# FlowTurbo Implementation and Improvement

Aastha Mariam John (241040602)

November 6, 2024

## 1 Introduction

The field of image generation has seen significant advancements with the development of flow-based models, which map complex data distributions to simpler ones. **FlowTurbo** is an innovative approach that focuses on reducing inference time while preserving high image quality, enabling real-time applications. **This project's objective is to implement the FlowTurbo model and evaluate the impact of proposed modifications to enhance its performance further.**

The FlowTurbo framework, is designed to accelerate flow-based generative models, providing a substantial improvement in inference speed without compromising the quality of generated images. FlowTurbo leverages the unique properties of flow-based models, where the velocity predictor's output remains relatively stable during the sampling process. This stability enables a more efficient sampling mechanism that bypasses the traditional need for computationally intensive, iterative model evaluations

## 2 Background and Paper Summary

The FlowTurbo paper, authored by Wenliang Zhao et al. and presented at NeurIPS 2024, introduces a flow-based model for image generation. The primary contributions include:

- A lightweight **velocity refiner** for efficient sampling, reducing the required steps in image generation.
- **Pseudo corrector** and **sample-aware compilation** methods to achieve significant reductions in latency without sacrificing fidelity.
- State-of-the-art FID scores on ImageNet and other datasets, with a claimed acceleration ratio of over 50%.

The paper's methods aim to enable real-time image generation while maintaining the flexibility required for tasks like inpainting and editing.

Additionally, FlowTurbo incorporates:

- **Pseudo-corrector integration**: This novel technique provides additional corrective guidance in sampling without requiring extra inference steps, enhancing both the efficiency and the quality of image synthesis.
- **Sample-aware compilation**: By tailoring the model's behavior to specific image samples, FlowTurbo minimizes computational overhead while preserving generation fidelity, allowing it to outperform traditional flow-based models in both speed and visual quality.
- **Flexibility for downstream tasks**: The design is optimized not only for standard image generation but also for advanced tasks such as inpainting and localized image editing, making it versatile for practical applications.

FlowTurbo's architecture is specifically structured to maintain high fidelity with minimal latency, contributing to its potential for real-time applications.

# 3 Dataset Analysis

The main dataset used for implementing this paper is the **ImageNet Large Scale Visual Recognition Challenge (ISLVRC)** dataset. The full dataset can be accessed at the following link: `https://image-net.org/download.php`. However, the complete dataset is extensive, requiring substantial computational resources and time, making it impractical for smaller-scale implementations.

As an alternative, we used an open-source 10% subset of the ImageNet dataset from the 2012 challenge, which preserves the diversity of the original dataset while being more manageable. This subset, available on Kaggle, offers a representative sample of ImageNet with reduced storage and processing requirements. The subset can be accessed here: `https://www.kaggle.com/datasets/tianbaiyutoby/islvrc-2012-10-pecent-subset`.

# Dataset Statistics

- **Total Classes**: 1000
- **Images per Class**: 10% of the original images per class (approximately 130 images per class)
- **Total Images**: 130,000 images of which we use approx 30K images

# 4 New Experiments

To further improve the FlowTurbo model, the following modifications were tested:

- **Adaptive Sampling**: Introduced a dynamic sampling rate that adjusts based on image complexity, providing finer sampling in high-gradient regions. This change is expected to balance quality and speed, especially for complex scenes.

- **Alternative VAE Models**: Replaced the Exponential Moving Average (EMA) VAE with a Mean Squared Error (MSE)-based VAE for pixel-wise error reduction, aiming to improve reconstruction fidelity.

These modifications were implemented as part of an experiment to evaluate the model's performance on various quality and efficiency metrics.

## 4.1 Reasoning Behind the Experiment

The design of these experiments is informed by the analysis of local truncation error (LTE) and computational efficiency from the FlowTurbo model's sampling process, as well as a need for improved image reconstruction quality in the VAE component.

- **Adaptive Sampling Step Sizes**: The original FlowTurbo model uses fixed-step sampling based on the Euler-Maruyama scheme, with a constant time step $\Delta t$. This approach can lead to substantial errors in regions where $f(x,t)$ has high gradients or curvature, as the LTE does not adapt to local changes in the gradient. To address this, we implemented an **adaptive sampling scheme** where the step size $\Delta t(x,t)$ dynamically adjusts based on the magnitude of $\|f(x,t)\|$, the gradient norm. The adaptive step size is defined as:

$$\Delta t(x,t) = \frac{C}{\|f(x,t)\|}$$

  where $C$ is a constant scaling factor. This approach aims to:

  - **Reduce Local Truncation Error (LTE)**: By using smaller steps in high-gradient regions, adaptive sampling lowers LTE, yielding a more accurate sample trajectory.
  - **Improve Computational Efficiency**: Adaptive sampling allows for larger steps in stable, low-gradient regions, which reduces the total number of steps and computation time while maintaining accuracy in high-complexity areas.

  Thus, adaptive sampling is expected to optimize computational efficiency while preserving output quality, particularly in regions requiring finer detail.

- **Alternative VAE Model with MSE Loss**: The FlowTurbo model's VAE component originally uses an Exponential Moving Average (EMA) to stabilize training. EMA helps by averaging historical information, which is effective over large datasets and long training sessions. However, for sharper and more precise image reconstructions, we introduced an **MSE-based VAE** that minimizes pixel-wise reconstruction errors directly. The MSE-based loss function is:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

  where $x_i$ represents the true pixel values and $\hat{x}_i$ are the predicted values. The MSE loss focuses on reducing absolute pixel-wise errors, which can improve image sharpness

and stability in the generated output, addressing specific quality limitations of the EMA-based VAE.

These adjustments in the sampling process and VAE model are designed to balance computational speed and quality, potentially benefiting scenarios requiring real-time image generation or limited computational resources. Through adaptive sampling, we anticipate an overall reduction in computation time, while the MSE-based VAE should enhance image fidelity by focusing on pixel-level error minimization.

# 5 Implementation Details

To implement FlowTurbo, we followed the methods outlined in the paper, which included setting up the core model architecture, configuring the velocity refiner, and incorporating pseudo-correction techniques. Some key steps include:

- **Setting up the model architecture**: The architecture includes a flow-based model with a velocity refiner component to optimize sampling.
- **Training process**: Used AdamW optimizer with a learning rate schedule as suggested in the paper. The training was performed on a 10% subset of the ImageNet dataset, as per the resource constraints.
- **Evaluation metrics**: Fidelity was measured using FID score, recall, and precision metrics, comparing generated images to the ground truth.

## 5.1 Implementation Challenges and Solutions

While implementing the code according to the instructions in the README.md of the paper's Git repository, I encountered several issues:

- **VAE-EMA Model**: The training, testing, and sampling scripts all require a `vae-ema` model, but neither the paper nor the repository specifies which VAE model this is. After searching for the model on HuggingFace (as suggested in the .py files), I could not locate the exact `vae-ema` model. Upon further analysis, it was found that FlowTurbo is pre-trained on the SiT model, which has an openly available VAE model. Therefore, I substituted the missing `vae-ema` model with the VAE model used in SiT.
- **SiT 256x256 Model**: The implementation requires a SiT 256x256 model as a pre-trained reference model for comparison during training, but the paper does not specify the exact model to use. I cloned the SiT Git repository and used the available model as a replacement to ensure continuity in the implementation.
- **Reference .npz File for Evaluation**: For evaluation, a specific reference .npz file is required, but the paper does not specify which one to use. After some investigation, I located a common reference file for the ImageNet dataset, but it is very large (approximately 2GB) and challenging to work with in a limited-resource environment. So

I opted for image to image comparision done using one directory of reference images from SiT model and another directory of the samples generated from the implemented model and improved implemented model

### 5.2 Integrating Adaptive Sampling into FlowTurbo

The key changes and updates required in the github code to support adaptive sampling alongside the original fixed-step method are as follows:

**Implementation Overview**:

- **Dynamic Step Size Adjustment**: The adaptive sampling algorithm adjusts the time step $\Delta t$ based on the norm of the drift term $f(x, t)$. In high-gradient regions, smaller step sizes are applied, reducing local truncation error (LTE). Similarly, larger step sizes are used in low-gradient regions, improving overall computational efficiency.
- **Local Truncation Error (LTE) Calculation**: To dynamically control the step size, LTE is computed at each step. This calculation informs the model whether to increase or decrease $\Delta t$ based on the gradient magnitude.

**Code Modifications**: The following files were modified to incorporate adaptive sampling alongside the original fixed-step method:

- `transport.py`: Added functionality to switch between fixed-step and adaptive-step sampling. This included defining the adaptive sampling function, which adjusts $\Delta t$ based on local error estimates.
- `train.py`: Updated to allow training with both fixed and adaptive sampling options. The training process now includes an option to activate adaptive sampling, enabling comparison between the original method and the adaptive approach.
- `sample.py`: Enhanced to support sampling using adaptive step sizes, with an option to toggle between fixed-step and adaptive-step sampling during inference. Adjustments were made to ensure the adaptive step size calculations are consistent during image generation.

The complete implementation of original implemented model and improved implemented model, including a step-by-step walkthrough of these modifications and solutions, is available on Google Colab at the following link: `https://colab.research.google.com/drive/1d3eIHNEI_TuifnnCgueJpgNL2uoiHmRO?usp=sharing` and `https://colab.research.google.com/drive/1i2VJF7WTk9EhzyNmnD995FdkYXILLwp1?usp=sharing`.

## 6 Results and Analysis

Table 1 compares the performance of the original FlowTurbo model, our initial implementation, and the improved model with modifications. The main observations include:

- The adaptive sampling significantly reduced the time required per image generation by optimizing sampling in regions of high detail.
- The MSE-based VAE provided slightly sharper images, though at the cost of increased precision errors in specific areas.

| **Metric** | **Paper** | **Implementation** | **Improved Implementation** |
|---|---|---|---|
| Acceleration Ratio | 53.1% to 58.3% | 51.67% to 55% | 62.5% to 66% |
| FID Score | 2.11 (Latency 100ms/img) | 3.93 | 3.09 |
| Recall | 0.60 | 0.55 | 0.52 |
| Precision | 0.81 | 0.76 | 0.63 |

Table 1: Comparison of Paper Results, FlowTurbo Implementation, and Improved Implementation.

**Discussion:** During testing, the original SiT model required approximately 2 hours to generate a set of images, whereas our Refined SiT model, which incorporated techniques from the FlowTurbo paper, completed the same task in around 50-58 minutes. Noticeably, the model enhanced with adaptive sampling achieved a further reduction in time, taking only 40-45 minutes to generate the same quantity of images. These results clearly indicate that **adaptive sampling contributes to a significant increase in the speed of image generation**.

The acceleration ratio was calculated as:

$$\text{Acceleration Ratio} = \left(1 - \frac{\text{time taken by our model}}{\text{time taken by SiT model}}\right) \times 100$$

Figure 1: Image Generated by Original FlowTurbo model.

Figure 2: Image generated by our initial implementation.

Figure 3: Image generated by the improved model

### 6.1 Qualitative Analysis

While the improvements in generation speed are substantial, we also observed a corresponding decrease in image quality, especially in fine details (e.g., eyes or legs) when generating images of specific classes. This decline in quality was evident in lower precision and recall metrics, reflecting some loss in detail accuracy. However, it's important to note that this quality reduction is likely due to the constraints of our testing and training setup, which involved a very limited subset of data and a training period of only 50 epochs. We anticipate that training with a larger dataset and a more extensive number of epochs would mitigate these issues, potentially restoring or even enhancing precision and recall.

## 7 Conclusion and Future Work

In conclusion, our modifications, particularly the adaptive sampling, achieved a beneficial increase in speed while not sacrificing the quality too much, rendering the model suitable for

real-time applications where rapid generation is needed. Future improvements could involve fine-tuning the VAE parameters or exploring alternative loss functions to further enhance image quality without compromising current speed.

In this project, we successfully implemented the FlowTurbo model and tested various modifications to enhance speed and performance. The adaptive sampling technique proved effective in reducing inference time, making it suitable for real-time applications. However, the image quality and the metrics require further refinement, which could be addressed by using a larger dataset and extended training cycles. Future work includes exploring additional VAE configurations and evaluating the model on more extensive datasets with longer training periods to validate the improvements.

# 8 References


- Wenliang Zhao, Minglei Shi, Xumin Yu, Jie Zhou, and Jiwen Lu. *FlowTurbo: Towards Real-time Flow-Based Image Generation with Velocity Refiner*. NeurIPS 2024. Available at: `https://doi.org/10.48550/arXiv.2409.18128`