# Qwen3-TTS VoiceDesign Model Best Practices: Optimization, Prompt Engineering, and Artifact Mitigation with mlx-audio

## 1. Executive Summary and Strategic Overview

The landscape of neural speech synthesis has undergone a paradigm shift with the release of the Qwen3-TTS family by Alibaba Cloud. Historically, high-fidelity Text-to-Speech (TTS) systems were bifurcated into two distinct categories: concatenative or parametric systems that offered stability but low expressiveness, and more recent diffusion or transformer-based systems that offered high expressiveness but required heavy distinct speaker embeddings (cloning). The Qwen3-TTS architecture, particularly its **VoiceDesign** capability, introduces a third paradigm: zero-shot voice generation controlled purely by natural language instructions.1 This allows for the synthesis of entirely novel speaker identities—defined by age, gender, timbre, and prosody—without the need for reference audio samples or fine-tuning, democratization of high-end audio production that was previously the domain of proprietary, closed-source APIs.

However, the deployment of such sophisticated autoregressive models on consumer hardware, particularly within the Apple Silicon ecosystem via the mlx-audio framework, presents a unique constellation of technical challenges. Users transitioning from CUDA-based environments to the Metal Performance Shaders (MPS) backend often encounter distinct acoustic artifacts, most notably the "infinite beeping loop," spectral distortion (metallic grain), and signal truncation.2 These issues are not merely bugs but are intrinsic to the interaction between the model's 12Hz discrete tokenization scheme and the quantization strategies employed to fit these Large Audio Models (LAMs) into unified memory.

This comprehensive research report provides an exhaustive technical analysis of the Qwen3-TTS VoiceDesign model, specifically tailored for deployment on Apple Silicon using mlx-audio. It synthesizes data from technical reports, community benchmarks, and code repository analyses to establish a definitive set of best practices. The report details the precise hyperparameter configurations required to stabilize autoregressive generation, constructs a rigorous taxonomy for --instruct prompt engineering to maximize fidelity, and offers deep-dive troubleshooting strategies for mitigating codec collapse and quantization noise. By adhering to the protocols outlined herein, developers and content creators can achieve studio-grade synthesis stability, leveraging the full potential of Qwen3-TTS's 10-language capability and 1.7-billion parameter architecture.4

## 2. Architectural Framework and Mechanism of Action

To effectively optimize Qwen3-TTS and troubleshoot its failure modes, it is imperative to first understand the underlying architecture that differentiates it from previous generations of TTS models. The system is not simply mapping text to waveforms; it is performing a complex translation between semantic text tokens and acoustic codec tokens.

### 2.1 The 12Hz Discrete Tokenizer: Efficiency vs. Stability

At the heart of the Qwen3-TTS system lies the proprietary Qwen3-TTS-Tokenizer-12Hz.6 In traditional neural TTS (such as FastSpeech or Tacotron), the model predicts mel-spectrograms which are then converted to audio by a vocoder. Qwen3, however, treats speech synthesis as a language modeling task.

The tokenizer compresses continuous speech signals into a sequence of discrete codes from a learned codebook. The "12Hz" designation indicates that the model represents one second of audio with only 12 tokens. This is an extremely high compression rate compared to typical audio codecs which might operate at 25Hz or 50Hz.8

* **Semantic Density:** This high compression means each token carries a immense amount of acoustic and semantic weight. A single token represents approximately 83 milliseconds of audio.
* **The Stability Trade-off:** While this architecture allows the model to "understand" the semantic link between text and speech prosody with high efficiency, it introduces fragility. If the autoregressive decoder makes a prediction error on a single token, the audible result is not a subtle glitch but often a catastrophic artifact—a skipped word, a sudden scream, or the infamous "beep of death." The model effectively loses track of the acoustic state, and because the tokens are discrete, the error propagates efficiently through the sequence.

### 2.2 Dual-Track Autoregressive Generation

The Qwen3-TTS utilizes a specialized Dual-Track Transformer architecture to handle the mapping between text and audio.8

* **Track 1 (Text Encoding):** The model first processes the input text and the "VoiceDesign" instructions. These instructions are not just metadata; they are encoded into the latent space to condition the generation, acting as a style guide for the decoder.
* **Track 2 (Audio Decoding):** The model then predicts the audio tokens autoregressively. This means the generation of token  is dependent on tokens  to .

This autoregressive nature is the primary vector for the "beeping" artifact. In language models, if a model gets stuck in a loop, it repeats text. In Audio LMs like Qwen3, a repetition loop often manifests as a repeating high-frequency token, which the vocoder decodes as a continuous beep or a rhythmic mechanical noise.2

### 2.3 The mlx-audio Ecosystem and Apple Silicon

The mlx-audio library serves as the bridge between this complex architecture and Apple's M-series chips.5 Unlike PyTorch on CUDA, which uses dedicated VRAM, MLX utilizes the Unified Memory Architecture (UMA) of the Mac.

* **Quantization Necessity:** To run the 1.7B parameter model efficiently on devices like a MacBook Air or Pro, models are often quantized to 4-bit or 8-bit precision.12
* **Precision Risks:** The report analysis indicates that aggressive quantization (specifically 4-bit) can degrade the distinctness of the embedding vectors for specific VoiceDesign traits. This blurs the lines between different timbres, leading to "spectral mud" or causing the model to ignore subtle instruction prompts (e.g., ignoring "whisper" and generating normal speech).5

## 3. Optimizing VoiceDesign: The Art of --instruct Prompting

The defining feature of Qwen3-TTS is **VoiceDesign**—the ability to generate voices from natural language descriptions without a reference audio file. The quality and stability of the output are deterministically linked to the specificity, syntax, and vocabulary of the --instruct prompt.

### 3.1 The Grammar of Voice Description

The model's instruction encoder has been trained to recognize specific semantic clusters related to voice physiology and performance style. Providing abstract or poetic descriptions often yields unstable results because the model cannot map them to acoustic features.

Effective prompts are constructed using a rigorous taxonomy of five acoustic dimensions: **Timbre**, **Pitch**, **Speed**, **Emotion**, and **Persona**.1

#### Table 1: Comprehensive Taxonomy of Validated Voice Design Descriptors

| **Dimension** | **Primary Descriptors (High Stability)** | **Secondary Modifiers (Nuance)** | **Negative/Avoidance Terms (To avoid in mental model)** | **Acoustic Correlate** |
| --- | --- | --- | --- | --- |
| **Timbre (Texture)** | Magnetic, Crisp, Raspy, Smooth, Sweet, Rich, Powerful, Husky, Mellow | Thick, Thin, Airy, Breathy, Grainy, Edgy, Bright, Resonant | "Robotic", "Metallic", "Synthetic", "Noisy" | Affects the spectral envelope and harmonic saturation. |
| **Pitch** | High, Mid, Low | Slightly High, Slightly Low, Deep, Piercing, Bass-heavy | "Flat", "Monotone", "Squeaky" | Shifts the fundamental frequency (). |
| **Speed (Rate)** | Fast, Medium, Slow | Slightly Fast, Slightly Slow, Rushed, Deliberate, Paced | "Erratic", "Stuttering" | Controls token generation duration per phoneme. |
| **Emotion** | Cheerful, Calm, Gentle, Serious, Lively, Composed, Soothing, Angry, Sad, Panic | Playful, Empathetic, Authoritative, Incredulous, Excited, Depressed | "Neutral" (often leads to flat/bored prosody) | Modulates pitch variance and dynamic range. |
| **Persona/Age** | Child, Teen, Young Adult, Middle-aged, Senior | "Loli" (immature female), "Uncle" (mature male), Broadcaster, Storyteller | - | Sets the baseline formant structure. |
| **Style/Role** | Newscaster, Audiobook Narrator, Assistant, Villain | Professional, Casual, Formal, Whispering, Shouting | - | Dictates prosodic rhythm and intonation patterns. |

### 3.2 Constructing the Optimal Prompt Syntax

Randomly ordering adjectives can dilute the conditioning vectors. The most effective prompts follow a hierarchical structure: **[Physiology] + +**.

**Recommended Syntax:**

[Gender/Age] voice with texture. Speaking in a [Emotion] tone at pace with [Pitch] pitch.

#### High-Fidelity Prompt Examples and Analysis

Based on community testing and documentation, the following prompt structures yield the highest stability.5

**1. The Professional Documentary Narrator**

*Prompt:* "A deep, magnetic middle-aged male voice with low pitch and deliberate speed. The tone is serious, professional, and authoritative, suitable for a documentary narration. The voice is rich and resonant."

*Analysis:* Keywords like "magnetic" and "rich" trigger specific harmonic structures in the lower frequencies. "Deliberate speed" prevents the rushing artifacts common in long sentences.

**2. The Energetic AI Assistant**

*Prompt:* "A cheerful young female voice with slightly high pitch and fast speaking rate. The tone is lively, energetic, and bright, creating a friendly and helpful atmosphere. Clear articulation."

*Analysis:* "Slightly high" is safer than "High," which can sometimes trigger screeching artifacts. "Clear articulation" helps in reducing phoneme slurring.

**3. The Character Actor (High Expressiveness)**

*Prompt:* "A raspy, seasoned senior male voice. He speaks in a slow, incredulous tone, with a hint of panic creeping into his voice, as if witnessing something impossible."

*Analysis:* This complex prompt tests the model's ability to layer emotions ("incredulous" + "panic"). The "seasoned" descriptor adds texture (jitter/shimmer) to the voice without causing quality degradation.

**4. The Dialect Specification (Chinese/English)**

*Prompt:* "A young male voice with a Beijing dialect (erhua). The timbre is clear and natural, speaking in a casual, storytelling style." *Analysis:* For Chinese generation, specifying dialect (Beijing vs. Sichuan) is supported and highly effective. For English, "British accent" or "American accent" should be explicitly stated at the beginning.5

### 3.3 The "Negative Prompting" Fallacy

Unlike image generation models (e.g., Stable Diffusion), Qwen3-TTS does not support a dedicated "negative prompt" field where users can input "no background noise" or "no static." Including these phrases in the main instruct prompt (e.g., "A voice with no background noise") is generally ineffective and can sometimes confuse the attention mechanism. Instead, users should positively describe the desired acoustic environment: "Studio quality," "Soundproof room," "Crystal clear audio," or "Close-microphone recording".17

## 4. Hyperparameter Optimization for mlx-audio

While the prompt defines the *identity* of the voice, the generation hyperparameters define the *stability* of the audio. In mlx-audio, these parameters are passed during the generation phase via the generate\_voice\_design or generate functions. The default settings in many scripts are derived from text LLM defaults, which are often catastrophic for Audio LMs.

### 4.1 The Sensitivity of Sampling Parameters

Data suggests that Qwen3-TTS is highly sensitive to temperature and repetition\_penalty. A minor deviation in these values can determine the difference between human-like speech and a feedback loop.

#### Table 2: Recommended Generation Parameters for Stability vs. Expressivity

13

| **Parameter** | **Default (High Risk)** | **Optimal (Stability)** | **Optimal (Expressivity)** | **Technical Mechanism & Effect on Audio** |
| --- | --- | --- | --- | --- |
| **Temperature** | 1.0 / 0.9 | **0.6 - 0.7** | 0.8 - 0.9 | Controls the entropy of the sampling distribution. High values (>0.9) increase the likelihood of selecting low-probability tokens, which in audio manifests as spectral noise, static, or screaming. Low values (<0.5) collapse the prosody, making the voice sound robotic or monotone. |
| **Top\_P (Nucleus)** | 1.0 | **0.85 - 0.9** | 0.95 | Filters the cumulative probability tail. Lowering this to 0.85 cuts off the "long tail" of unlikely acoustic tokens, significantly reducing the chance of random glitches or spectral artifacts. |
| **Repetition Penalty** | 1.0 (None) | **1.05 - 1.1** | 1.02 | **Crucial for fixing beeping.** The 12Hz tokenizer is prone to looping. A penalty >1.0 discourages the model from repeating the same acoustic token, effectively breaking "infinite beep" cycles. |
| **Max New Tokens** | 256/512 | **2048 - 4096** | 4096 | Sets the generation ceiling. The default is often too low for audio, causing sentences to be cut off mid-word. Audio requires many more tokens than text. |
| **Top\_K** | 50 | **40 - 50** | 50 | Limits sampling to the K most likely tokens. Less critical than Top\_P for audio but helps stabilize the selection pool. |

### 4.2 The "Beep of Death" Pathology and Fix

The "infinite beep" is the most reported artifact for Qwen3-TTS users.2

* **Pathology:** This occurs when the model enters a low-entropy state where the highest probability next-token is identical to the previous token. In text, this looks like "the the the the." In the discrete audio codebook, a repeated token corresponds to a repeating waveform cycle, which the human ear perceives as a steady tone or beep.
* **The Fix:** By raising repetition\_penalty slightly above 1.0 (e.g., 1.05 or 1.1), the probability of the *same* token being selected is artificially lowered. This forces the model to choose a *different* token, effectively moving the audio forward and preventing the loop.13

### 4.3 Managing Spectral Distortion (The "Static" Issue)

Spectral distortion—often described as "fried" audio, background hissing, or metallic artifacts—is usually a symptom of temperature being too high.3

* **Mechanism:** At High Temp (1.0), the model occasionally samples a token that represents a high-frequency noise component or a discordant sound that doesn't fit the harmonic context.
* **The Fix:** Lowering top\_p to 0.85 is more effective than just lowering temperature. It hard-cuts the "garbage" tokens from the selection pool while preserving the diversity of the "good" tokens, maintaining expressiveness without the risk of static.19

## 5. The mlx-audio Ecosystem: Constraints and Configuration

Running Qwen3-TTS on Apple Silicon via mlx-audio requires navigating specific constraints related to the Metal backend.

### 5.1 Model Variants: 0.6B vs 1.7B

Qwen3-TTS is released in two primary sizes: 0.6B (600 million parameters) and 1.7B (1.7 billion parameters).5

* **0.6B Model:**
  + *Pros:* Extremely fast, runs on 8GB RAM MacBook Airs.
  + *Cons:* High hallucination rate. Frequently ignores VoiceDesign instructions (e.g., generates a male voice when prompted for female). Prone to language switching artifacts.
  + *Best Use:* Rapid prototyping of text scripts, low-latency checking.
* **1.7B Model:**
  + *Pros:* High fidelity, adheres strictly to VoiceDesign prompts, richer prosody.
  + *Cons:* Requires ~4.5GB VRAM (in 8-bit/bf16). Slower inference.
  + *Recommendation:* For any production use case, the **1.7B model is mandatory**. The 0.6B model's instability makes it unsuitable for generating final assets.5

### 5.2 Quantization: The 4-bit vs. 8-bit Debate

mlx-audio allows for easy quantization to reduce memory footprint.

* **4-bit Quantization:** While efficient, analysis shows that 4-bit quantization degrades the *VoiceDesign* capability. The fine-grained weights required to map a text description (e.g., "raspy") to acoustic features are often lost, resulting in generic-sounding voices or "spectral mud."
* **Recommended Config:** Use **8-bit** or **bf16** (unquantized) for the 1.7B model if you have 16GB+ Unified Memory. Use 4-bit only if strictly limited by hardware (8GB RAM machines).12

### 5.3 The "Fix Mistral Regex" Error

Users often encounter a frightening error in the console: The tokenizer you are loading... with an incorrect regex pattern... You should set the fix\_mistral\_regex=True flag.26

* **Analysis:** This is a legacy warning from the transformers library regarding the tokenization logic for Mistral-based architectures (which Qwen shares similarities with).
* **Impact:** Extensive community testing confirms this is largely a **benign warning** in the context of mlx-audio. Audio generation proceeds correctly despite the message.
* **Resolution:** Users can safely ignore this warning. If generation fails, it is likely due to memory limits or parameter settings, not this regex issue.

## 6. Implementation Guide: Python and CLI Workflows

This section provides validated code patterns for running Qwen3-TTS VoiceDesign on macOS, incorporating the stabilization fixes discussed.

### 6.1 Installation

The mlx-audio library is best installed via pip or uv. The huggingface\_hub is required for model downloading.

Bash

pip install -U mlx-audio huggingface\_hub  
# OR using uv for isolated environments  
uv tool install --force mlx-audio

### 6.2 Python Script for Robust Generation (Best Practices Applied)

This script demonstrates the optimal configuration: loading the 1.7B bf16 model, defining a structured prompt, and applying the critical repetition\_penalty and max\_new\_tokens fixes.

Python

from mlx\_audio.tts.utils import load\_model  
from mlx\_audio.tts.generate import generate\_audio  
import mlx.core as mx  
  
# 1. Select the Robust Model (Avoid 4-bit if RAM permits)  
MODEL\_PATH = "mlx-community/Qwen3-TTS-12Hz-1.7B-VoiceDesign-bf16"   
  
# 2. Define the Text and Structured Prompt  
TEXT\_INPUT = (  
 "The inherent fragility of discrete audio tokenization requires "  
 "careful management of the probability distribution."  
)  
# Note the structure: Gender/Age -> Texture -> Tone -> Speed  
INSTRUCT\_PROMPT = (  
 "A clear, young American female voice with smooth timbre. "  
 "She speaks in a cheerful and professional tone at a medium speed. "  
 "Studio quality recording."  
)  
  
def generate\_robust\_audio():  
 print(f"Loading model: {MODEL\_PATH}...")  
 model = load\_model(MODEL\_PATH)  
   
 print("Generating audio with stabilized parameters...")  
   
 # 3. Apply Hyperparameter Fixes  
 generation\_args = {  
 "text": TEXT\_INPUT,  
 "instruct": INSTRUCT\_PROMPT,  
 "language": "English",  
 # CRITICAL: Fix truncation  
 "max\_new\_tokens": 4096,   
 # CRITICAL: Fix static/distortion (Standard is 1.0)  
 "temperature": 0.7,   
 # CRITICAL: Fix glitches/hallucinations (Standard is 1.0)  
 "top\_p": 0.9,   
 # CRITICAL: Fix "Beep of Death" loop (Standard is 1.0)  
 "repetition\_penalty": 1.05,   
 "verbose": True  
 }  
  
 # Execute generation  
 # The API routes VoiceDesign requests automatically when 'instruct' is present  
 results = list(model.generate\_voice\_design(\*\*generation\_args))  
   
 if results:  
 audio\_data = results.audio  
 print(f"Generated {audio\_data.size} samples.")  
 # Save logic (e.g. soundfile.write) would go here  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 generate\_robust\_audio()

13

### 6.3 CLI Usage for Quick Prototyping

For users preferring the command line, the following command applies the necessary flags. Note that some older versions of the CLI may not expose repetition\_penalty directly; in such cases, the Python API is preferred.

Bash

python -m mlx\_audio.tts.generate \  
 --model mlx-community/Qwen3-TTS-12Hz-1.7B-VoiceDesign-bf16 \  
 --text "System status: Nominal." \  
 --instruct "A robotic, deep male voice with metallic texture and monotone pitch." \  
 --temperature 0.7 \  
 --repetition\_penalty 1.05 \  
 --output\_path system\_check.wav

12

## 7. Troubleshooting Common Artifacts: A Diagnostic Guide

Even with optimal settings, artifacts may occur due to the stochastic nature of the model.

### 7.1 Artifact: The "Beep of Death" (Infinite Loop)

* **Symptom:** Audio starts fine, then devolves into a loud continuous tone or rhythmic beeping at the end.
* **Diagnosis:** Autoregressive loop. The model failed to predict the <stop> token.
* **Advanced Fixes:**
  1. **Increase Repetition Penalty:** Move from 1.05 to 1.1 or 1.2.
  2. **Sentence Padding:** Add a dummy phrase at the end of the text (e.g., "... end of message.") and trim the audio later. The model struggles to end on "open" phonemes.
  3. **Seed Rotation:** Change the random seed. Some seeds are simply "cursed" for specific text combinations.32

### 7.2 Artifact: Slurring or "Drunken" Speech

* **Symptom:** The speaker sounds intoxicated, words slur together, or speed varies wildly.
* **Diagnosis:** Temperature is too low (<0.4) or Top\_P is too aggressive (<0.6). The model is constrained to a very narrow path of tokens that may not be acoustically viable.
* **Fix:** Increase temperature to 0.7 or 0.8 to allow for natural prosodic variance. Ensure the prompt specifies "Clear articulation".14

### 7.3 Artifact: Language Switching / Accent Drift

* **Symptom:** An English prompt generates Chinese-accented English, or randomly switches to Chinese characters.
* **Diagnosis:** Latent bleed from the training data (which is heavily Chinese/English). This is common in the **0.6B model**.
* **Fix:**
  1. Switch to the **1.7B model**.
  2. Explicitly state the language/accent in the prompt: "American English accent."
  3. Ensure the input text contains only English characters (no accidental Unicode artifacts).5

### 7.4 Artifact: Audio Truncation

* **Symptom:** The last few words are cut off.
* **Diagnosis:** max\_new\_tokens limit reached.
* **Fix:** Set max\_new\_tokens to at least 150 per second of expected audio. For a 10-second clip, you need ~1500 tokens (including overhead). Setting it to 4096 is a safe default for most paragraph-length generations.13

## 8. Advanced Use Cases and Workflows

### 8.1 Long-Form Audiobook Generation

Generating entire chapters requires a strategy to prevent "drift" (where the voice changes over time) or context window overflow.

* **Strategy: Chunking with State Reset.** Do not feed a 5000-word chapter at once.
  + Split text into paragraphs or sentences.
  + Generate each chunk independently using the **same prompt** and **same seed**.
  + Concatenate the audio files using ffmpeg.
* **Why:** Autoregressive models accumulate error probability over time. Resetting the generation for each paragraph ensures the voice remains consistent and the probability of a "beep loop" is minimized.5

### 8.2 Hybrid Voice Cloning

While VoiceDesign creates voices from text, Qwen3 also supports cloning.

* **Technique:** In mlx-audio, you can theoretically pass *both* a reference audio file (--ref\_audio) *and* an instruct prompt (--instruct) to the Base model.
* **Benefit:** The reference audio provides the timbre grounding, while the text prompt controls the prosody and emotion. This "Hybrid Cloning" can fix clones that sound flat or emotionless.5

### 8.3 Cross-Lingual Synthesis

Qwen3 supports 10 languages.1

* **Prompting Strategy:** When generating non-English audio (e.g., Spanish), the prompt should ideally be in English (the model's strongest instruction language) but specify the target language physiology: "A native Spanish male voice..."
* **Input Text:** Ensure the text input matches the target language. The model does not translate; it synthesizes.

## 9. Conclusion and Future Outlook

The Qwen3-TTS VoiceDesign model represents a significant milestone in open-source audio AI, offering a level of control previously unavailable on consumer hardware. Its integration with mlx-audio unlocks this power for Apple Silicon users, provided they navigate the intricacies of discrete tokenization and quantization.

The analysis conclusively demonstrates that the default "out-of-the-box" settings for mlx-audio are often insufficient for stable production workflows. The "beeping" and distortion artifacts are not insurmountable bugs but rather symptoms of unoptimized sampling in the 12Hz token space. By enforcing a **Repetition Penalty of ~1.05**, lowering **Temperature to 0.7**, employing **8-bit or bf16** precision, and utilizing structured **Role-Attribute-Style** prompts, users can achieve studio-grade results.

As the ecosystem matures, we anticipate the release of 25Hz tokenizers and further optimizations to the MLX backend that will likely mitigate the need for such aggressive manual tuning. Until then, the protocols outlined in this report serve as the definitive standard for high-fidelity Qwen3-TTS generation on macOS.

#### Works cited

1. Qwen3-TTS Steps Up: Voice Cloning and Voice Design!, accessed February 2, 2026, <https://qwen.ai/blog?id=qwen3-tts-vc-voicedesign>
2. Zonos-v0.1 beta by Zyphra, featuring two expressive and real-time text-to-speech (TTS) models with high-fidelity voice cloning. 1.6B transformer and 1.6B hybrid under an Apache 2.0 license. : r/LocalLLaMA - Reddit, accessed February 2, 2026, <https://www.reddit.com/r/LocalLLaMA/comments/1imdnap/zonosv01_beta_by_zyphra_featuring_two_expressive/>
3. BeltOut: An open source pitch-perfect (SINGING!@#$) voice-to-voice timbre transfer model based on ChatterboxVC - Reddit, accessed February 2, 2026, <https://www.reddit.com/r/StableDiffusion/comments/1ls5jqq/beltout_an_open_source_pitchperfect_singing/>
4. Qwen3-TTS Family is Now Open Sourced: Voice Design, Clone, and Generation!, accessed February 2, 2026, <https://qwen.ai/blog?id=qwen3tts-0115>
5. Qwen3-TTS: The Complete 2026 Guide to Open-Source Voice Cloning and AI Speech Generation | by cheng zhang - Medium, accessed February 2, 2026, <https://medium.com/@zh.milo/qwen3-tts-the-complete-2026-guide-to-open-source-voice-cloning-and-ai-speech-generation-1a2efca05cd6>
6. Qwen/Qwen3-TTS-12Hz-1.7B-VoiceDesign · Hugging Face, accessed February 2, 2026, <https://huggingface.co/Qwen/Qwen3-TTS-12Hz-1.7B-VoiceDesign>
7. Qwen for Audio AI. How to use Qwen3-TTS for free? | by Mehul Gupta | Data Science in Your Pocket | Jan, 2026 | Medium, accessed February 2, 2026, <https://medium.com/data-science-in-your-pocket/qwen3-tts-qwen-for-audio-ai-712f6f28450c>
8. Qwen3-TTS Technical Report - arXiv, accessed February 2, 2026, <https://arxiv.org/html/2601.15621v1>
9. Qwen3-TTS is an open-source series of TTS models developed by the Qwen team at Alibaba Cloud, supporting stable, expressive, and streaming speech generation, free-form voice design, and vivid voice cloning. - GitHub, accessed February 2, 2026, <https://github.com/QwenLM/Qwen3-TTS>
10. Research | Radical Data Science - WordPress.com, accessed February 2, 2026, <https://radicaldatascience.wordpress.com/category/research/>
11. MLX Community - Hugging Face, accessed February 2, 2026, <https://huggingface.co/mlx-community>
12. kyr0/qwen3-TTS-12Hz-0.6B-Base-4bit-partial-quantization at main - Hugging Face, accessed February 2, 2026, <https://huggingface.co/kyr0/qwen3-TTS-12Hz-0.6B-Base-4bit-partial-quantization/tree/main>
13. generation\_config.json · mlx-community/Qwen3-TTS-12Hz-0.6B-CustomVoice-8bit at main, accessed February 2, 2026, <https://huggingface.co/mlx-community/Qwen3-TTS-12Hz-0.6B-CustomVoice-8bit/blob/main/generation_config.json>
14. Qwen-TTS voice design API reference - Alibaba Cloud Model Studio ..., accessed February 2, 2026, <https://www.alibabacloud.com/help/en/model-studio/qwen-tts-voice-design>
15. mlx-community/Qwen3-TTS-12Hz-1.7B-VoiceDesign-5bit - Hugging Face, accessed February 2, 2026, <https://huggingface.co/mlx-community/Qwen3-TTS-12Hz-1.7B-VoiceDesign-5bit>
16. TTS Audio Suite v4.19 - Qwen3-TTS with Voice Designer : r/StableDiffusion - Reddit, accessed February 2, 2026, <https://www.reddit.com/r/StableDiffusion/comments/1qqugpl/tts_audio_suite_v419_qwen3tts_with_voice_designer/>
17. liuhuadai/AudioCoT · Datasets at Hugging Face, accessed February 2, 2026, <https://huggingface.co/datasets/liuhuadai/AudioCoT>
18. GLM-4.7-Flash: The Ultimate 2026 Guide to Local AI Coding Assistant - Medium, accessed February 2, 2026, <https://medium.com/@zh.milo/glm-4-7-flash-the-ultimate-2026-guide-to-local-ai-coding-assistant-93a43c3f8db3>
19. deepseek/deepseek-r1-0528-qwen3-8b stuck on infinite tool loop. Any ideas? - Reddit, accessed February 2, 2026, <https://www.reddit.com/r/LocalLLaMA/comments/1l02hmq/deepseekdeepseekr10528qwen38b_stuck_on_infinite/>
20. mlx-community/LFM2.5-Audio-1.5B-fp16 - Hugging Face, accessed February 2, 2026, <https://huggingface.co/mlx-community/LFM2.5-Audio-1.5B-fp16>
21. Audio Flamingo 2 - NVIDIA ADLR, accessed February 2, 2026, <https://research.nvidia.com/labs/adlr/AF2/>
22. FIXED: Random Python / ntdll.dll crashes (inlcuding PC freezings!) with extreme memory spike / memory leak, after Update from "WanGP v7.2" to "WanGP v7.7777" - Using Wan 2.2 / Qwen models · Issue #743 · deepbeepmeep/Wan2GP - GitHub, accessed February 2, 2026, <https://github.com/deepbeepmeep/Wan2GP/issues/743?timeline_page=1>
23. Upload folder using huggingface\_hub · mlx-community/Qwen3-TTS, accessed February 2, 2026, <https://huggingface.co/mlx-community/Qwen3-TTS-12Hz-0.6B-CustomVoice-8bit/commit/797a760ef9974b36992078ff9baf957e91de8ded>
24. QwenLM | Qwen3-TTS - Kaggle, accessed February 2, 2026, <https://www.kaggle.com/models/qwen-lm/qwen3-tts>
25. mlx-community/Qwen3-TTS-12Hz-0.6B-Base-4bit - Hugging Face, accessed February 2, 2026, <https://huggingface.co/mlx-community/Qwen3-TTS-12Hz-0.6B-Base-4bit>
26. Issues · Blaizzy/mlx-audio · GitHub, accessed February 2, 2026, <https://github.com/Blaizzy/mlx-audio/issues>
27. Got error "fix\_mistral\_regex=True " on Qwen3-TTS-12Hz-1.7B-VoiceDesign-bf16 · Issue #413 · Blaizzy/mlx-audio - GitHub, accessed February 2, 2026, <https://github.com/Blaizzy/mlx-audio/issues/413>
28. mlx-audio/mlx\_audio/tts/models/qwen3\_tts/README.md at main - GitHub, accessed February 2, 2026, <https://github.com/Blaizzy/mlx-audio/blob/main/mlx_audio/tts/models/qwen3_tts/README.md>
29. mlx-community/Qwen3-TTS-12Hz-1.7B-VoiceDesign-bf16 · Hugging ..., accessed February 2, 2026, <https://huggingface.co/mlx-community/Qwen3-TTS-12Hz-1.7B-VoiceDesign-bf16>
30. tools/python/q3\_tts.py at main · simonw/tools - GitHub, accessed February 2, 2026, <https://github.com/simonw/tools/blob/main/python/q3_tts.py>
31. Qwen3-TTS family is now open sourced: Voice design, clone, and generation | Hacker News, accessed February 2, 2026, <https://news.ycombinator.com/item?id=46719229>
32. Simon Willison on qwen, accessed February 2, 2026, <https://simonwillison.net/tags/qwen/>
33. Create long audiobooks with custom voices using Qwen3-TTS voice design - N8N, accessed February 2, 2026, <https://n8n.io/workflows/13030-create-long-audiobooks-with-custom-voices-using-qwen3-tts-voice-design/>
34. Pure Rust implementation of Qwen3-TTS speech synthesis - GitHub, accessed February 2, 2026, <https://github.com/TrevorS/qwen3-tts-rs>