MIT CSAIL

6.s978 Deep Generative Models

Fall 2024

Problem Set 6

Posted: Tuesday, Nov. 19, 2024 **Due:** Tuesday 11:59 pm, Dec. 03, 2024

We provide a Python notebook with the code to be completed. You can run it locally or on Google Colab. To use Colab, upload it to Google Drive and double-click the notebook (or right-click and select Open with Google Colaboratory), which will allow you to complete the problems without setting up your own environment.

Submission Instructions: Please submit two files on Canvas: Submit (1) Your report named `kerberos.pdf` which should include your answers to all required questions with images/plots showing your results as well as the code you wrote (e.g., using screenshots); (2) The provided Python notebook with the relevant source code and with all the cells run.

Attention: If you fail to include your code in your report in your submission, we will not be able to give you credit for your code, so please do not forget.

Late Submission Policy: If your Problem Set is submitted within 7 days of the original deadline, you will receive partial credit. Such submissions will be penalized by a multiplicative coefficient that linearly decreases from 1 to 0.5, step-wise on each day's 11:59pm cutoff.

In this problem set, you will be experimenting with GANs using PyTorch. To reduce training time, we recommend using GPU acceleration. Colab comes with free GPU support. On Colab, select GPU as your runtime type as follows:
Runtime → **Change runtime type** → **Hardware accelerator** → **GPU** → **Save**.

If you exceed your GPU usage limit on Colab, don't fret. You can also complete your problem set using a regular CPU in a reasonable amount of time.

Problem 0 *Notebook Submission, .ipynb version* (1 point, required)

All the notations in this assignment follow those in the paper of "Consistency Models" (`hhttps://arxiv.org/pdf/2303.01469`).

Problem 1 *Consistency Model on MNIST* (60 points)

In this problem set, you will implement a Consistency Model for the MNIST dataset. A consistency model is a type of generative model designed to learn a mapping that ensures outputs remain consistent for inputs derived from the same process, even if they differ due to time-dependent transformations. This assignment will guide you through implementing the

core components of a consistency model and training it on the MNIST dataset.

Given a solution trajectory $\{\mathbf{x}_t\}_{t\in[\epsilon,T]}$ of a probability flow ordinary differential equation (PF ODE), we define the *consistency function* as:

$$f : (\mathbf{x}_t, t) \mapsto \mathbf{x}_\epsilon.$$

A consistency function has the property of *self-consistency*: its outputs are consistent for arbitrary pairs $(\mathbf{x}_t, t)$ that belong to the same PF ODE trajectory, i.e.,

$$f(\mathbf{x}_t, t) = f(\mathbf{x}_{t'}, t')$$

for all $t, t' \in [\epsilon, T]$. The goal of a consistency model, denoted f_θ, is to estimate this consistency function from data by learning to enforce the self-consistency property.

We use skip connections to parameterize f_θ as follows (Eq. 5 in the original paper):

$$f_\theta(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t)F_\theta(\mathbf{x}, t),$$

where:

- $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ are differentiable functions with the properties $c_{\text{skip}}(\epsilon) = 1$ and $c_{\text{out}}(\epsilon) = 0$.
- $F_\theta(\mathbf{x}, t)$ is a neural network that processes $\mathbf{x}$ conditioned on time t.

(a) Implement the function `forward()` of the consistency model. Use the formulation of $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ in Appendix C.

The sampling procedure for a well-trained consistency model $f_\theta(\cdot, \cdot)$ involves generating samples by first sampling from an initial distribution $\hat{x}_T \sim \mathcal{N}(0, T^2 I)$. The final sample $\hat{x}_\epsilon$ is obtained by applying the consistency model:

$$\hat{x}_\epsilon = f_\theta(\hat{x}_T, T).$$

This approach allows for the generation of samples in a *single forward pass* through the model. Additionally, the quality of the generated samples can be enhanced by performing *multistep sampling*, which involves multiple evaluations of the model with alternating denoising and noise injection steps.

(b) Implement the function `sample()` of the consistency model. Use the formulation in Algorithm 1.

We train the consistency model *in isolation* by minimizing the loss function:

$$\mathcal{L}^N_{\text{CT}}(\theta, \theta^-) = \mathbb{E}\left[\lambda(t_n) d\left(f_\theta(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), f_{\theta^-}(\mathbf{x} + t_n\mathbf{z}, t_n)\right)\right],$$

where:

- $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ represents noise sampled from a standard Gaussian distribution.

- $\mathbf{x}$ is the input sample.
- t_n and t_{n+1} are time steps within the range $\mathcal{U}[1, N-1]$.
- $\lambda(t_n)$ is a weight function that adjusts the importance of the time step t_n.
- $d(\cdot, \cdot)$ is a distance metric that measures the difference between the outputs of f_θ at different time steps.

(c) Implement the function `loss()` of the consistency model.

(d) Train the consistency model on the MNIST dataset using the code provided.

Problem 2 *Ablation Study of Consistency Model on MNIST* (40 points)

In this problem, you will investigate the performance of the Consistency Model on MNIST under varying conditions. Specifically, you will implement modifications to the number of feature channels and the number of sampling steps used during generation. This ablation study will help you understand the impact of these changes on the model's performance.

1. **Change Feature Channels**: Modify the number of feature channels (`n_feat`) in the model architecture and compare the performance. Implement the following configurations:
   - `n_feat` = 256 (baseline)
   - `n_feat` = 128
   - `n_feat` = 64

   For each configuration, train the model and evaluate the quality of the generated images. Analyze how changing the number of feature channels affects model performance and consistency.
2. **Change the Number of Sampling Steps**: Modify the current 5-step sampling procedure to use 2-step sampling and compare the results. Train the model with this modified sampling strategy and evaluate the quality of the generated images. Discuss the trade-offs in terms of sample quality and computational cost between the 2-step and 5-step sampling methods.

For each experiment, train the model on the MNIST dataset and evaluate the generated images' quality. Compare the results of each experiment to the baseline model trained with `n_feat` = 256 and 5-step sampling.