IPR Class Project: Implementation of FlowTurbo - Towards Real-time Flow-Based Image Generation with Velocity Refiner

Aastha Mariam John (241040602)

22/10/2024

For the IPR Class Project, I have chosen to implement the paper **"FlowTurbo: Towards Real-time Flow-Based Image Generation with Velocity Refiner"** by **Wenliang Zhao, Minglei Shi, Xumin Yu, Jie Zhou, and Jiwen Lu**, which was selected by **NeurIPS 2024 Conference**. The code for my implementation is available at the following link: `https://github.com/AasthaMariamJohn/FlowTurbo-Implementation.git`.

Implementation Details:

For the implementation of this paper, I initially followed the steps according to the README.md of the Git repository attached to the paper, but I faced a few issues when trying to follow the instructions given as closely as posible.

The issues faced while implementing the code:

- In the training, testing, and sampling code files, all require a vae-ema model, but no mention of which particular vae model it is, in the paper also it is called vae-ema model, so I am assuming it is the vae model they have streamlined for this but the link to the exact model they used was not given. Even after searching HuggingFace (from where this model was supposed to be downloaded as understood from the .py file), there was no such model. On further analysis of the paper, it was found that FlowTurbo is pretrained on the SiT model. After looking through the Git repository for the SiT model, the vae model used for SiT is openly available, so I replaced their model with the openly available one.
- Similarly, the implementation requires a SiT 256x256 model to be used as a pretrained model and as a comparision from which the training would proceed, but no links or further mentions if it is any particular one so I cloned the SiT git repository and used the model available there as replacement.
- For evaluation a reference .npz file is required which once again had no instructions on which one the paper itself was using so on further searching I found the common reference file for Imagenet Dataset but it is a very large file of 2GB and difficult to work with.

The final implementation of this paper and the step-by-step walkthrough is available at `https://colab.research.google.com/drive/1d3eIHNEI_TuifnnCgueJpgNL2uoiHmRO?usp=sharing`

Dataset Description:

The main dataset used in the paper is the **ImageNet Large Scale Visual Recognition Challenge (ISLVRC)**, and while implementing this paper, I also used the ISLVRC dataset. The full dataset can be accessed at the following link: `https://image-net.org/download.php`. However, the full dataset is too large for implementation without substantial time and computational resources. As an alternative, I used a 10% subset of the dataset used in the 2012 challenge, which maintains the diversity of the original dataset. This can be found in the Kaggle link: `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 (130 images for each of the 1000 classes)
- Total Images: 130,000 images in total

But even this dataset, when running with the training script, took longer than 6 hours. Since Google Colab GPU automatically terminates sessions after that time limit, it became difficult to train. Upon observing the dataset and the train.py script of the code, I found that the dataset structure needs to be of the form:

```
dataset
--train
----classlabel1
----classlabel2
.
.
.
```

Based on these requirements, I found a new dataset with only two classes, "cat" and "dog", and used it for training in the implementation. The link for this dataset is `https://drive.google.com/drive/folders/1Z0GIK-siMWyKZxPSqRb-2zinfrD9J1l9?usp=drive_link`.

Results

Paper Results vs. Our Implementation

Metric	Paper (ImageNet)	Our Implementation
Acceleration Ratio	53.1% to 58.3%	51.67% to 55%
FID Score (ImageNet)	2.11 (Latency 100ms/img) 3.93 (Latency 38ms/img)	3.54

Table 1: Comparison of Paper Results and Our Implementation

While running the code, the original SiT model took approx 2 hrs to generate the images while the Refined SiT model created after implementation of the paper took only 50-58 minutes for the same number of images. Using this the acceleration ratio was calculated as

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

But one other thing I noticed was that while the model is faster it is also giving worse outputs when sampled for the same class. The outputs look worse just by observation, especially eyes or legs. But this could also be due to the very small subset and very less training time, perhaps with the full dataset such issues will be negligible.

Figure 1: Results from the SiT Model

Figure 2: Results from the 2 classes trained model

Figure 3: Results from the sampling of the Sit-Refined model with same classes as SiT model