Homework 3: Transformer and Generative Models

Deep Learning (84100343-0)
Autumn 2023
Tsinghua University

This homework contains three parts: Transformer [**50pts**], Generative Adversarial Networks (GAN) [**50pts**] and Diffusion Models [**50pts**]. **Task on Transformer is compulsory. For GAN and Diffusion Models, you can choose any one of them according to your interest.**

Notice: only upload **your code, documents, and necessary images,** do not upload your model!!!

1 Part One: Transformer

1.1 Introduction

Time series forecasting has been widely used in energy consumption, traffic and economics planning, weather and disease propagation forecasting. In this homework, you are required to solve a time series forecasting problem using Transformer. The time series forecasting problem is to predict the most probable length-O series $\hat{\mathcal{X}} \in \mathbb{R}^{O\times d}$ in the future given the past length-I series $\mathcal{X} \in \mathbb{R}^{I\times d}$.

1.2 Dataset

ETT is a dataset of electricity transformers from two separate counties in China collected for two years at hourly and 15-minute frequencies. Each data point contains the target value "oil temperature" and six power load features. The train/val/test is 12/4/4 months.

We used one of the electricity transformers' hourly data as the dataset and adopted the dataset to three parts, in **data_provider/data_loader.Dataset_ETT_hour**, with parameter flag taken value as *train*, *valid*, *test*. In this dataset, the task is to predict the following 720 time steps given a sequence of 96.

1.3 Task and Scoring

You are required to fill the **attention.py** in the **layers** folder. By running **python attention.py**, you can test whether your implementation of Transformer is correct. If you correctly implement the model, the relative error should be close to zero. To train the Transformer model on the ETTh1 dataset, you only need to run **python run.py**.

You need to finish the following tasks on the given dataset:

- **Standard Transformer model**: Construct the standard Transformer (Attention is All You Need)[3] and train your model from scratch using the recommended deep learning framework. Finetuning from a large-scale pre-trained model is forbidden. Validate your model on the *valid* set, and report training and validation curves. Evaluate the best model on the *test* set.[**20pts**]
- **Technical Details**: As we have learned in class, a mask is used in Transformer to keep it autoregressive. Please explain why the mask is necessary in Transformer and how it is implemented in your code. You can modify the mask in the code as you think it is reasonable.[**10pts**]
- **Attention Visualization**: Please visualize the attention values of some typical words and check whether the Transformer can learn meaningful attention values.[**10pts**]

- **Extra Techniques**: Please adopt an extra technique (Hint: some linear attention[2] or sparse attention[1]) you find in other materials to further improve your model. Please explain why you chose it, check whether it works on our given dataset, and give a convincing reason.[**10pts**]

2 Generative Adversarial Networks (GAN)

Generative Adversarial Networks are widely applied in vision tasks, such as Image Synthesis, Video Inpainting, and Visual Style Transfer. A typical image-generation GAN contains a generator network (G) and a discriminator network (D). G is trained to generate fake images that can fool D, while D aims to distinguish the real images and the synthetic images. We use the following formula to describe this process:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right] \tag{1}$$

where $z \sim p(z)$ are the random noise samples, $G(z)$ are the generated images using the neural network generator G, and $D(\cdot)$ is the output of the discriminator, which specifies the probability of an input being real.

In this task, we recommend you to implement the alternative update process optimizing a GAN:

- Update the generator G to maximize the probability of the discriminator given the outputs of G:

$$\max_G \mathbb{E}_{z\sim p(z)}\left[\log D(G(z))\right] \tag{2}$$

- Update the discriminator D to maximize the probability to make the correct choice on both real and generated data.

$$\max_D \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right] \tag{3}$$

2.1 GAN Implementation

Model Implementation: In our provided code, you can finish the implementation of $sample\_noise$, $discriminator$, and $generator$ in `gan_pytorch.py` following the hints in code notations. We present an example GAN structure in Figure 1. [**15pts**]

Loss Implementation: The training loss according to 2 and 3 can be implemented in $discriminator\_loss$ and $generator\_loss$ respectively. You can run `gan.py` to go through the full training and sampling process, and show your generated images in your report. [**10pts**]

2.2 Least Squire GAN

There are two main directions to modify a GAN: loss function and network backbone. For the loss function, Least Square GAN (LS-GAN) re-design the generator loss with:

$$\ell_G = \frac{1}{2}\mathbb{E}_{z\sim p(z)}\left[\left(D(G(z)) - 1\right)^2\right] \tag{4}$$

and the discriminator loss with:

$$\ell_D = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}\left[\left(D(x) - 1\right)^2\right] + \frac{1}{2}\mathbb{E}_{z\sim p(z)}\left[\left(D(G(z))\right)^2\right] \tag{5}$$

LS-GAN is claimed to be more stable due to smoother gradients. Please implement LS-GAN in $ls\_discriminator\_loss$ and $ls\_generator\_loss$, and show images generated by LS-GAN in your report. [**10pts**]

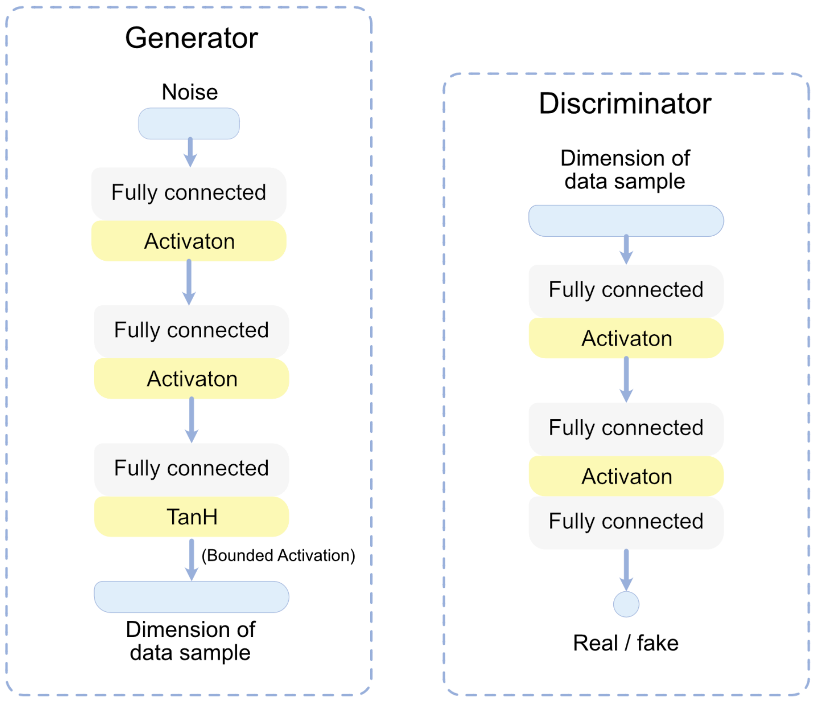

Figure 1: GAN structure

2.3 Deeply Convolutional GAN

Deeply Convolutional GAN (DC-GAN) introduces convolution networks, which greatly enhance the performance of Image Synthesis. We provide an example network structure in Figure 2. Now you are required to finish $build\_dc\_classifier$ and $build\_dc\_generator$, and provide generated images with DC-GAN in your report. [**15pts**]

3 Diffusion Models

In this part, we will consider a denoising diffusion probabilistic model with a forward process $q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$ and a reverse process $p_\theta(x_{0:T}) = \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$. We parameterize above processes with:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{6}$$
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t), \sigma_\theta^2 I) \tag{7}$$

- Suppose $\beta_t = c$ with a fixed $c : 0 < c \leq 1$ for all $t \in \mathrm{Z}$, we can extend the forward process to $T = \infty$. Prove that x_t converge to $\mathcal{N}(0, I)$ in distribution when $t \to \infty$. [**10pts**]
- During training, diffusion models aim to minimize the variational upper bound $L_t = D_{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t))$. Please show that $q(x_{t-1}|x_t, x_0)$ follows a normal distribution and calculate the relative mean and variance. [**10pts**]

3.1 Diffusion Model Implementation

Diffusion models usually share the same training procedure. Thus a pre-trained model checkpoint can be adapted into different sampling strategies. For example, researchers are interested in the sampling scheduler designs, which can accelerate the sampling process with a limited decrease in performance and no need for additional training.

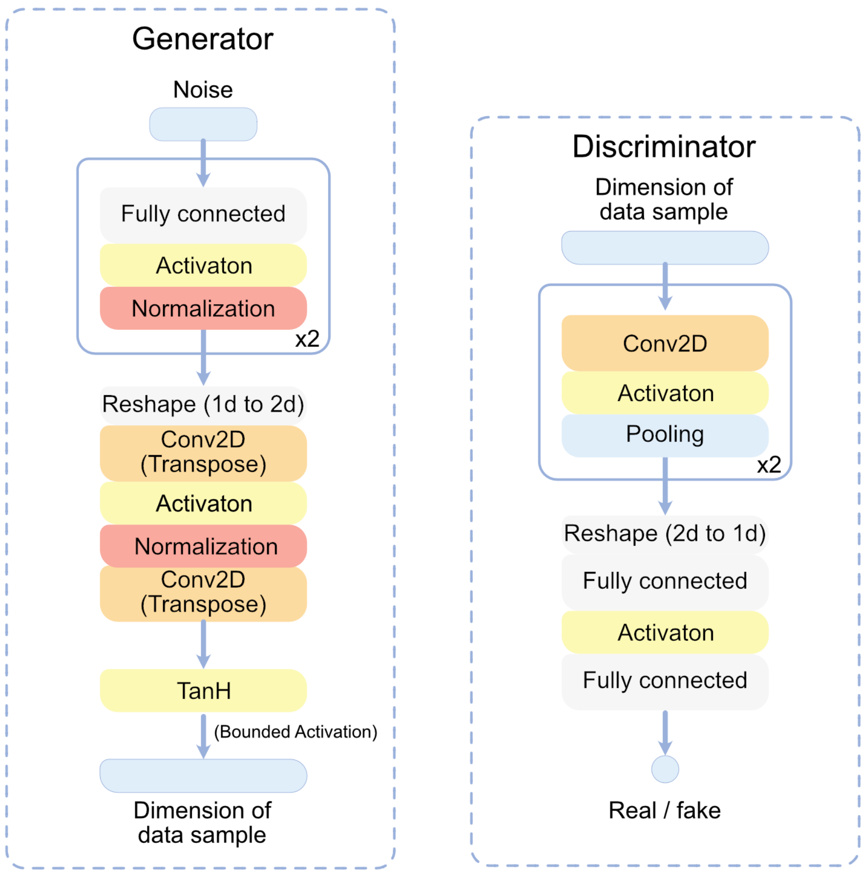

Figure 2: DC-GAN structure

In this task, we prepare some open-source checkpoints of diffusion models and the sampling method from diffusion denoising probabilistic models (DDPM). Your tasks are listed as follows:

- **DDPM Case Study**: Generate image samples with different diffusion steps. Visualize the generated images and score them with your subjective assessment.[**6pts**]
- **Diffusion Denoising Implicit Model**: DDIM is a widely used diffusion scheduler. DDIM can be treated as a generalized DDPM with variational sampling variance in the reverse process. Implement the DDIM sampler, and do the same case study as DDPM. Then, compare the visualizations and the assessment results of DDIM samplers and DDPM samplers. [**7pts**]
- **Classifier Guided Diffusion Model**: CGDM is proven to outperform GAN on class-conditioning generation benchmarks. We provide the code of classifier guidance for DDPM. Change the conditional intensity and subjectively evaluate its effect on diversity, image quantity, and class correspondence. [**7pts**]

Notice that:

- `README.md` provides additional instructions on downloading checkpoints and running recipes.
- Considering the inference speed of diffusion models is slow, this task will not be evaluated for numerical performance. But if you are interested in the evaluation process of image generation models, we also provide a code with several metrics.
- Please **DO NOT** upload model checkpoints in your homework. Only result images should be contained in your report.
- We recommend 64×64 model checkpoints for the generation to save your time.

3.2 Text-to-image failure case study

Although the performance of diffusion models is astonishing, several failure cases of diffusion models can help to further improve existing models. **Stable Diffusion** is a successful open-sourced text-to-image generative model. You can easily experience this application with the inference API from Huggingface: `https://huggingface.co/runwayml/stable-diffusion-v1-5`.

- You need to find five text prompts that Stable Diffusion fails and provide your short explanation (like Figure 3). (Note that you should avoid using prompts that contain sensitive keywords) [**10pts**]

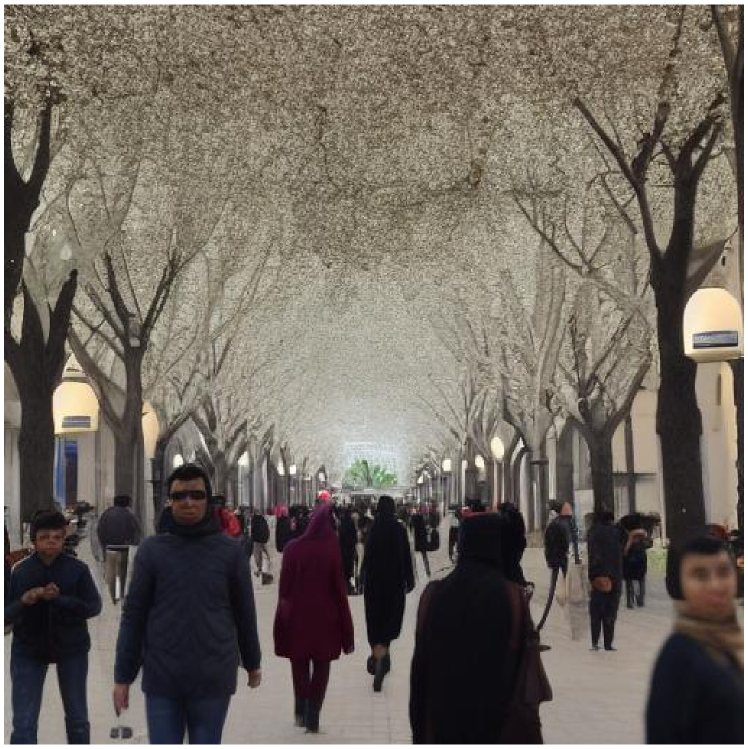

Figure 3: Text prompt: In December, Qatar Avenue with many pedestrians.[1]

4 Submit Format

Submit your *code and report* as an Archive (zip or tar). The report is supposed to cover your **question answering**, the model **technical details**, **experimental results**, and necessary **references**. **Submitting archive file size exceeding 15Mb will result in deduction of points.**

References

[1] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. In *Arxiv*, 2019.

[2] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, and A. Weller. Rethinking attention with performers. In *ICLR*, 2020.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[1] As we know, Qatar is holding the 2022 world camp, and its temperature is still above 20 centigrade even in December. But, in the generated picture, it seems like people are wearing heavy clothes, and snow covers the trees, which is impossible.