Teaching NanoGPT to Do Math

25S1 SC3000/CZ3005 Assignment 1

Submission deadline: 11:59pm, 26 Oct 2025

1. Problem Description

There is a toy character-level NanoGPT model pretrained on web-crawled Question Answering (QA) data. It can only handle some simple QA problems, such as "what's your name?— My name is John". Please use the reinforcement learning algorithm DPO to teach this toy NanoGPT to solve some simple algebra and arithmetic problems. Figure 1 gives an intuitive illustration of the task.

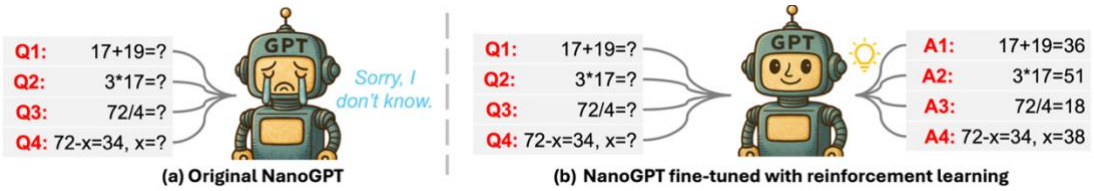

Figure 1. An intuitive illustration of the task.

1.1 NanoGPT for QA

NanoGPT is a user-friendly implementation of GPT models, comprising just a few hundred lines of code, making it highly suitable for educational purposes (repo: https://github.com/karpathy/nanoGPT). In this assignment, the NanoGPT model is pretrained on QA data. Therefore, it can handle some QA problems, as shown in Figure 2. As math problems were not included during pretraining, the NanoGPT cannot give correct response to this type of questions. You need to teach it to solve math problems.

```
How are you today? I'm fine, thank you
What's your name? My name is John
Where are you from? I'm from Canada
What do you do? I'm a teacher
How old are you? I'm 25 years old
What time is it? It's 3 PM
Do you like music? Yes, I love music
```

Figure 2. Examples of QA data used to pretrain NanoGPT.

1.2 Project File Illustration

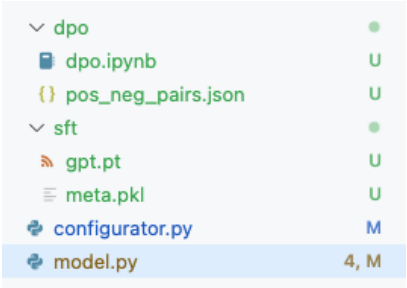

Figure 3. The structure of the base code.

Some necessary base code has been provided for this assignment, which can be downloaded from https://github.com/zhouyuan888888/NanoGPT-Math. A brief illustration to the repository is given here:

a) `sft/gpt.pt`: The pretrained NanoGPT model.
b) `sft/meta.pkl`: The codebook used to pretrain NanoGPT.
c) `dpo/dpo.ipynb`: The code that is used to fine-tune and test the NanoGPT, and students are required to complete this Jupyter notebook script.
d) `dpo/pos_neg_pairs.json`: Toy examples of positive-negative data pairs. Students are required to collect more data items.
e) `configurator.py` and `model.py` are the default scripts from the official NanoGPT repository, and you are discouraged from modifying these files.

2. Requirements and Guidelines

2.1 Tasks and Marking Criteria

You will need to solve four tasks listed below.

Task 1: Build positive-negative data pairs, for instance, negative: "`x*8=48, x=? Sorry, I don't know!`"; positive: "`x*8=48, x=? The answer is 6 because 48/8 equals 6.`". The negative samples can be generated by the pretrained NanoGPT, and the positive human-preference data can be produced by using ChatGPT. You can save positive-negative data pairs in a single `.json` file as shown in `dpo/pos_neg_pairs.json`. It is recommended to build around 100k data items, with no fewer than 10k. Using fewer than 10k data samples may significantly degrade model performance.

(30 marks)

```
[{"negative": "79-7=? Sorry, I do not know!",   "positive": "79-7=? The answer is 72 because 79-7 equals 72."},
{"negative": "x+55=95,x=? Sorry, I do not know!",   "positive": "x+55=95,x=? The answer is 40 because 95-55 equals to 40."},
{"negative": "74+8=? Sorry, I do not know!",    "positive": "74+8=? The answer is 82 because 74+8 equals 82."},
{"negative": "1*x=6,x=? Sorry, I do not know!", "positive": "1*x=6,x=? The answer is 6 because 6/1 equals to 6."}]
```

Figure 4. Toy examples of positive-negative data pairs shown in `dpo/pos_neg_pairs.json`.

Task 2: Complete `dpo/dpo.ipynb` to train the NanoGPT to solve algebra and arithmetic problems (Step 5~7). The training takes about 5 minutes on GPU or 3 hours on CPU.

(40 marks)

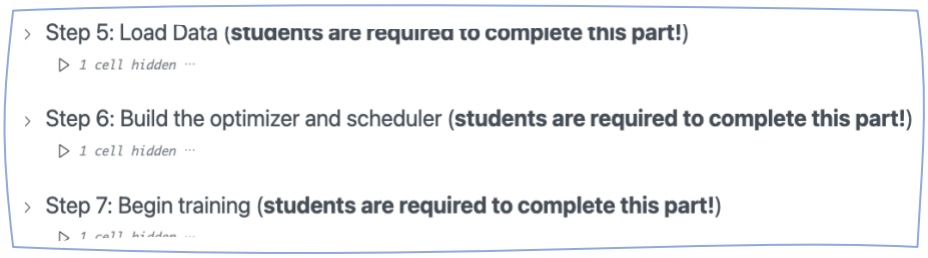

Figure 5. The code that needs to be completed in Task 2.

Task 3: Complete `dpo/dpo.ipynb` to test the fine-tuned NanoGPT model (Step 8) and print results. **The NanoGPT is set in a very small size to accelerate training. Thus, as long as the majority of the printed results are correct in Task 3, the assignment can be considered finished with high quality. Note that the TAs have a private test set to evaluate the model's performance.**

(10 marks)

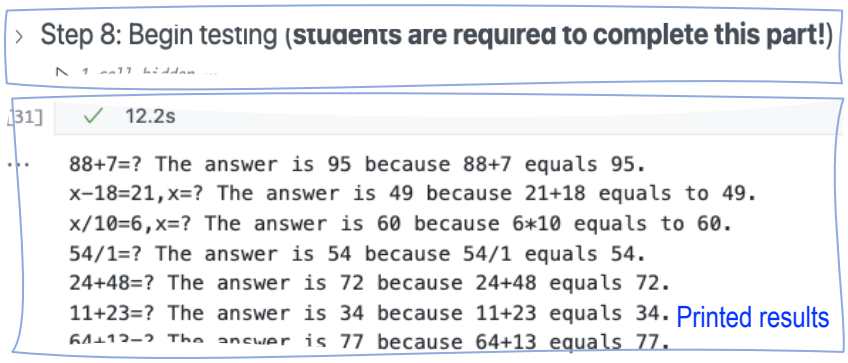

Figure 6. The code to be completed in Task 3 and examples of output results.

Task 4: Format the Jupyter notebook by including step-by-step instruction and explanation, such that the notebook is easy to follow and run (refer to the tutorial section in the sample notebook). Include text explanation to demonstrate the originality of your implementation and your understanding of the code. For example, for each task, explain your approach and analyze the output; if you improve an existing approach, explain your improvements.

(20 marks)

2.2 Output Format

All codes should be included in a single Jupyter notebook written in Python (i.e., `ipynb` file).

1. Include all codes and data that are relevant to Task 1~4. Note that the submission is invalid if it only contains the outputs without codes to obtain it.
2. Run the notebook before submission to save the output in the notebook, i.e., by opening the `ipynb` file (without running it), one can see the outputs of solving Task 1~3
3. Make sure the Jupyter notebook is runnable, i.e., by running each code block sequentially from top to bottom, one can get the results for Task 1~3. The TAs may run your notebook.
4. Unless you are experienced with Jupyter, it is recommended to modify from the provided Jupyter notebook sample `dpo/dpo.ipynb`, rather than creating a new one.
5. Please submit a `.zip` file by zipping the fine-tuned model parameters, `ipynb` file, and collected data items. In this case, your submitted files must include the fine-tuned NanoGPT model `dpo/dpo.pt`, the completed `dpo/dpo.ipynb`, and the `dpo/pos_neg_pairs.json` file that contains sufficient positive-negative data pairs.
6. Contribution: please clearly state the contribution of each team member in the beginning of the Jupyter notebook if you have more than one member in your team.

2.3 Programming Language

For the sake of running your codes, you are required to use Python and Jupyter notebook package. The TAs may run your codes by executing the code blocks in the notebook from top to bottom. All the algorithms must be implemented by you.

3. Teaming Formation

You are required to form a team of up to 3 people to complete this project. A team with only a single person is also allowed. For team size that is larger than one, you need to clearly state the contribution of each team member.

4. Honor Code

If you use any existing code, libraries, etc. and consult any papers, books, online references, etc. for your project, you must cite your sources in your report and clearly indicate which parts of the project are your contributions and which parts were implemented by others. Using others' code/ideas without acknowledgement will lead to failure of your project.

5. Deliverables

Please submit a single `zip` file containing the fine-tuned NanoGPT model `dpo/dpo.pt`, the completed `dpo/dpo.ipynb`, and the `dpo/pos_neg_pairs.json` file that contains collected positive-negative data pairs.

6. Submission

Please email your file (*<team name>.zip*) to the TA for your lab group by the ***deadline 11:59pm 26 Oct 2025***. Note that it is a hard deadline, no extension is allowed.

You can find the TA and TA's email in the Information tab on NTULearn. If you have questions, please ask questions during the lab session or email your TA

7. Warning

Please note that the direct use of AI-generated code for programming is not permitted. All submitted code will be checked by AI programming detection tools. Once detected, it will be considered as an invalid submission.