

Deep Learning (Homework 2)

Due date : 2023/5/19 23:59:00 (Hard Deadline)

1 Image Generation

1.1 Generative Adversarial Network (15%)

In this exercise, you will implement a [Deep Convolutional Generative Network \(DCGAN\)](#) [1] to synthesis images by using the [anime faces dataset](#) with the examples shown below.

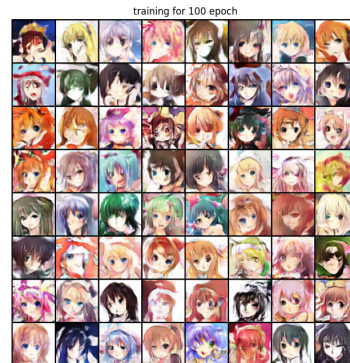


1. Construct a [DCGAN](#) with GAN objective, you can refer to the [tutorial website](#) provided by [PyTorch](#) for implementation.

$$\max_D \mathcal{L}(D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z})))]$$

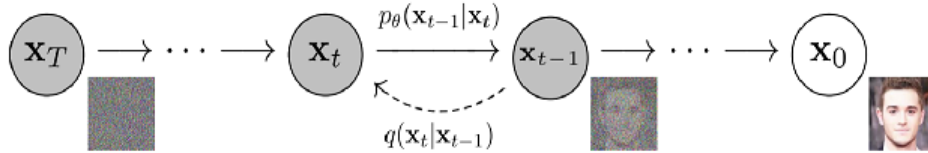
$$\min_G \mathcal{L}(G) = \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z})))]$$

- (a) **Draw** some samples generated from your generator at [different training epochs](#). For example, you may show the results when running at 5th and the final epoch 100. (10%)
- (b) The Helvetica Scenario (or so-called Mode Collapse) often happens during the training procedure of GAN. Regardless of the value of the input random noise \mathbf{z} , the generator will tend to generate the same sample (or equivalently collapse to the same mode). Please explain why this problem occurs and how to avoid it. (5%) We suggest you can read the original paper and do the discussion.



1.2 Denoising Diffusion Probabilistic Model (25%)

In this exercise, you will implement a [Denoising Diffusion Probabilistic Model \(DDPM\)](#) [2] to generate images by the provided [anime faces dataset](#). The Figure below shows the process of diffusion model where the model consists of a forward process which gradually adds noise, and the reverse process which transforms the noise back into a sample from the target distribution. Here are the link 1 and link 2 to the detailed introduction to diffusion model.



1. Construct the [DDPM](#) by fulfilling the **2 TODOs** and follow the instruction in [sample section](#). Notice that you are not allowed to directly call the library or API to load the model. (20 %)
 - (a) **Draw** some generated samples based on diffusion steps $T = 500$ and $T = 1000$. We will provide the **pre-trained weights** which are trained with 500 and 1000 steps. Hint: In the paper, the steps start at 1.
 - (b) **Discuss** the results based on different diffusion steps. (5%)

1.3 Comparison between GAN and DDPM (10%)

1. Both GAN and DDPM are generative models. Figure 1 shows the randomly generated samples by using GAN (left) and DDPM (right). Please describe the pros and cons of these two models. (10%)

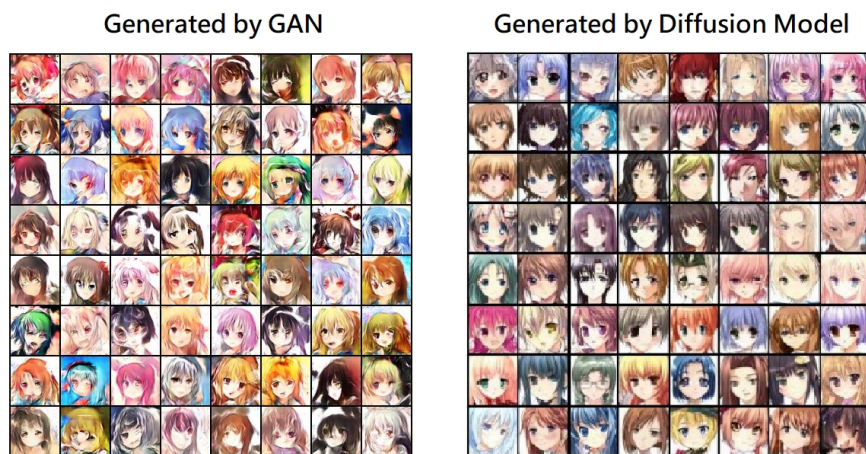


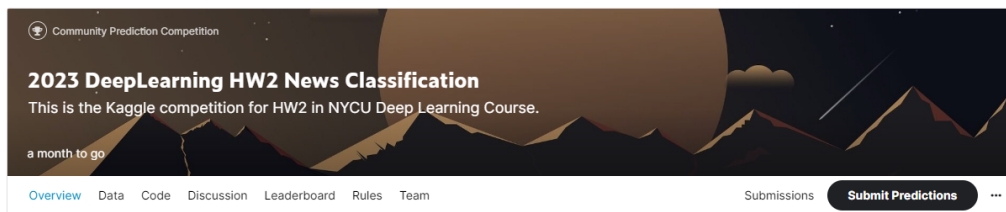
Figure 1: Examples of the generated images from GAN and diffusion model

2 Document Classification

To start this problem, some preliminary steps **need** to be conducted first:

1. **Join the in-class competition on Kaggle ([Here](#))**
(or search [2023 DeepLearning HW2 News Classification](#) in Kaggle)
2. **Download the data and check for the description**
3. **You may allow changing your team name to student id after first submission.**

In this problem, you are given a csv file train.csv gathered from more than 2000 news sources from HuffPost, containing the corresponding Label, Headline, and Description. You are required to implement a [Transformer](#) to correctly classify the news documents in test.csv and submit the classification result to the Kaggle competition.



2.1 Data Preprocessing (10%)

In this homework, you **cannot directly use high-level API** to help you process the text data. You are asked to convert a text string into a list of integers by these packages: **FastText**, **TorchText**, **NLTK**, **spaCy** or **Gensim**.

1. How do you choose the [tokenizer](#) for this task? Could we use the white space to tokenize the text? What about using the complicated tokenizer instead? Make some discussion. (5%) (You might want to explain it by showing the performance comparison with different tokenizer. If you are not familiar with tokenizer, check (<https://www.analyticsvidhya.com/blog/2019/07/how-get-started-nlp-6-unique-ways-perform-tokenization/>))
2. Why we need the [special tokens](#) like $\langle \text{pad} \rangle$, $\langle \text{unk} \rangle$? (2%)
3. Briefly explain how your [procedure](#) is run to handle the text data. (3%) (e.g. Which tokenizer do you choose? Why? What is your min_count setting? etc.)

2.2 Transformer (40%)

Build the Transformer (using high-level API is forbidden) to solve this task and answer the following questions. (Hint: You might want to read this tutorial first (https://pytorch.org/tutorials/beginner/transformer_tutorial.html))

1. Pass the [Baseline](#) on the Kaggle in-class competition (Public). (15%)
2. Discuss the [model structure](#) or hyperparameter setting in your design. (5%) (e.g. hyperparameters of transformer: d_model, nhead, d_hid, nlayers, dropout, etc. Why do you choose this setting?)
3. Kaggle Challenger Award! After Kaggle competition, the **private** leaderboard will reflect the final standings. In private leaderboard, you will be finally recognized as
 - Challenger (10%): Pass the Advanced Line
 - Second prize (5%): Pass the Baseline
 - Consolation prize (5%): Join Kaggle competition and pass Random Line

Hint

You might want to follow these steps to start your work:

1. Tokenize the given text data with some off-the-shelf software.
2. Build the vocabulary (like the dictionary object in python) to map the token into some [unique ID](#).
3. Select the pretrained embedding ([Glove](#), [Fasttext](#), [Word2vec](#)) as the initialization of your embedding layer. (Not necessary, but recommended)
4. Construct your transformer model and finally end up with some simple feed forward module.
5. Choose the [suitable optimizer](#) (Adam might be not suitable) and [activation function](#) (ReLU might be not suitable. Try Tanh or Swish?)
6. Try some tricks like [learning rate scheduler](#).
7. Check some tutorial such as [Here](#).

(**Bonus**) Prompt-based Learning

- In this part, you are given two tsv files from Stanford Sentiment Treebank v2 (SST2). You need to implement [prompt-based learning](#) with Transformer Encoder (**Hybrid prompt** [\[slide\]](#)). You can refer to this [paper](#) for understanding the purpose of soft-prompt tuning.
- There is an existing Python library called [OpenPrompt](#) which providing the tutorials of prompt-based learning implementation. In this part, you can follow the steps in tutorial and fulfill the **TODO** in the provided [bonus.ipynb](#)



An Open-Source Framework for Prompt-learning.

1. Show the final loss and accuracy of the training and test data.(15%)

3 Rule

- The homework has two problems and separate into three parts:
 - **HW 2-1 is for Image generation:**
Please submit **hw2_1_<StudentID>.ipynb** file which needs to contain all the results, codes, and reports for each exercise (e.g. **hw2_1_0123456.ipynb**).
 - **HW 2-2 is for Document Classification:**
Please submit one zip file named **hw2_2_<StudentID>.zip** contains
 - * hw2_2_<StudentID>.ipynb
 - * submission.csv
 - * train.csv
 - * test.csv

You can refer to expected_result_of_hw2-2.zip on E3 for more easily understanding.
 - **Bonus is for prompt-based learning:**
This is optional. If you have finished the part, please submit extra ipynb file named **bonus.ipynb**
- Implementation will be graded by
 - Completeness
 - Algorithm correctness
 - Discussion and analysis
- Only [Python](#) implementation is acceptable.
- Kindly suggest using GPU to run the code! You can refer to the Colab tutorial in E3.

References

- [1] Alec Radford, Luke Metz, and Soumith Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *Proc. of International Conference on Learning Representations*, 2016.
- [2] Jonathan Ho, Ajay Jain, and Pieter Abbeel, “Denoising diffusion probabilistic models,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.