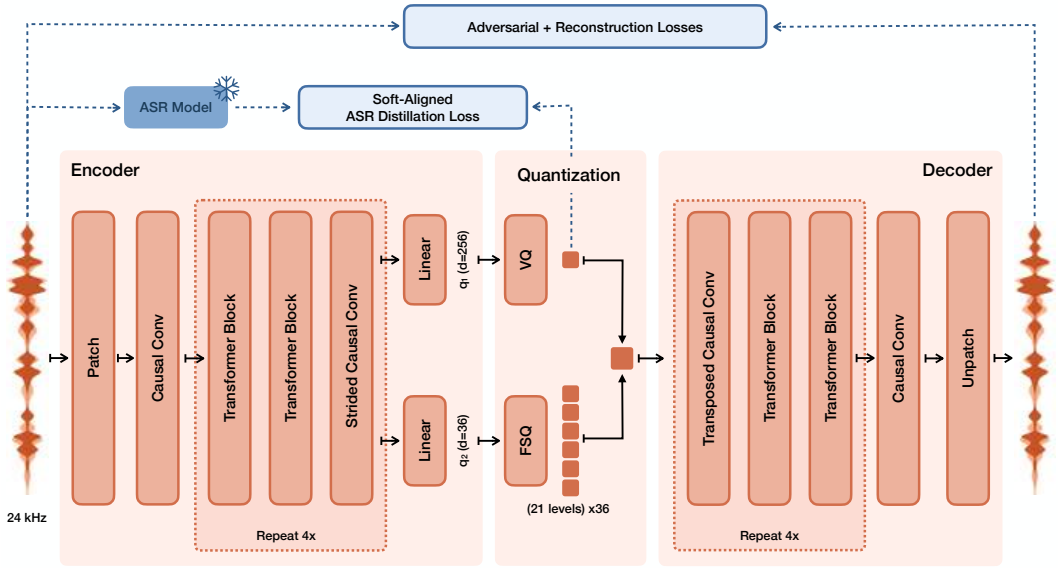

Figure 3: Architecture overview and training of Voxtral Codec. It consists of a split semantic VQ codebook and acoustic FSQ codebooks. Both semantic and acoustic tokens are combined for reconstruction. The semantic token has an additional distillation loss from a supervised ASR model.

The 292-dimensional latent is subsequently quantized to audio tokens (detailed below). The decoder mirrors the encoder in reverse: a causal CNN first projects the 292-dimensional latent back to 1024 dimensions, followed by 4 blocks each containing a transposed CNN (for $2\times$ upsampling) and a 2-layer causal self-attention transformer, gradually restoring the 12.5 Hz latent to 100 Hz. A final causal convolution with kernel size 7 maps from 1024 dimensions back to the patch size of 240 samples to reconstruct the waveform.

Representation Quantization. The 292-dimensional latent is split into a 256-dimensional *semantic* component and a 36-dimensional *acoustic* component, which are quantized independently:

- The semantic component is quantized through a learned vector quantizer (VQ; [Van Den Oord et al., 2017]) with a codebook of size 8192. During training, VQ is applied with 50% probability; the remaining samples pass through unquantized.
- Each of the 36 acoustic dimensions is passed through a $\tanh$ activation and independently quantized to 21 uniform levels via finite scalar quantization (FSQ; [Mentzer et al., 2023]). During training, we apply dither-style FSQ [Parker et al., 2024]: 50% of samples are quantized with FSQ, 25% receive uniform noise of magnitude $1/L$ (where L=21 is the number of levels), and 25% pass through unquantized.

The total bitrate is $12.5 \times (\log_2 8192 + 36 \times \log_2 21) \approx 2.14$ kbps.

Semantic Token Learning. To better incorporate the semantic content of speech into the semantic tokens, we adopt an auxiliary ASR distillation loss. Unlike prior works that learn "semantic" tokens by distilling self-supervised speech representations [Zhang et al., 2023, Défossez et al., 2024], which are more *phonetic* than semantic [Liu et al., 2024], we distill from a supervised ASR model. This has been shown to produce more effective semantic representations [Vashishth et al., 2024].

A frozen Whisper [Radford et al., 2023] model is run auto-regressively on the input audio to generate decoder hidden states and cross-attention weights. The post-VQ semantic embeddings are linearly projected to match the Whisper hidden dimension and then aligned to the decoder hidden states from the last decoder layer using a cosine distance loss:

$$\mathcal{L}_{\text{ASR}} = 1 - \frac{1}{L}\sum_{l=1}^{L} \frac{\tilde{\boldsymbol{z}}_l \cdot \boldsymbol{h}_l}{\|\tilde{\boldsymbol{z}}_l\| \, \|\boldsymbol{h}_l\|}, \qquad \tilde{\boldsymbol{z}}_l = \sum_{f=1}^{F} A_{l,f} \, \boldsymbol{z}_f \tag{1}$$

where $\boldsymbol{z}_f$ are the projected post-VQ semantic embeddings at codec frame f, $\boldsymbol{h}_l$ are the last-layer decoder hidden states from Whisper at token position l, and $A \in \mathbb{R}^{L\times F}$ is a soft alignment matrix derived from a subset of Whisper's cross-attention heads identified as best correlating with word-level timestamps via dynamic time warping (DTW) [Berndt and Clifford, 1994]. To compute A, the cross-attention weights from these heads are normalized across the decoder token dimension, median-filtered, and averaged over heads. The resulting matrix is linearly interpolated along the encoder frame axis to match the codec frame rate (12.5 Hz), so that $\tilde{\boldsymbol{z}}_l$ is the attention-weighted sum of codec embeddings aligned to the l-th decoder token.

This design allows the tokenizer to learn text-aligned semantic tokens without requiring an external forced aligner or paired transcripts, since the alignment is derived implicitly from Whisper's cross-attention weights. Distilling from continuous hidden states rather than hard transcript labels provides richer supervision, including model confidence and phonetic similarities.

Adversarial Training. A multi-resolution discriminator with 8 STFT sizes (2296, 1418, 876, 542, 334, 206, 126, 76) is trained along with the codec. Each discriminator is trained as a binary classifier between real audios $\boldsymbol{x}$ and reconstructed audios $\hat{\boldsymbol{x}}$ using a hinge loss. An L_1-based feature-matching loss is computed on the activations of every layer of each discriminator:

$$\mathcal{L}_{\text{feature}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N} \|D_n^m(\boldsymbol{x}) - D_n^m(\hat{\boldsymbol{x}})\|_1 \tag{2}$$

Here, D_n^m denotes the m-th layer of the n-th discriminator, where each of the N discriminators has M layers. Following Défossez et al. [2024], Parker et al. [2024], we use this feature-matching

Table 1: Key hyperparameters of the Voxtral Codec.

Parameter	Value
Input / Preprocessing	
Sampling rate	24000
Patch size	240
AutoEncoder	
Encoder patch projection kernel size	7
Encoder patch projection dimension	1024
Encoder transformer layers[1]	$2 \to 2 \to 2 \to 2$
Encoder sliding window size	$16 \to 8 \to 4 \to 2$
Encoder conv kernels	$4 \to 4 \to 4 \to 3$
Encoder conv strides	$2 \to 2 \to 2 \to 1$
(Decoder flips all $\to$ to $\leftarrow$ and uses transposed convolutions)	
Discrete bottleneck	
Semantic VQ[2] codebook size	8192
Acoustic FSQ[3] codebook count$\times$size	36×21
Discriminator	
FFT sizes	2296, 1418, 876, 542, 334, 206, 126, 76
Channels	256

[1] For training stability, we use LayerScale with initial scale of 0.01 and QK normalization with $\epsilon = 10^{-6}$.
[2] During training, VQ is applied with 50% probability.
[3] During training: 50% quantized with FSQ, 25% dithered (uniform noise of magnitude $1/L$), 25% unquantized.

loss *in place of* the standard GAN generator loss, as the evolving discriminator features provide an increasingly discriminative reconstruction signal throughout training.

Training Objective. Voxtral Codec is trained end-to-end with the following losses:

$$\alpha\mathcal{L}_{\text{feature}} + \beta\mathcal{L}_{\text{ASR}} + \gamma\mathcal{L}_{\text{L1}} + \gamma\mathcal{L}_{\text{STFT}} + \delta\mathcal{L}_{\text{commit}} \tag{3}$$

where α=1.0, β=1.0, γ=0.9999^t (with t the current training step), and δ=0.1. $\mathcal{L}_{\text{L1}}$ is the L_1 distance between the original and reconstructed waveforms, and $\mathcal{L}_{\text{STFT}}$ is an L_1 loss on their STFT magnitudes. Both reconstruction losses share the same exponential decay schedule γ, which bootstraps learning early in training and diminishes their influence as the adversarial signal strengthens [Parker et al., 2024]. $\mathcal{L}_{\text{commit}} = \|\boldsymbol{z}_e - \text{sg}(\boldsymbol{z}_q)\|_2^2$ is the VQ commitment loss [Van Den Oord et al., 2017], where sg denotes the stop-gradient operator.

Table 1 presents a summary of the Voxtral Codec configuration. The full model has approximately 300M parameters. All decisions are ablated and the final configuration achieves stable optimization with the best audio quality.

2.2 Decoder Backbone

The decoder backbone of Voxtral TTS follows the architecture of Ministral 3B [Liu et al., 2026], an auto-regressive decoder-only transformer. The input sequence consists of voice reference audio tokens followed by text tokens, from which the output audio tokens are auto-regressively generated. Each audio frame is represented by 37 discrete tokens (1 semantic, 36 acoustic). Each codebook has its own embedding lookup table (8192 entries for semantic and 21 for each acoustic), which are summed to produce a single embedding per audio frame.

The decoder backbone generates a sequence of hidden states. A linear head projects each hidden state h to logits over the semantic codebook vocabulary (8192 entries plus a special End of Audio (**<EOA>**) token), trained with a standard cross-entropy loss. To predict the acoustic tokens, h is fed to a flow-matching transformer, described in Section 2.3. The float-valued outputs of the flow-matching transformer are discretized before the next AR step to maintain a fully discrete token interface.

2.3 Flow-Matching Transformer

To predict the acoustic tokens, a flow-matching (FM) transformer operates independently on the hidden state h from each generation step in the decoder backbone. We model acoustic tokens in continuous space to leverage the smooth velocity field of FM, and discretize only at the output to interface with the AR backbone's discrete token vocabulary.

The FM transformer consists of a bidirectional 3-layer transformer with the same width as the decoder backbone. It models the velocity field that transports Gaussian noise (x_0) to acoustic embedding (x_1) over a series of function evaluation steps $0 \leq t \leq 1$. It receives as input h, the current function evaluation step t encoded as a sinusoidal embedding, and the current acoustic embedding $x_t \in \mathbb{R}^{36}$. We use a separate projection layer for each input h, t and x_t, because the scale of activations are different for each one. We also ablated providing conditioning using DiT style adaptive LayerNorm (AdaLN) layers [Peebles and Xie, 2023], but found our approach superior.

During training, the hidden state is dropped out 10% of the time for "unconditional" modeling. For inference, we use the Euler method to integrate the velocity vector field v_t using 8 function evaluations (NFEs) and classifier-free guidance (CFG) [Ho and Salimans, 2022]. Concretely, the form of v_t and x_t is:

$$v_t = \alpha v_\theta(x_t, t, h) + (1 - \alpha) v_\theta(x_t, t, \emptyset) \tag{4}$$

$$x_{t-\Delta t} = x_t - v_t \cdot \Delta t \tag{5}$$

where h is the hidden state from decoder backbone and $\emptyset$ is the unconditional case where we pass a vector of zeros with the same shape as h. $v_\theta(x_t, t, h)$ is the predicted velocity field at time step t, sample x_t and conditioning input h. We set $\Delta t = 1/8$ and $\alpha = 1.2$ based on the analysis in Section 5.2.

Note that in our architecture, CFG is applied independently at every frame in the FM transformer. Hence, it only requires an extra forward-propagation of only the FM transformer, and is thus significantly cheaper than applying CFG in the decoder backbone backbone. The float values predicted by the FM transformer are converted to discrete integer values by quantizing to the 21 FSQ levels. These discretized tokens are provided as input to the decoder backbone in the next decoding step.

Given the inputs to the decoder backbone are discrete tokens with embedding lookup, we also considered alternative architectures inspired by MaskGIT [Chang et al., 2022] and Depth Transformer [Défossez et al., 2024]. Both approaches performed reasonably well, but were inferior to FM in human evaluations, especially on expressivity. In addition, MaskGIT requires attending over all 36 acoustic codebook positions and conditioning tokens, resulting in a per-frame sequence length of 38, compared to just 3 in the FM transformer (h, t, x_t). Similarly, the Depth Transformer requires 36 auto-regressive decoding steps, compared to 8 NFEs for FM. Thus, FM is superior in quality, compute and latency.

3 Training

3.1 Pretraining

We train the model using paired audio and transcripts pseudo-labelled with Voxtral Mini Transcribe [Liu et al., 2025]. Each training sample consists of a tuple (A_1, T_2, A_2) where A_1 is a voice reference and T_2 is the transcript for A_2, which is our target for generation. Similar to Voxtral, we interleave these segments with a <next> special token between A_1 and T_2, and a <repeat> special token between T_2 and A_2. We ensure that A_1 and A_2 are single-speaker segments from the same speaker, but not necessarily temporally adjacent. The maximum duration of A_1 and A_2 are 180 seconds, and we ensure A_1 is at least 1 second long. Due to the long-tailed nature of natural conversational human speech duration, we find the model works best on voice prompts (A_1) between 3 and 25 seconds.

The loss is computed only on the tokens of A_2. We optimize the model using a two-part loss function consisting of a cross-entropy loss on the semantic token $\mathcal{L}_{\text{semantic}}$ and flow-matching loss $\mathcal{L}_{\text{acoustic}}$ on the acoustic tokens. We use the simple conditional flow-matching objective as shown below:

$$\mathcal{L}_{\text{acoustic}} = \mathbb{E}_{t\sim\mathcal{U}[0,1],x_0\sim\mathcal{D},x_1\sim\mathcal{N}(0,1)} \|v_\theta(x_t, t) - u_t(x_t|x_1, x_0)\|_2^2 \tag{6}$$

$$u_t(x_t|x_1, x_0) = x_1 - x_0 \tag{7}$$

where u_t is the conditional velocity target, v_θ is the velocity predicted by the FM transformer, x_1 is sampled from a normal distribution, and x_0 the data distribution $\mathcal{D}$. We initialize the decoder backbone with Ministral 3B. Newly introduced modules, such as the FM transformer, audio codebook embedding lookup-tables and output projection layers, are initialized from random. During training, we freeze the text-embedding layers in the decoder backbone to improve robustness to text tokens that appear with low-frequency in the Voxtral Mini Transcribe transcriptions. To avoid overfitting to silence, we also use a lower loss weight for frames that have no speech as determined by a voice-activity-detection (VAD) model and set the loss weight to 0 for extremely long silences. We also perform simple LLM based rewrites of the transcripts to introduce robustness to normalized vs un-normalized text (e.g. "5 - 4" vs "five minus four").

3.2 Direct Preference Optimization

We use Direct Preference Optimization (DPO) [Rafailov et al., 2023] to post-train the model, focusing on improving word error rate (WER) and speaker similarity. For the semantic codebook, we use the standard DPO objective. Given that the acoustic codebooks are predicted with flow-matching, we adapt the objective from Ziv et al. [2025]:

$$\mathcal{L}(\theta) = -\mathbb{E}_{t\sim\mathcal{U}(0,1),\, x^w, x^l} \log \sigma\big(-\beta\big(\Delta_\theta(x^w, x^l, t) - \Delta_{\theta_{\text{ref}}}(x^w, x^l, t)\big)\big)\,, \tag{8}$$

where

$$\Delta_\theta(x^w, x^l, t) = \|v_\theta(x_t^w, t) - u_t(x_t^w|x^w)\|_2^2 - \|v_\theta(x_t^l, t) - u_t(x_t^l|x^l)\|_2^2. \tag{9}$$

We make the objective suitable for our auto-regressive setup (note the bold $\boldsymbol{t}$ showing each token has a differently sampled t) by computing:

$$\Delta_\theta(x^w, x^l, \boldsymbol{t}) = \sum_{i=1}^{N_w} \|v_\theta(x^w_{i,\boldsymbol{t}_i}, \boldsymbol{t}_i) - u^w_{i,\boldsymbol{t}_i}\|_2^2 - \sum_{i=1}^{N_l} \|v_\theta(x^l_{i,\boldsymbol{t}_i}, \boldsymbol{t}_i) - u^l_{i,\boldsymbol{t}_i}\|_2^2 \tag{10}$$

and find that length normalization (dividing by length of winner) causes instability.

We ensure that the t and x_0 sampled for each location in the sequence is consistent for the policy model θ and reference model θ_{ref}. The two DPO losses are added with uniform weights but we use a $\beta_{\text{semantic}} = 0.1$ and $\beta_{\text{acoustic}} = 0.5$ as training is sensitive to the flow-DPO loss. A low learning rate of $8\text{e}{-8}$ is used for training stability.

The data for DPO is gathered using a rejection-sampling pipeline that takes as input a set of voice samples from a held-out set of single-speaker voice samples and diverse synthetically generated text-prompts. We prompt Mistral Small Creative [1] with the transcript of the voice prompt and randomly chosen personas to synthesize a diverse array of texts which continue or reply to the conversational context. The pretrained checkpoint then takes as input the voice and text prompts and generates multiple samples from each input, from which winner and loser pairs can be constructed. Winners and losers are determined from WER, speaker similarity, loudness consistency, UTMOS-v2 [Baba et al., 2024] and other LM judge metrics. We optimize the model using the combined DPO loss along with the pretraining objective on high-quality speech for 1 epoch, as we found that training longer on synthetic data led to more robotic speech.

4 Results

4.1 Voxtral Codec

Table 2 shows a comparison between Voxtral Codec and Mimi on the Expresso dataset [Nguyen et al., 2023]. We evaluate on the following objective metrics: Mel distance, STFT distance, perceptual evaluation of speech quality (PESQ), extended short-time objective intelligibility (ESTOI), word error rate between transcriptions generated using an ASR model corresponding to the source and reconstruction (ASR-WER), speaker similarity score computed using a speaker embedding model. We also report the bitrates and frames per second (fps), which are relevant as these codecs are used in the context of auto-regressive decoder models. Given Mimi uses an RVQ design for acoustic

[1] https://docs.mistral.ai/models/mistral-small-creative-25-12

codebooks, it has the flexibility to choose a subset of codebooks to trade-off bitrate and quality. When Voxtral Codec is compared to Mimi in a 16 codebook configuration, such that the bitrates are similar, Voxtral Codec outperforms on all the objective metrics. On an internal subjective assessment, we found Voxtral Codec to be comparable or better than Mimi at 16 codebooks on audios consisting of speech which is our main focus.

Table 2: Comparison of Voxtral Codec and Mimi on the Expresso dataset.

Model	fps	token/frame × vocab. size	bitrate	Reconstruction (↓)		Intrusive (↑)		Perceptual	
			(kbps)	Mel	STFT	PESQ	ESTOI	ASR-WER (%)↓	Speaker Sim↑
Mimi – 8cb (Moshi)	12.5	8 × (2048)	1.1	0.702	1.177	2.07	0.803	11.75	0.672
Mimi – 16cb	12.5	16 × (2048)	2.2	0.618	1.100	2.67	0.865	11.01	0.829
Mimi – full 32cb	12.5	32 × (2048)	4.4	0.552	1.040	3.18	0.910	10.25	0.902
Voxtral Codec	12.5	1 × (8192) + 36 × (21)	2.1	0.545	0.982	3.05	0.882	10.66	0.843

4.2 Automatic Evaluations

We evaluate Voxtral TTS, ElevenLabs v3 and ElevenLabs Flash v2.5 on SEED-TTS [Anastassiou et al., 2024] and the nine supported languages in MiniMax-TTS [Zhang et al., 2025] using automated metrics:

1. **Word Error Rate (WER)**: Measured by Voxtral Mini Transcribe v2 to capture the intelligibility of speech.
2. **UTMOS-v2** [Baba et al., 2024]: Predicts the Mean Opinion Score (MOS) of generated speech.
3. **Speaker Similarity**: Speaker embeddings are predicted using the ECAPA-TDNN model [Desplanques et al., 2020] and the cosine similarity is computed against the reference embedding. This evaluates how closely generated speech emulates the provided voice reference.

The results for the three models are presented in Table 3. While both ElevenLabs models achieve low WERs across languages, Voxtral TTS significantly outperforms ElevenLabs on the speaker similarity metrics. Surprisingly, we find that ElevenLabs Flash v2.5 performs better on most automated metrics and ElevenLabs v3 better on human evaluations, particularly with emotion steering. This highlights the importance of performing human evaluations in conjunction with automatic evaluations.

Table 3: WER, UTMOS, and Speaker Similarity scores for Voxtral TTS, ElevenLabs v3, and ElevenLabs Flash v2.5.

	WER (%) ↓			UTMOS ↑			Speaker Sim ↑		
Task	Voxtral	ElevenLabs v3	ElevenLabs Flash	Voxtral	ElevenLabs v3	ElevenLabs Flash	Voxtral	ElevenLabs v3	ElevenLabs Flash
MiniMax									
Arabic	**2.68**	3.67	2.86	**3.07**	2.50	2.89	**0.746**	0.546	0.539
German	0.83	**0.45**	1.08	3.12	2.90	**3.27**	**0.721**	0.457	0.489
English	0.63	0.48	**0.33**	**4.30**	4.27	4.27	**0.786**	0.484	0.489
Spanish	0.51	0.87	**0.49**	**3.41**	3.18	2.99	**0.762**	0.443	0.541
French	3.22	2.34	**2.26**	2.83	2.90	**2.94**	**0.587**	0.339	0.378
Hindi	**4.99**	8.71	5.08	**3.56**	**3.56**	3.35	**0.839**	0.707	0.679
Italian	1.32	0.58	**0.55**	**3.43**	3.08	3.09	**0.739**	0.527	0.485
Dutch	1.99	1.52	**0.83**	**3.89**	3.53	3.68	**0.720**	0.397	0.598
Portuguese	1.02	**0.92**	1.15	**3.66**	3.41	3.41	**0.785**	0.571	0.642
Seed TTS	1.23	1.26	**0.86**	**4.11**	3.92	4.09	**0.628**	0.392	0.413

4.3 Human Evaluations

Automated metrics cannot measure the naturalness and expressivity of a TTS model, especially the ability of the model to speak with a specific emotion. We find that UTMOS is only a loose proxy, not well calibrated across languages and only weakly correlated with human preference. Hence, we perform two sets of human evaluations in which annotators compare generations between two models without knowing their identities. The evaluation consists of 77 prompts, with 11 of them neutral while 66 of them have an associated expected emotion. For all evaluations, annotators are instructed to choose whether one of the generations is "slightly better", "much better" or if they are "both good"

or "both bad". During labeling, all audio samples are resampled to 24 kHz WAV format (even the voice samples) to ensure there is no bias due to audio quality.

4.3.1 Flagship voices

First, we compare our flagship voices (*British-Female, British-Male, American-Male, French-Female*) against the flagship voices of same gender and accent provided by competitors. We run two sub-evaluations:

1. **Explicit steering**: We test the ability to bias a TTS model's generation toward a specific emotion. The TTS prompts which have an associated emotion (not Neutral) are provided as free-form instruction to Gemini 2.5 Flash TTS as it supports free-form instructions such as "Speak in an angry tone.". For ElevenLabs v3 we provide emotion tags enclosed in brackets [2]. While Voxtral TTS does not support emotion tags/text-instructions, we steer the generation by leveraging a different voice prompt provided from the same speaker which embodies the requested emotion.
2. **Implicit steering**: We test the model's capabilities to infer emotion from provided text (e.g. "This is the best day of my life!"). No emotion label or instruction is provided to the model. For Voxtral TTS, we use a neutral voice prompt.

We use three annotators who are native speakers of the same dialect for each language per pair. The win rates of Voxtral TTS (excluding ties) are presented in Table 4. Gemini 2.5 Flash TTS is the strongest model, and Voxtral TTS is competitive against ElevenLabs v3. In the implicit steering setting, Voxtral TTS consistently outperforms both ElevenLabs models.

Table 4: Voxtral TTS win rates by steering type. In the explicit steering setting, Voxtral TTS is competitive with ElevenLabs v3, while having a higher win rate compared to both ElevenLabs models in the implicit steering setting.

Emotion steering	Opponent Model	Voxtral TTS Win Rate (%)
Explicit	ElevenLabs v3	51.0
	Gemini 2.5 Flash TTS	35.4
Implicit	ElevenLabs Flash v2.5	58.3
	ElevenLabs v3	55.4
	Gemini 2.5 Flash TTS	37.1

4.3.2 Zero-Shot Voice Cloning

To evaluate voice cloning capabilities, we source high quality audios from two recognized speakers in each language. We generate speech from each model in a zero-shot setting, and instruct annotators to rate the generations based on (a) likeness of the generated audio to voice prompt and (b) naturalness of speech and expressivity.

Table 5 shows the Voxtral TTS win rates against ElevenLabs Flash v2.5 across languages. Overall, Voxtral TTS has a win rate of 68.4%, with significantly better results across both high and low-resource languages (such as Arabic and Hindi). Notably, the Voxtral TTS win rate is much higher in the zero-shot setting (68.4%) compared with flagship voices (57.1%), highlighting that Voxtral TTS is a far more generalizable model, capturing a diverse range of user-voices.

5 Analysis

In this Section, we provide a comparison between the pretrained and DPO checkpoints and ablate the pertinent inference parameters.

[2] https://elevenlabs.io/blog/eleven-v3-audio-tags-expressing-emotional-context-in-speech

Table 5: Voxtral TTS win rate against ElevenLabs Flash v2.5 across languages. Voxtral TTS matches or outperforms ElevenLabs Flash v2.5 on every language, and has an overall micro-average win rate of 68.4%.

Language	**Voxtral TTS Win Rate** (%)
Arabic	72.9
Dutch	49.4
English	60.8
French	54.4
German	72.0
Hindi	79.8
Italian	57.1
Portuguese	74.4
Spanish	87.8
Overall	**68.4**

5.1 DPO improvements

Table 6 shows the WER and UTMOS metrics for the pretrained and DPO checkpoints. Overall, DPO improves on both metrics, with the largest gains in German and French and regressions only on Hindi. Qualitatively, we find that the DPO model hallucinates less and skips fewer words. DPO also ameliorates the pretrained model's occasional tendency to significantly taper in volume throughout the audio. Interestingly, DPO has minimal effect on speaker similarity, which is within ± 0.01 of pretrain checkpoint (not presented here for brevity).

Table 6: DPO improves WER and UTMOS across languages.

	WER (%) ↓		UTMOS ↑	
Task	Pretrain	DPO	Pretrain	DPO
MiniMax				
Arabic	2.80	**2.68** (-0.12)	3.01	**3.07** (+0.06)
German	4.08	**0.83** (-3.25)	3.05	**3.12** (+0.07)
English	0.84	**0.63** (-0.21)	4.25	**4.30** (+0.05)
Spanish	0.56	**0.51** (-0.06)	3.38	**3.41** (+0.04)
French	5.01	**3.22** (-1.79)	2.76	**2.83** (+0.07)
Hindi	**3.39**	4.99 (+1.61)	3.43	**3.56** (+0.13)
Italian	2.18	**1.32** (-0.85)	3.36	**3.43** (+0.07)
Dutch	3.10	**1.99** (-1.11)	3.85	**3.89** (+0.04)
Portuguese	1.17	**1.02** (-0.15)	3.60	**3.66** (+0.06)
Seed TTS	1.58	**1.23** (-0.35)	4.07	**4.11** (+0.04)

5.2 Inference Parameters

Figure 4 demonstrates the effect on the automatic evaluation metrics as the number of functional evaluations (NFEs) and choice of CFG α are varied. There are marked improvements across metrics as the NFEs is increased from 2 to 8. We find that increasing number of NFEs beyond 8 yields marginal improvement in speaker similarity and minor degradations in WER. Thus, we chose 8 NFEs as our default inference setting.

Increasing the value of CFG α, we find that there is a nearly monotonic improvement in all metrics except UTMOS-v2. However, internal human evaluations found that a higher α led to over-adherence to the provided voice-prompt and the model failed to bias towards emotions that are implicit in the text prompt. We also find that lower $\alpha = 1.2$ works best for higher quality audio (e.g. professional recordings), while in-the-wild recordings might benefit from a higher α.

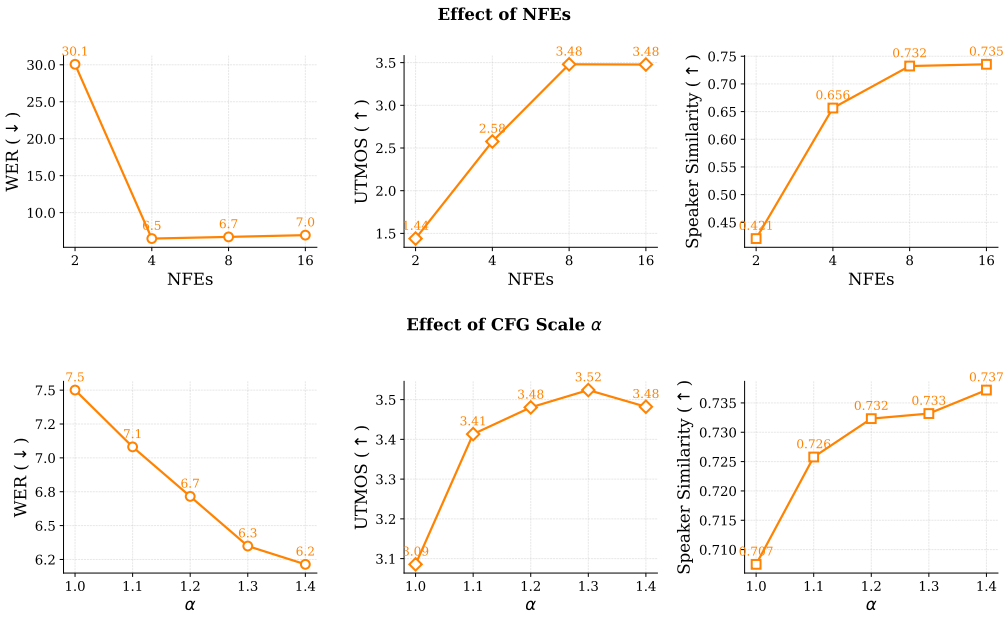

Figure 4: Effect of NFEs and CFG on automatic evaluations. The metrics are averaged over SEED-TTS and the 9 languages in MiniMax. Increasing the NFEs from 2 to 8 improves speaker similarity and UTMOS metrics. There is a slight regression in WER as the NFEs is increased beyond this. The metrics monotonically increase with higher CFG, but human evaluations flagged regressions in text-adherence with high α.

6 Inference and Serving in vLLM-Omni

Voxtral TTS is served through vLLM-Omni [Yin et al., 2026], an extension of the vLLM [Kwon et al., 2023] for multi-stage multimodal models. Voxtral TTS is decomposed into a two-stage pipeline: a generation stage that predicts the audio tokens (semantic and acoustic), followed by a codec decoding stage that converts the tokens into a waveform. The two stages communicate through an asynchronous chunked streaming protocol over shared memory, enabling first-audio latency well before the full waveform has been generated.

6.1 CUDA Graph Acceleration for Flow-Matching Transformer

The flow-matching transformer is the computational bottleneck of the generation stage. Each decoding step requires N functional evaluations with CFG, requiring $2 \times N$ forward passes per generated frame.

To eliminate Python-level overhead and kernel-launch latency, the entire ODE solver is captured into CUDA graphs. At startup, an eager warmup pass is performed for each bucket size and the corresponding CUDA graph is then captured. During inference, the actual batch size is rounded up to the nearest bucket by padding the input with zeros. Next, the CUDA graph is replayed and outputs are sliced back to the actual batch size. If the batch size exceeds the largest captured bucket, the model falls back to eager execution.

To evaluate the effect of CUDA graph acceleration, we compare the latency and real-time factor (RTF) when decoding with eager mode and CUDA graphs. Table 7 reports results for a 500-character text input, a 10-second audio reference and concurrency 1 on a single H200. Enabling CUDA graphs results in a 47% improvement to latency and reduces the RTF by 2.5x.

Table 7: Effect of CUDA graph acceleration on the flow-matching transformer.

Configuration	Latency	RTF
Eager mode	133 ms	0.258
CUDA graph	70 ms	0.103

6.2 Asynchronous Chunked Streaming

The two pipeline stages run in separate scheduling loops. To overlap the autoregressive generation stage decoding with codec decoding stage waveform synthesis, an asynchronous chunked streaming protocol is introduced.

After each generation step, the vLLM-Omni transfer manager stores the audio tokens to a per-request buffer. Once the buffer length reaches a pre-defined length, a chunk of tokens are emitted to the codec decoding stage. To ensure coherence between chunks, each emitted chunk includes a slice of previous frames in addition to the new frames. This overlap enables the codec decoder's causal sliding-window attention to maintain temporal coherence across chunk boundaries.

6.3 Inference Throughput

With the techniques introduced in this section, Voxtral TTS achieves low-latency, high-throughput inference. Table 8 shows the serving performance on a single NVIDIA H200 from concurrency 1 to 32 with 500-character text inputs and 10-second audio references. As the concurrency is increased from 1 to 32, the throughput scales from 119 to 1,431 characters per second per GPU, a 12x increase, while latency remains sub-second. The wait rate, defined as the fraction of audio chunks for which the client must stall since it is waiting for outputs, remains zero across all concurrency levels. As concurrency grows, per-request RTF increases modestly to 0.302 at concurrency 32, still well within the real-time boundary.

These results demonstrate that Voxtral TTS is suitable for production deployment: a single H200 can serve over 30 concurrent users with uninterrupted streaming output and sub-second time to first audio.

Table 8: Serving performance of Voxtral TTS on a single H200.

Concurrency	Latency	RTF	Throughput (char/s/GPU)	Wait Rate
1	70 ms	0.103	119.14	0%
16	331 ms	0.237	879.11	0%
32	552 ms	0.302	1430.78	0%

7 Conclusion

We introduced Voxtral TTS, a multilingual TTS model that leverages a hybrid architecture for auto-regressive generation of semantic tokens and flow-matching for acoustic tokens. The tokens correspond to those from Voxtral Codec, a speech tokenizer that combines ASR-distilled semantic tokens with FSQ acoustic tokens.

Voxtral TTS is able to generate expressive, voice-cloned speech from as little as 3 seconds of reference audio, and is preferred to API baselines in human evaluations. We release Voxtral TTS as open weights under the CC BY-NC license to support further research and development of expressive TTS systems.

Core contributors

Alexander H. Liu, Alexis Tacnet, Andy Ehrenberg, Andy Lo, Chen-Yo Sun, Guillaume Lample, Henry Lagarde, Jean-Malo Delignon, Jaeyoung Kim, John Harvill, Khyathi Raghavi Chandu, Margaret Jennings, Patrick von Platen, Pavankumar Reddy Muddireddy, Rohin Arora, Sanchit Gandhi, Samuel Humeau, Soham Ghosh, Srijan Mishra, Van Phung.

Contributors

Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andrew Bai, Angele Lenglemetz, Anmol Agarwal, Anton Eliseev, Antonia Calvi, Arjun Majumdar, Avi Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault, Benjamin Tibi, Charlotte Cronjäger, Clémence Lanfranchi, Connor Chen, Corentin Barreau, Corentin Sautier, Cyprien Courtot, Darius Dabert, Diego de las Casas, Elizaveta Demyanenko, Elliot Chane-Sane, Enguerrand Paquin, Etienne Goffinet, Fabien Niel, Faruk

Ahmed, Federico Baldassarre, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Genevieve Hayes, Georgii Novikov, Giada Pistilli, Guillaume Kunsch, Guillaume Martin, Guillaume Raille, Gunjan Dhanuka, Gunshi Gupta, Han Zhou, Harshil Shah, Hope McGovern, Hugo Thimonier, Indraneel Mukherjee, Irene Zhang, Jaeyoung Kim, Jan Ludziejewski, Jason Rute, Joachim Studnia, John Harvill, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Julien Tauran, Karmesh Yadav, Kartik Khandelwal, Kilian Tep, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Manan Sharma, Marie Pellat, Mark Prins, Martin Alexandre, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mert Unsal, Mia Chiquier, Minh-Quang Pham, Nathan Grinsztajn, Neha Gupta, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Philippe Pinel, Philomène Chagniot, Pierre Stock, Piotr Miłoś, Prateek Gupta, Pravesh Agrawal, Quentin Torroba, Ram Ramrakhya, Rishi Shah, Romain Sauvestre, Roman Soletskyi, Rosalie Millner, Rupert Menneer, Sagar Vaze, Samuel Barry, Samuel Humeau, Sandeep Subramanian, Sean Cha, Shashwat Verma, Siddhant Waghjale, Siddharth Gandhi, Simon Lepage, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Edwards, Tyler Wang, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Vedant Nanda, Victor Jouault, Vincent Maladière, Virgile Richard, Vladislav Bataev, Wassim Bouaziz, Wen-Ding Li, William Havard, William Marshall, Xinghui Li, Xingran Guo, Xinyu Yang, Yannic Neuhaus, Yassine El Ouahidi, Yassir Bendou, Yihan Wang, Yimu Pan, Zaccharie Ramzi, Zhenlin Xu.

7.1 Acknowledgements

We would like to thank Han Gao, Hongsheng Liu, Roger Wang, and Yueqian Lin from the vLLM-Omni team for their support and contributions in integrating Voxtral TTS into the vLLM-Omni framework.

References

Philip Anastassiou, Jiawei Chen, Jitong Chen, Yuanzhe Chen, Zhuo Chen, Ziyi Chen, Jian Cong, Lelai Deng, Chuang Ding, Lu Gao, Mingqing Gong, Peisong Huang, Qingqing Huang, Zhiying Huang, Yuanyuan Huo, Dongya Jia, Chumin Li, Feiya Li, Hui Li, Jiaxin Li, Xiaoyang Li, Xingxing Li, Lin Liu, Shouda Liu, Sichao Liu, Xudong Liu, Yuchen Liu, Zhengxi Liu, Lu Lu, Junjie Pan, Xin Wang, Yuping Wang, Yuxuan Wang, Zhengnan Wei, Jian Wu, Chao Yao, Yifeng Yang, Yuanhao Yi, Junteng Zhang, Qidi Zhang, Shuo Zhang, Wenjie Zhang, Yang Zhang, Zilin Zhao, Dejian Zhong, and Xiaobin Zhuang. Seed-tts: A family of high-quality versatile speech generation models, 2024. URL https://arxiv.org/abs/2406.02430.

Kaito Baba, Wataru Nakata, Yuki Saito, and Hiroshi Saruwatari. The t05 system for the VoiceMOS Challenge 2024: Transfer learning from deep image classifier to naturalness MOS prediction of high-quality synthetic speech. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 818–824, 2024. doi: 10.1109/SLT61566.2024.10832315.

Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audiolm: A language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2523–2533, 2023. doi: 10.1109/TASLP.2023.3288409.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11315–11325, June 2022.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.

Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification. In *Interspeech 2020*, pages 3830–3834, 2020. doi: 10.21437/Interspeech.2020-2650.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. URL https://arxiv.org/abs/2207.12598.

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. doi: 10.1145/3600006.3613165.

Matt Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, and Wei-Ning Hsu. Voicebox: Text-guided multilingual universal speech generation at scale. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/2d8911db9ecedf866015091b28946e15-Abstract-Conference.html.

Alexander H Liu, Sung-Lin Yeh, and James R Glass. Revisiting self-supervised learning of speech representation from a mutual information perspective. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12051–12055. IEEE, 2024.

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, Sanchit Gandhi, Soham Ghosh, Srijan Mishra, and Thomas Foubert. Voxtral, 2025. URL https://arxiv.org/abs/2507.13264.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple, 2023.

Tu Anh Nguyen, Wei-Ning Hsu, Antony d'Avirro, Bowen Shi, Itai Gat, Maryam Fazel-Zarani, Tal Remez, Jade Copet, Gabriel Synnaeve, Michael Hassid, et al. Expresso: A benchmark and analysis of discrete expressive speech resynthesis. *arXiv preprint arXiv:2308.05725*, 2023.

Julian D Parker, Anton Smirnov, Jordi Pons, CJ Carr, Zack Zukowski, Zach Evans, and Xubo Liu. Scaling transformers for low-bitrate high-quality speech coding. *arXiv preprint arXiv:2411.19842*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, October 2023.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-TTS: A diffusion probabilistic model for text-to-speech. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8599–8608. PMLR, 2021. URL https://proceedings.mlr.press/v139/popov21a.html.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 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*, pages 28492–28518. PMLR, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.18290.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 32–42, 2021.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Shikhar Vashishth, Harman Singh, Shikhar Bharadwaj, Sriram Ganapathy, Chulayuth Asawaroengchai, Kartik Audhkhasi, Andrew Rosenberg, Ankur Bapna, and Bhuvana Ramabhadran. Stab: Speech tokenizer assessment benchmark. *arXiv preprint arXiv:2409.02384*, 2024.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023. URL https://arxiv.org/abs/2301.02111.

Haibin Wu, Naoyuki Kanda, Sefik Emre Eskimez, and Jinyu Li. Ts3-codec: Transformer-based simple streaming single codec. *arXiv preprint arXiv:2411.18803*, 2024.

Peiqi Yin, Jiangyun Zhu, Han Gao, Chenguang Zheng, Yongxiang Huang, Taichang Zhou, Ruirui Yang, Weizhi Liu, Weiqing Chen, Canlin Guo, Didan Deng, Zifeng Mo, Cong Wang, James Cheng, Roger Wang, and Hongsheng Liu. vllm-omni: Fully disaggregated serving for any-to-any multimodal models, 2026. URL https://arxiv.org/abs/2602.02204.

Bowen Zhang, Congchao Guo, Geng Yang, Hang Yu, Haozhe Zhang, Heidi Lei, Jialong Mai, Junjie Yan, Kaiyue Yang, Mingqi Yang, Peikai Huang, Ruiyang Jin, Sitan Jiang, Weihua Cheng, Yawei Li, Yichen Xiao, Yiying Zhou, Yongmao Zhang, Yuan Lu, and Yucen He. Minimax-speech: Intrinsic zero-shot text-to-speech with a learnable speaker encoder, 2025. URL https://arxiv.org/abs/2505.07916.

Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, and Xipeng Qiu. Speechtokenizer: Unified speech tokenizer for speech large language models. *arXiv preprint arXiv:2308.16692*, 2023.

Alon Ziv, Sanyuan Chen, Andros Tjandra, Yossi Adi, Wei-Ning Hsu, and Bowen Shi. Mr-flowdpo: Multi-reward direct preference optimization for flow-matching text-to-music generation, 2025. URL https://arxiv.org/abs/2512.10264.