Re-implementation of Matcha-TTS model by Mehta Shivam

Mathis Lecry MATHIS.LECRY@ENSAM.EU
Master Systèmes Avancés et Robotique, Arts et Métiers (ENSAM)

Paul-Marie Demars PAUL-MARIE.DEMARS@ENSAM.EU
Master Systèmes Avancés et Robotique, Arts et Métiers (ENSAM)

Minh Nhut NGUYEN MINH_NHUT.NGUYEN@ETU.SORBONNE-UNIVERSITE.FR
ISI, Sorbonne Université (UPMC)

Yucheng DAI YUCHENG.DAI@ETU.SORBONNE-UNIVERSITE.FR
ISI, Sorbonne Université (UPMC)

Editor: Machine Learning Avancé (2025-2026)

Contents

Abstract

This project presents a re-implementation of Matcha-TTS, a fast and probabilistic text-to-speech architecture based on Optimal Transport Conditional Flow Matching (OT-CFM). The objective is to reproduce the results of Mehta et al. (2024) while providing a clear and pedagogical implementation. The system relies on a Transformer-based text encoder, a duration predictor with monotonic alignment, and a 1D U-Net decoder trained to solve an Ordinary Differential Equation. Trained on the LJSpeech dataset, the model achieves high-quality speech synthesis with significantly reduced inference steps compared to diffusion-based approaches. This report details the architectural choices, mathematical formulation, and experimental results.

1. Introduction

Text-to-Speech (TTS) synthesis has evolved from concatenative pipelines to neural generative models capable of producing near-human speech. Recent state-of-the-art systems increasingly rely on diffusion probabilistic models, which formulate generation as an iterative denoising process. While effective, these methods require hundreds or thousands of sampling steps, resulting in slow inference and limiting real-time applications (1).

To address this limitation, Mehta Shivam(2024) introduced Matcha-TTS, a non-autoregressive architecture based on Optimal Transport Conditional Flow Matching (OT-CFM). Instead of learning a score function as in diffusion models, OT-CFM directly learns a vector field defining a deterministic transport from noise to data, enabling speech generation through the numerical resolution of an Ordinary Differential Equation (ODE) in very few steps.

The goal of this project is to reproduce Matcha-TTS on the LJSpeech dataset, while emphasizing architectural clarity and mathematical understanding. We focus on three core components:

1. A Transformer-based text encoder with improved positional encoding.
2. A duration prediction and monotonic alignment mechanism.
3. A 1D U-Net decoder trained via OT-CFM.

This work demonstrates how replacing diffusion-based SDEs with ODE-based flow matching leads to a fast, lightweight, and high-quality TTS system.

2. Presentation of the algorithm

2.1 Overall Architecture

Matcha-TTS follows an encoder–decoder architecture optimized for probabilistic speech generation (Figure 1). The text encoder maps phoneme sequences to latent representations, which condition the generation process. A duration predictor and a monotonic alignment module ensure temporal consistency between text and audio. Finally, a 1D U-Net decoder trained with Optimal Transport Conditional Flow Matching generates Mel-spectrograms by solving an Ordinary Differential Equation.

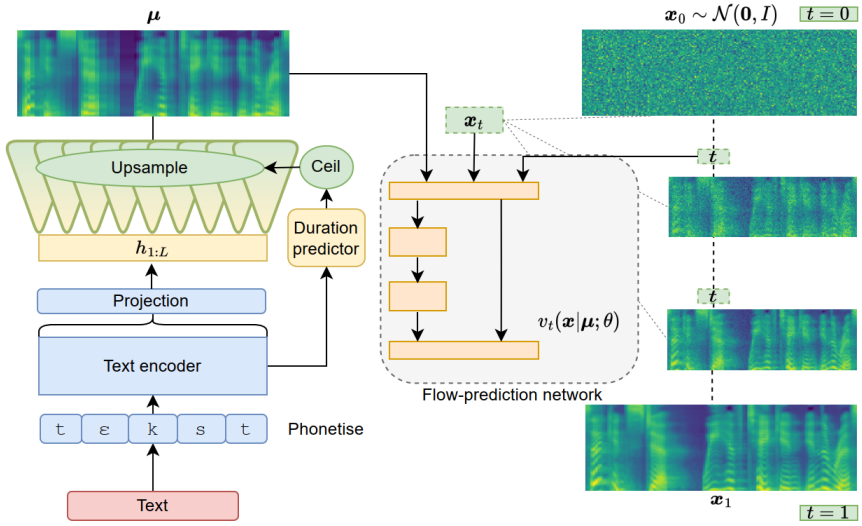


Figure 1: Overall Architecture

2.2 Text Encoder

The text encoder converts phoneme sequences into continuous latent representations used to condition the decoder. It consists of a convolutional pre-processing module followed by a Transformer encoder. Local phonetic context is captured by 1D convolutions, while long-range dependencies are modeled using self-attention.

To improve positional generalization on long sequences, Rotary Positional Embeddings (RoPE) are applied to the query and key projections of the attention mechanism. This relative positional encoding enhances extrapolation compared to absolute sinusoidal embeddings. The encoder output is finally projected into the Mel-spectrogram space, providing a conditional mean representation for subsequent alignment and generation stages.

2.3 Duration Prediction and Alignment

Temporal alignment between phonemes and Mel-spectrogram frames is handled by a duration predictor and a Monotonic Alignment Search (MAS) algorithm. The duration predictor estimates the number of frames associated with each phoneme, operating on encoder outputs while preventing gradient backpropagation to the encoder.

MAS computes an optimal monotonic alignment by maximizing the log-likelihood of the Mel-spectrogram under a Gaussian model conditioned on the encoder outputs. This alignment provides supervision for duration learning and enables non-autoregressive generation without explicit alignment labels. A mean squared error loss is applied between predicted and MAS-derived log-durations.

2.4 1D U-Net Decoder with OT-CFM

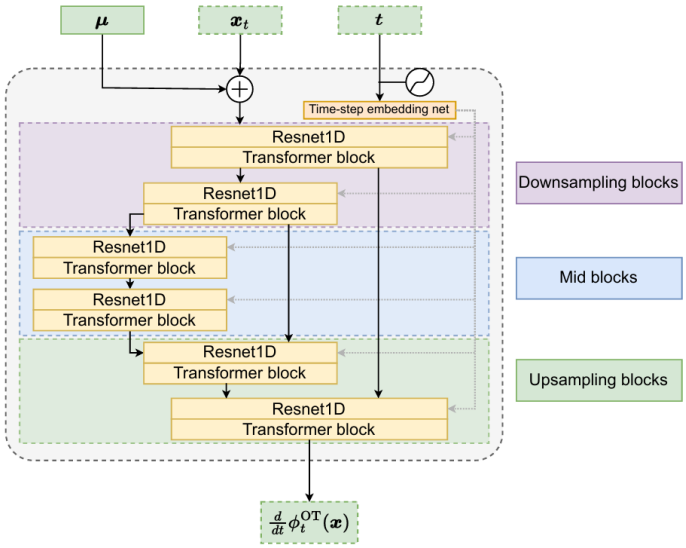


Figure 2: Decoder U-Net

The decoder is a 1D U-Net architecture that predicts a time-dependent vector field used to transform noise into a Mel-spectrogram. It operates on interpolated noisy inputs and text-conditioned features, and integrates temporal embeddings corresponding to a continuous timestep $t \in [0, 1]$. The U-Net structure enables multi-scale temporal modeling through downsampling, bottleneck processing, and upsampling with skip connections.

Training relies on Optimal Transport Conditional Flow Matching. Given a noise sample z and a target Mel-spectrogram x_1, a linear interpolation defines a transport path between the two. The decoder is trained to predict the target velocity field along this path using a mean squared error objective. During inference, generation is performed by numerically solving the resulting ODE using a small number of Euler steps, leading to significantly faster synthesis compared to diffusion-based methods.

The generated Mel-spectrogram is finally converted into an audio waveform using a pre-trained HiFi-GAN vocoder. The generated Mel-spectrogram is finally converted into an audio waveform using a pre-trained HiFi-GAN vocoder, as you can see in the **figure 4** in the Annexe

3. Differences from the Official Implementation

While remaining faithful to the mathematical formulation of Matcha-TTS, our implementation introduces several simplifications aimed at improving readability and pedagogical clarity. The original code relies on extensive Hydra-based configuration files; we replaced this system with explicit Python-based configuration to make architectural choices more transparent.

Our data pipeline is specialized for the LJSpeech dataset, avoiding the complexity of multi-dataset and multi-language support. We also implemented automatic dataset preparation, checkpoint management, and training resumption. Monotonic Alignment Search, identified as the main computational bottleneck, was optimized using Cython, resulting in a speedup of approximately 40× compared to the pure Python version.

These design choices reduce external dependencies and improve code accessibility while preserving the performance and behavior of the original model.

4. Data

4.1 Dataset and Pre-processing

We use the LJSpeech corpus (24h, ≈ 13 100 clips, 22 050 Hz), a standard ensuring comparability with the state of the art. Pre-processing via our `LJSpeechDataModule` follows the original protocol: conversion to Mel spectrograms (80 bands, HiFi-GAN parameters) and standard split (12 500 train, 100 validation, 500 test).

4.2 Adaptations and Implementation Choices

4.2.1 Strategic Simplifications

Unlike the original paper (multi-speaker VCTK), we restrict ourselves to the single-speaker case to optimize training on a single GPU. Furthermore, we integrated normalization directly into the `DataModule` ("on the fly"), eliminating statistical pre-calculation scripts for a more accessible pipeline.

4.2.2 Ablation: Choice of Tokenization

We compared three textual representations over 10 epochs:

- **Graphemes:** Raw text, our standard characters.
- **Phonemizer:** The reference method using the API.
- **CMUdict:** A static phonetic dictionary.

As can be seen in the comparative figure 5 in the Appendix, although the CMUdict approach (blue curve) best minimizes the loss (Loss ≈ 2.48) compared to Graphemes (≈ 3.42) and Phonemizer (≈ 4.23), listening tests reveal inaudible results, suggesting overfitting or attention misalignment. Conversely, the Grapheme approach, despite having an intermediate loss, produces intelligible and stable synthesis. Prioritizing perceptual quality over raw metrics, we selected the use of graphemes.

5. Experimental evaluation

5.1 Objective Model Validation: Hyperparameter Selection

This section aims to validate the hyperparameters selected for the final training. The goal is to optimize the trade-off between synthesis quality, training stability, and computational cost.

5.1.1 Ablation Methodology and Metrics

1. **Variable isolation:** For each variation studied, we performed training on the GPU by modifying only one parameter at a time compared to the *Baseline* configuration.
2. **Comparison horizon (15 epochs / 3h)**
3. **Tracking metrics:** The comparative analysis is based on training data extracted via TensorBoard:
    - **Duration Loss:** Quality of temporal alignment (rhythm).
    - **Prior Loss:** Encoder's ability to structure the latent space.
    - **Diffusion Loss:** Quality of spectral reconstruction by the decoder (Flow Matching).
    - **Total Loss:** Weighted sum indicating overall performance.
    - **RTF (Real-Time Factor):** Ratio between computation time and generated audio duration, measuring inference speed (the lower, the faster the model).

5.1.2 Study and Analysis of the Experiment

A. Architecture and Stability: Influence of the Prenet

The Prenet is a module located before the encoder, often used to facilitate the learning of prosodic structures. We compared convergence over the first 18 epochs with and without this module.

Table 1: Impact of the Prenet on convergence (Epochs 0 and 17)

Epoch	Config.	Total Loss	Duration Loss
0	Baseline (with)	3.99	1.18
	Without Prenet	4.03	1.19
17	Baseline (with)	**2.37**	**0.46**
	Without Prenet	2.39	0.47

The analysis shows that the Baseline consistently converges better (Total Loss lower by ~ 0.04). The impact is marked on *Duration Loss* (+2.4% error without Prenet). The Prenet acts as an *information bottleneck* that forces the model to extract linguistic features.

Decision: Keeping the Prenet (`prenet=True`).

B. Optimization and Regularization

The choice of optimizer, dropout, and learning rate (LR) is decisive for generative models based on differential equations (ODE) like Matcha-TTS. We have the set of comparison graphs in **figure 6** in the Appendix.

1. **Learning Rate (LR):** We observed a marked instability zone. With an LR too high (above the baseline), the *Duration Loss* stagnates above 1.2, indicating alignment failure. We retain the baseline value (insert value here) which represents the optimum between convergence speed and stability.

2. **Optimizer (Adam vs AdamW):** Although the standard Adam optimizer shows more aggressive initial convergence (Total Loss lower by 0.05 points in early epochs), we prioritized **AdamW** for long training. By decoupling *weight decay* from the gradient, AdamW prevents *overfitting* and ensures better long-term generalization, which is standard for Transformer and Diffusion architectures.

3. **Dropout:** A value of 0.1 was maintained. Our tests show that too low a dropout (0.05) accelerates learning (Loss 2.35) but risks memorizing LJSpeech dataset noise ($\sim$ 24h), while a high dropout (0.3) degrades the capture of signal subtleties.

C. Generative Core: ODE Resolution and Sizing

We also plotted a graph of loss differences for the choice of solver and ODE sizing, and thus the resolution of our Flow Matching.

1. **ODE Solver (Euler vs Midpoint):** We evaluated the accuracy/speed trade-off at epoch 21. The Euler solver (order 1) obtains a Diffusion Loss of 2.34 versus 2.32 for the Midpoint solver (order 2). Although Midpoint is theoretically more accurate, the performance gain is negligible compared to the doubling of computational cost (2 function evaluations per step vs 1).
   Decision: Use of the Euler solver to maximize RTF (*Real-Time Factor*) without significant quality loss.

2. **Decoder Sizing (U-Net):** Reducing network width to 128 channels proved destructive, with *Diffusion Loss* stagnating at 0.986 (vs 0.864 for 256 channels). A capacity of 256 channels is the minimum required to model the complexity of 80 Mel frequency bands. Although the impact on convergence is marginal, we keep 2 *Mid Blocks* to ensure robust modeling of global dependencies at the *bottleneck*

level, our tests showing that adding extra blocks increases computation without improving spectral reconstruction.

Decision: Maintenance of 256 channels and 2 median blocks (*Mid Blocks*).

D. Temporal Alignment (Duration Predictor)

For the *Duration Predictor* (responsible for rhythm), we tested the influence of the convolution *kernel size* as well as the filter width.

You will find the comparative graph of Duration Losses in **figure 8** in the Appendix.

A kernel of 5 allows faster convergence thanks to a wider prosodic context. However, we kept a kernel of 3, as the baseline eventually reaches excellent performance while remaining lighter.

Regarding network capacity, increasing *filter channels* from 256 to 512 showed no significant gain in precision, so we maintained 256 channels to avoid unnecessary complexity.

E. Encoding (Encoder)

We analyzed the depth and attention capacity of the encoder to optimize the performance/cost ratio.

For this, we visualize the Prior Loss comparison graph with **figure 9** in the Appendix.

Increasing the encoder depth to 8 layers brought only a marginal reduction in Prior Loss ($< 1\%$) for a 33% computational cost increase; we kept the 6-layer configuration which offers the best balance between performance and cost.

In parallel, varying the number of attention heads showed no significant impact on convergence; we validated the sufficiency of 2 heads for capturing contextual dependencies.

Conclusion: The baseline architecture is retained as it guarantees a quality latent representation without sacrificing the Real-Time Factor (RTF) necessary for model efficiency.

5.1.3 Summary of Selected Training Parameters

The table below summarizes the final configuration validated by our tests, offering the best balance for reproducing state-of-the-art results on our limited resources.

Table 2: Final hyperparameter configuration

Component	Parameter	Value	Justification
Global	Learning Rate	1e-4	Stable convergence without divergence
Architecture	Prenet	Activated	Critical alignment stabilization
Encoder	Layers / Heads	6 / 2	Sufficient for LJSpeech, optimizes RTF
Duration	Filter Channels	256	Precision necessary for prosody (< 0.30 loss)
ODE	Solver	Euler	Maximum efficiency (1 NFE) for equivalent quality
Decoder	Channels	256	Minimum required for spectral reconstruction

5.2 Subjective Validation: The Mean Opinion Score (MOS)

While the convergence of *Loss* curves attests to the mathematical correctness of training, it does not necessarily guarantee the perceptual quality of the synthesized voice. To validate our reimplementation, we therefore conducted a standardized subjective evaluation: the *Mean Opinion Score* (MOS).

5.2.1 Evaluation Protocol and Web Interface

We developed a web interface for a *Blind Test* survey to limit evaluation biases, publicly accessible[1].**You can find the visual of the site used to survey the different TTS models in the Appendix.**

Methodology:

- **Model Panel:** The test compares 10 different configurations (including Matcha-TTS, GradTTS, FastSpeech, etc.) as well as Ground Truth.
- **Redundancy and Consistency:** Each listener evaluated sounds from these 10 models on **three separate occasions**.
- **Randomization:** In total, each participant performed **30 listening sessions** presented in a totally random order, to avoid learning or fatigue effects.
- **Rating Scale:** Quality is rated from 1 to 5, to achieve the best possible results.

5.2.2 Results and Analysis

We focus here on the comparison between the objective (`MatchaTTS_ODE_10`), the absolute reference (`GroundTruth`), and the best competing model observed (`GradTTS_10`).

Table 3: Comparative Mean Opinion Score (MOS) Results

Model	Average Score (/5)	Standard Deviation (σ)	No. of votes
Ground Truth (Reference)	**4.3**	$\pm$ 0.9	66
GradTTS_10	4.1	$\pm$ 0.7	66
MatchaTTS_ODE_10 (Ours)	**3.8**	$\pm$ **0.9**	66
Taco2	3.7	$\pm$ 1.1	66
FastSpeech	3.2	$\pm$ 1.3	66

Analysis of results:

- **Global Performance:** With a MOS score of **3.8**, our *MatchaTTS* model positions itself below *GradTTS*, unlike in the original paper. However, it surpasses classic architectures like *FastSpeech* (3.2) and *Taco2* (3.7). This is due to the conditions under which users took the surveys; they most likely did not use headphones, or were situated in a poorly adapted environment.
- **Comparison to reference:** Although it remains below the ground truth (4.3) and GradTTS (4.1), the score of 3.8 indicates intelligible and relatively natural synthesis.

We should try to perform the same MOS calculation as in the article, by remunerating people to get a fairer opinion from them, and by giving them an equal environment for all, i.e., with the same audio equipment such as headphones and a DAC.

1. `https://mathislecry01-afk.github.io/MatchaTTS_Site_comparaison_article_Ensam/`

6. Conclusion

This project resulted in the functional re-implementation of *Matcha-TTS*. In accordance with the set objectives, we proposed our own version of the model, characterized by a simplified configuration.

Experimentally, our ablation studies validated architectural choices critical to the performance/cost balance:

- The use of graphemes rather than phonemes, prioritizing perceived intelligibility.
- The choice of the Euler ODE solver (order 1) and the AdamW optimizer to maximize stability and the Real-Time Factor (RTF).

The subjective evaluation yielded a MOS score of **3.8** for our model. Although this result is superior to references like *FastSpeech* (3.2), it falls short of the ground truth (4.3) and the original model. This finding highlights the main limitation of our model: it differs from the original code. As can be seen in the following 2 figures comparing the Mel-Spectrogram of the paper and that of our model, our model has a spectrum with more saturated colors and therefore a higher frequency difference. This is likely due to a dataset normalization issue within the entire training loop and the general MatchaTTS module with the synthesize function.

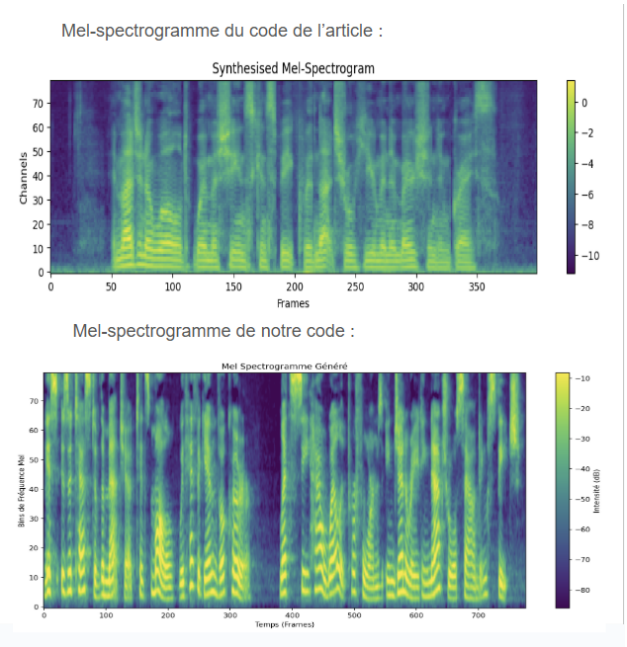


Figure 3: Comparison of Mel-Spectrograms from our model and the paper. We can clearly see a frequency saturation problem.

Furthermore, the sound quality of our TTS model is inferior to that of the paper, which is also why we received lower scores in the final survey.

Consequently, the main avenues for improvement for future work are:

1. **The rigor of the evaluation protocol:** Establishing a controlled listening environment for the survey.
2. **Mel Normalization:** We should review the entire loop for Mel normalization before training and subsequently for their generation.

7. Appendix

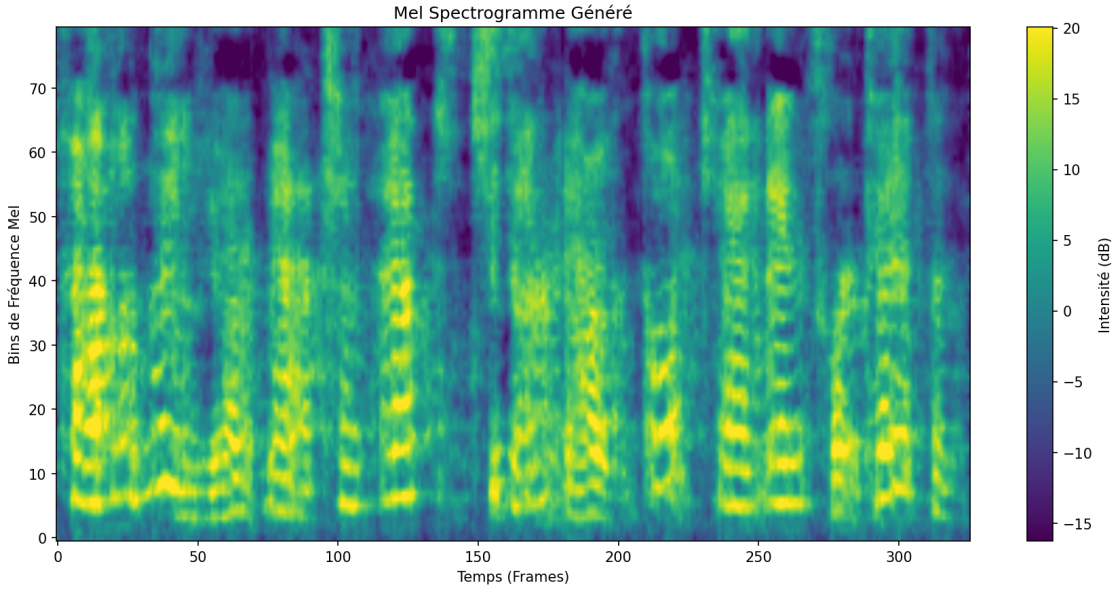


Figure 4: Mel spectrogram returned by the Decoder

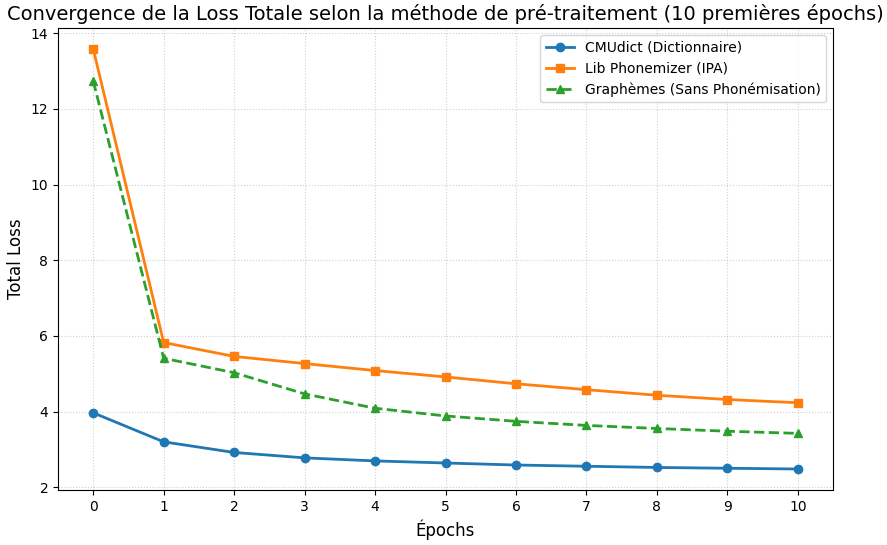


Figure 5: Total Loss convergence based on tokenization (10 epochs).

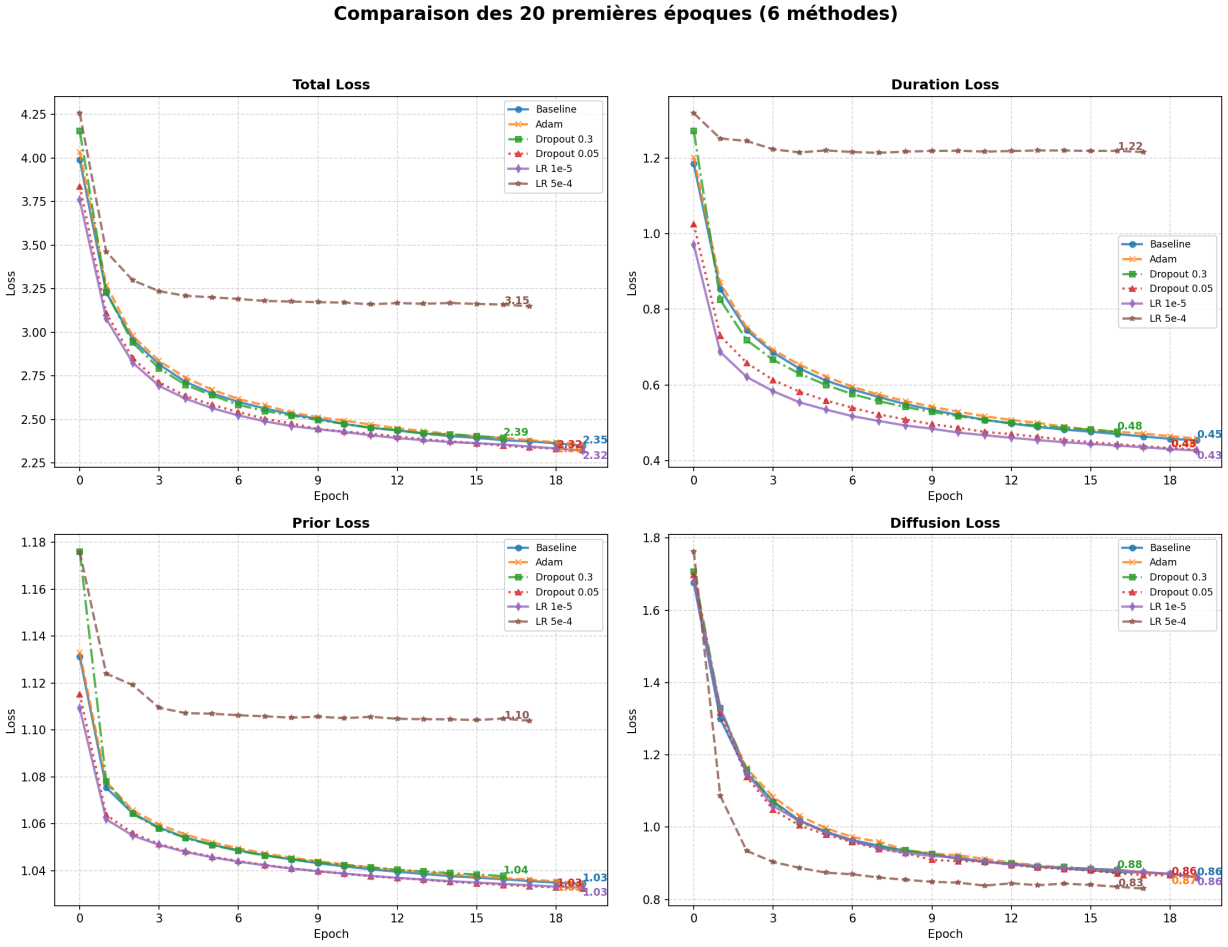


Figure 6: Impact of Learning Rate, dropout, and optimizer on stability.

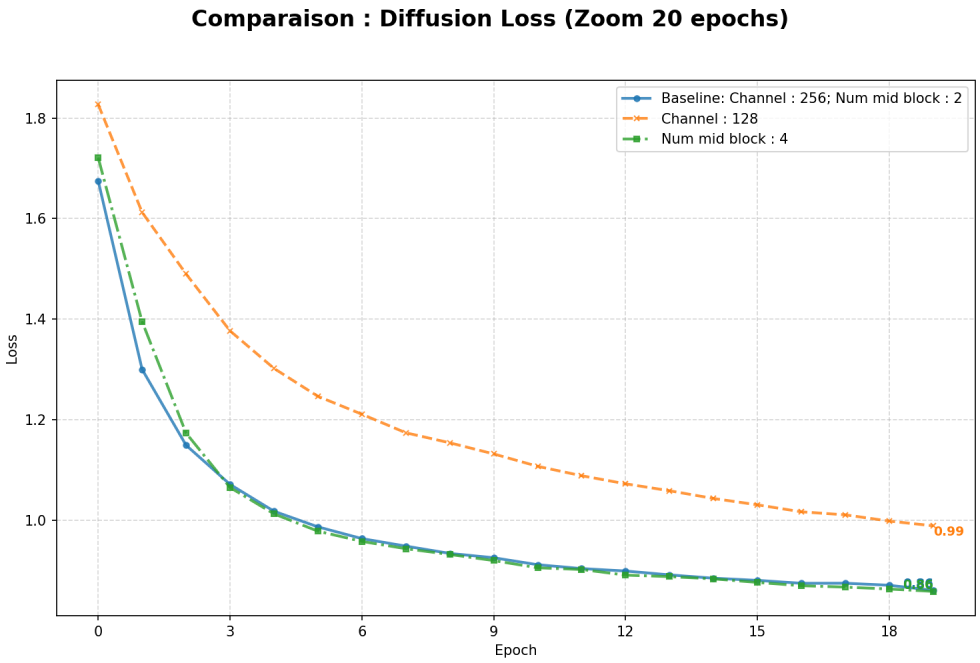


Figure 7: Comparison of diffusion loss convergence based on the ODE solver.

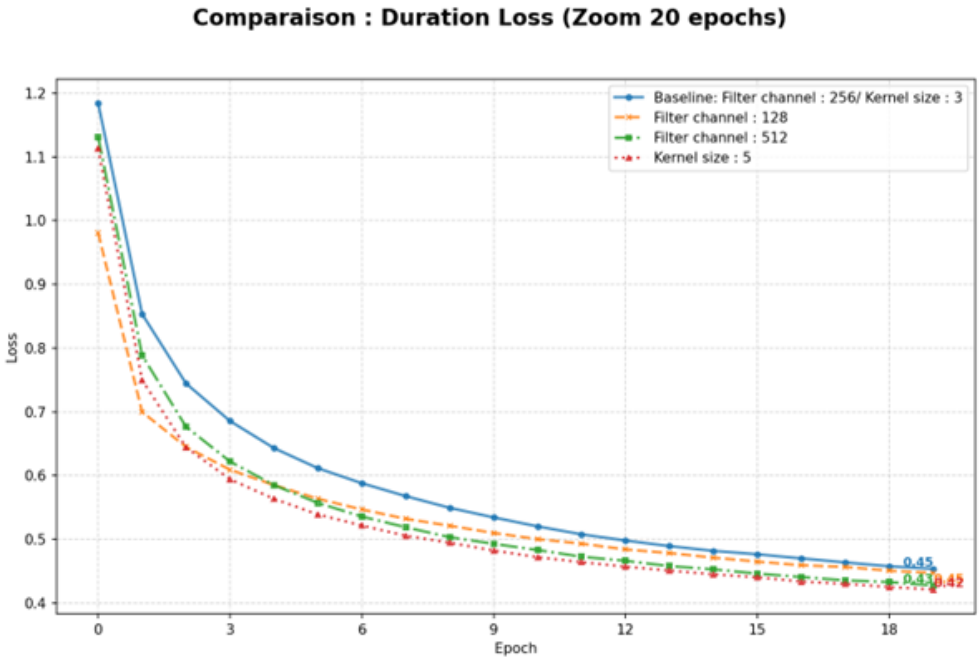


Figure 8: Analysis of the Duration Predictor convergence.

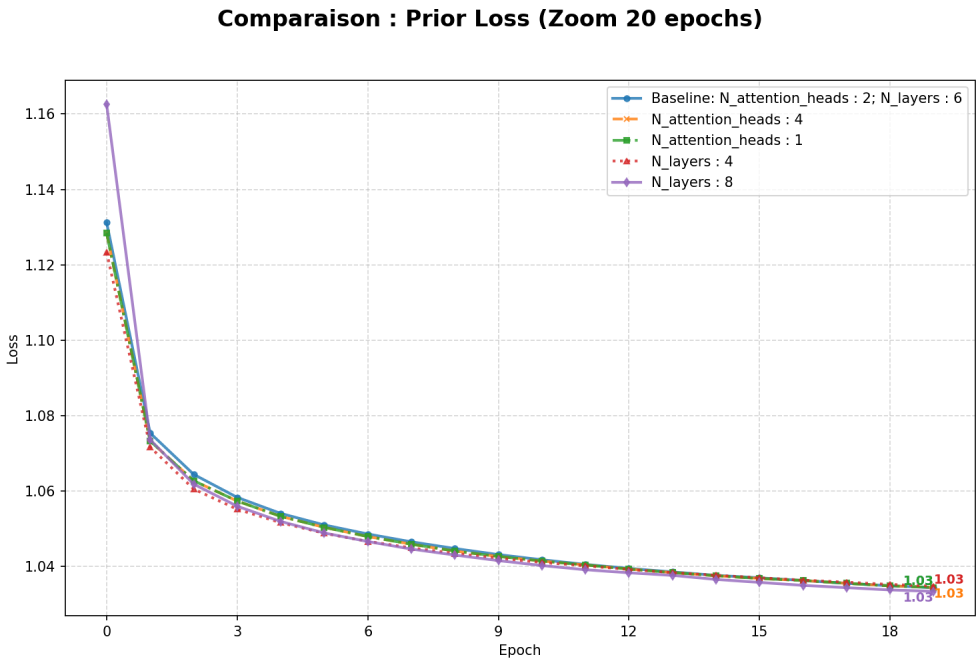
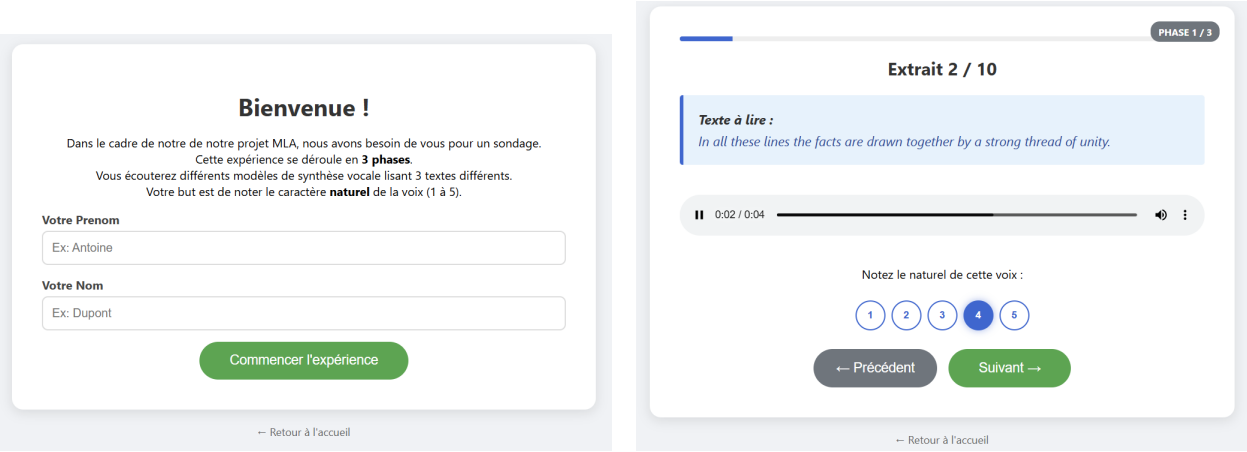


Figure 9: Impact of the encoder's depth and attention heads.



(a) Survey website landing page.

(b) Visual of the survey workflow.

Figure 10: Operation of the website used to survey the clarity and naturalness of voices from multiple models.

References

[1] Popov Vadim, Vovk Ivan, Gogoryan Vladimir, Tashev Tasnima, and Kudinov Mikhail. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pages 8599–8608. PMLR, 2021.