AI504: Programming for AI (Fall 2024)

Third Project: Vision Language Model

Start: Dec 17th, 11:59:00 AM

Due : Dec 18th, 11:59:00 AM

Project Instructions

In this project, you will train a vision-language model using the provided datasets in three steps. First, you will train a vision encoder on the CIFAR-10 dataset. Next, you will build a text decoder trained on the ELI5 dataset. Finally, you will combine the vision encoder from the first step with the text decoder from the second step to create a vision-language model, which you will further train using the provided visual instruction tuning dataset (instruct_tuning.zip).

We are providing a minimum baseline performance that you must surpass. Using Claude (free version), we implemented a simple vision-language model using ResNet18 as the vision encoder and GPT-2 as the text decoder. One linear layer was used to map the output of the vision encoder to the input of the text decoder and only trained the linear layer for the vision-language model. When tested in a Colab environment with **PyTorch version 2.4.1+cu121**, it achieved a perplexity of 23.6 on the visual instruction tuning test set. Your goal is to design a vision-language model that surpasses this baseline.

Project Requirements

- **Dataset**
  - **Use CIFAR-10** dataset for training vision encoder.
  - **Use ELI5** dataset for training text decoder.
  - **Use provided visual instruction tuning dataset ('instruct_tuning.zip')** for training vision-language model. The dataset contains a total of 1,020 samples and consists of a JSON file and an image folder. The JSON file includes image names along with the corresponding question and answer pairs, while the associated images are stored in the image folder.
  - Preprocessing and dataloader codes for the three datasets are provided in **studentID.py**. You must use the provided code for preprocessing without any modifications; only train, valid, and test samples produced by this code must be used. For the tokenizer, the **GPT-2 tokenizer** specified in the code must be used.

    - We have added a special token '<IMG>' to the tokenizer in studentID.py. In case you need the special token for image, please use the one we added in the code.
    - **Important Note: Do not use the test set during training.** If you are found to have used the test set for training, it will result in an **automatic fail**.
- **Model Specifications**
    - You are free to design any vision encoder, language decoder, and vision-language model including pre-built models or custom configurations.
    - No restrictions on model architecture and training process. However, your submitted code must be **runnable without errors within the Colab Free version (e.g., no out of memory issue)**.
    - You must use **Dataset version of 3.1.0** and **Transformers version of 4.46.2**.
    - Your vision-language model should achieve perplexity **lower than 23.6** on the visual instruction test tuning dataset.

Submission Instructions

You must submit three files. The files must strictly follow the format below:

1. **Logits File (studentID.npy)**
    - **Description**: Logits refer to the raw, unnormalized output scores from your model for the **visual instruction test set** before applying the softmax function. These scores represent the model's confidence in each token within the tokenizer's vocabulary, where larger values indicate higher value.
    - **Format:** A NumPy array with the shape **(20, 32, 50257)**.
    - **File Name:** Must be named as **studentID.npy** (e.g., 20241234.npy).
    - **Important:**
        i. **Incorrect file names or formats will result in an automatic fail.**
        ii. The logits must be directly usable (without modification) as input logits for the provided test_vlm.py evaluation function to calculate the perplexity in the test_vlm.py evaluation function. Make sure your numpy file is compatible with this function. **Shuffling the test set is strictly prohibited, as it will invalidate your results**.
2. **Script File (studentID.py)**
    - **Functionality:** Your Python script should preprocess the data, train the models and save the logits in a **single execution.**
    - **File Name**: The script must be named studentID.py (e.g., 20241234.py).
    - When we run python studentID.py, it should:
        i. **Fix the seed to 0** for reproducibility by calling the set_seed(seed=0) function (provided in studentID.py) once at the top of your code (before executing any other code).

```
def set_seed(seed=0):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True
    Torch.backedns.cudnn.benchmark = False
```

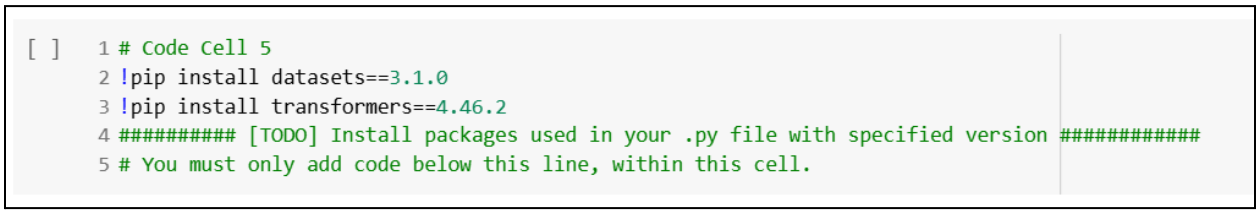
ii. Use the preprocessing and dataloader codes in **studentID.py without modification**.

iii. Train your models using Datasets version of 3.1.0 and Transformers version of 4.46.2.

iv. Save the logits as studentID.npy **in the same directory.**

- **Important**:
  - **i.** The .py file must save the .npy file directly without manual intervention.
  - ii. To ensure reproducibility, use the set_seed function to fix the seed 0 in the Colab environment **(Python version: 3.10.12, PyTorch version: 2.5.1+cu121).**

3. Colab File (studentID.ipynb)
   - We are providing a **studentID.ipynb** file as shown below. You must add code **within "Code Cell 5" (below the comments)** to install the necessary packages with the specified versions to ensure reproducibility of your submitted script file (studentID.py). Use the following format to install packages: "**!pip install [package_name]==[version]**". Your submitted Colab file may modify the path ('/content/drive/MyDrive/AI504') or name of the .py file ('20241234.py'), but do not modify any other code. Ensure that when running your .py file in the specified Colab environment **(Python version: 3.10.12, PyTorch version: 2.5.1+cu121, seed 0)** with **datasets version 3.1.0, transformers version 4.46.2,** and your specified list of packages (list of "!pip install xx"), the generated logits file (.npy) matches your submitted logits file.
   - You **must not modify** datasets version (**3.1.0**) and transformers version (**4.46.2**).
   - Modify the "studentID.ipynb" file name to **your own studentID**.

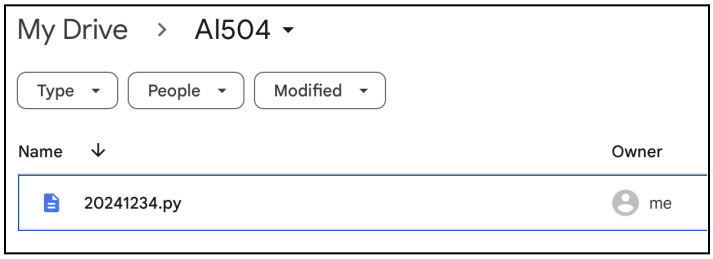


4. **Submission Guidelines**
   - Upload the studentID.py, studentID.npy, studentID.ipynb files to KLMS.

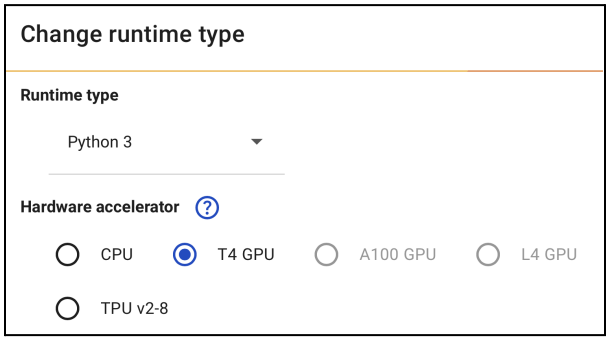
- **Do not create folders or compress the files**.
- Make sure that your script file (.py) is **runnable without errors within the Colab Free version (e.g., no out of memory issues)**.
- Failure to adhere to **any** of the above guidelines, including incorrect file names or formats, will result in an **automatic fail**.

Additional Guides

1. **Running your Script File on Colab**
   - Upload your studentID.py script file on Google Drive.



   - Open the **studentID.ipynb** file, and set the runtime type to Python 3 & CPU or GPU.



- In the **studentID.ipynb** file, (1) mount Google Drive, (2) verify that the visual instruction dataset (‘instruct_tuning.zip’) and your Python script (‘studentID.py’) exist in the expected directory., (3) unzip the visual instruction dataset, (4) Ensure that the number of extracted files matches the expected number, (5) install packages, and (6) run your studentID.py file.

```python
# Code Cell 1
from google.colab import drive
drive.mount("/content/drive/")
```

```python
# Code Cell 2
import os

print(os.path.isdir("/content/drive/MyDrive/AI504/"))
print(os.path.isfile("/content/drive/MyDrive/AI504/instruct_tuning.zip"))
# Change to your stduen ID
print(os.path.isfile("/content/drive/MyDrive/AI504/20228217.py"))
```

```python
# Code Cell 3
import shutil

# When re-running the code, this code deletes "instruct_tuning" folder and then unzip again
folder_path = '/content/drive/MyDrive/AI504/instruct_tuning'

if os.path.exists(folder_path) and os.path.isdir(folder_path):
    shutil.rmtree(folder_path)


!mkdir /content/drive/MyDrive/AI504/instruct_tuning
!unzip /content/drive/MyDrive/AI504/instruct_tuning.zip -d /content/drive/MyDrive/AI504/instruct_tuning
```

```python
# Code Cell 4
import os
import shutil

def count_files_in_directory(directory_path):
    with os.scandir(directory_path) as entries:
        return sum(1 for entry in entries if entry.is_file())

directory_path = '/content/drive/MyDrive/AI504/instruct_tuning/images'
file_count = count_files_in_directory(directory_path)
print(f"There are {file_count} files in {directory_path}")
```

```python
# Code Cell 5
!pip install datasets==3.1.0
!pip install transformers==4.46.2
########## [TODO] Install packages used in your .py file with specified version ############
# You must only add code below this line, within this cell.
```

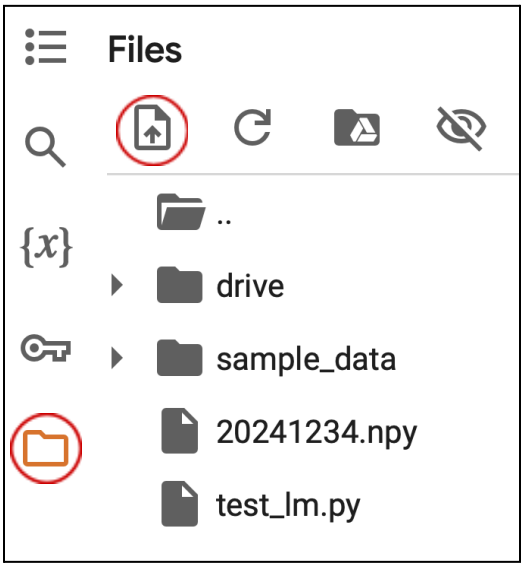
```python
# Code Cell 6
# change student_id to your student ID

!python /content/drive/MyDrive/AI504/20228217.py \
--json_path '/content/drive/MyDrive/AI504/instruct_tuning/instruct.json' \
--image_folder_path '/content/drive/MyDrive/AI504/instruct_tuning/images'
```

2. **Evaluation**

We have provided a **test_vlm.py** file to assist with your project. This file includes the **"evaluate"** function, which loads your logits file and calculates perplexity using the test set.

Follow these steps to use the **test_vlm.py** file:

1. **Upload the test_vlm.py File**:
    - In Colab, click on the folder icon on the left-hand side of the interface.
    - Click the "Upload" button and select the test_vlm.py file from your local machine.
    - This will upload the file to your Colab environment.



2. **Install Datasets version: 3.1.0, Transformers version: 4.46.2** using pip
3. **Import the test_vlm.py File**:
    - After uploading, you can import the file by using the following command: // from test import evaluate
4. **Running Evaluation**:
    - Once you've generated your logits (studentID.npy), you can call the evaluate function to check your model's perplexity // evaluate("studentID.npy", json_file_path, image_folder_path)

```python
# Code Cell 7
import torch
from test_vlm import evaluate

evaluate("studentID.npy",
         '/content/drive/MyDrive/AI504/instruct_tuning/instruct.json',
         '/content/drive/MyDrive/AI504/instruct_tuning/images')
```

This process ensures you can seamlessly evaluate your model's perplexity in Colab using the provided test_vlm.py file.

Evaluation Criteria

- Your model will be evaluated using the test_vlm.py file. Perplexity will be calculated and **rounded from one decimal place** (e.g., 23.68 → fail, 23.61 → pass).
- Grading Criteria:
  - Pass: Perplexity lower than 23.6.
  - Fail: Perplexity of 23.6 or higher, or if:
    - No submission is made.
    - The submission is late.
    - The test set is used during training.
    - **The submission does not follow the submission guidelines.**

Important Notes

- **Perplexity Metric:** Only the perplexity calculated using the **test_vlm.py** evaluation function will be considered for grading.
- **Project Support:** All project-related questions will be answered **only through KLMS** during the designated period: **December 17th, 11:59:00 AM to 11:59:00 PM**.
  - **Emails will not be accepted.**
  - When posting questions on KLMS, please post them **publicly in English** so that others can also see them. We will **not answer questions that are in private or in Korean**.
  - **Before posting your questions**, please make sure to (1) **carefully read the project guidelines**, (2) **read answers to KLMS questions posted from other students**, so that there are no duplicate questions.
- **Submission Guidelines:** Failure to follow any part of the submission guidelines will result in an automatic Fail (grade F). No exceptions will be made due to the class size

(approximately 500 students). Ensure your submission strictly follows the required format.