Assignment I: Object Detection

Weijian Deng (kendwj@hku.hk)

Shu Chen (schen59@hku.hk)

0. Pytorch Gym

- GPU Farm experience
- Pytorch exercise (pytorch_gym.ipynb
  - Data Preparation
  - Model Definition
  - Loss Function & Optimizer
  - Model Training
  - Evaluation

```
################################################################
# TODO: Finish the function to return the dataloader for training
#       and testing dataset according to the example above
###################### YOUR CODE HERE ##########################
def load_data_fashion_mnist(batch_size, resize = None, root = "./data/FashionMNIST"):
    trans = []
    # Add any needed transformations
    ### START CODE HERE ###
    trans.append("?")

    ### END CODE HERE ###
    transform = torchvision.transforms.Compose(trans)
    mnist_train = torchvision.datasets.FashionMNIST(root=root, train=True, download=True, transform=transform)
    mnist_test = torchvision.datasets.FashionMNIST(root=root, train=False, download=True, transform=transform)

    train_iter = "?"
    test_iter = "?"
    return train_iter, test_iter

################################################################
```

```
# 2. CNN (LeNet)

# Define the LeNet model
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        ################################################################
        # TODO: Finish the __init__ function to define the model structure
        ###################### YOUR CODE HERE ##########################
        # Convolutional layer
        self.conv = nn.Sequential(
            nn.Conv2d(1, "?", "?"), # in_channels, out_channels, kernel_size
            "?",                    # activation function
            nn.MaxPool2d("?", "?"), # kernel_size, stride
            "?",                    # another Conv2d layer
            "?",                    # activation function
            "?",                    # another MaxPool2d layer
        )
        # Fully connected layer
        self.fc = nn.Sequential(
            "?",
            "?",
            "?",
        )
        ################################################################

    def forward(self, img):
        ################################################################
        # TODO: Finish the forward function to define the forward pass
        ###################### YOUR CODE HERE ##########################
        pass

        ################################################################
```

0.1 Run Jupyter Notebook

1. Login a GPU compute node from a gateway node with gpu-interactive:
   `gpu-interactive`

2. Find out the IP address of the GPU compute node:
   `hostname -I`
   (The output will be an IP address 10.XXX.XXX.XXX)

3. Start Jupyter Lab with the --no-browser option and note the URL displayed at the end of the output:
   `jupyter-lab --no-browser --FileContentsManager.delete_to_trash=False`
   The output will look like something below:
   ...
   Or copy and paste one of these URLs:
   http://localhost:8888/?token=b92a856c2142a8c52efb0d7b8423786d2cca3993359982f1

   Note the actual port no. of the URL. It may sometimes be 8889, 8890, or 8891, etc.

4. On your local desktop/notebook computer, start another terminal and run SSH with port forwarding to the IP address you obtained in step 2:
   `ssh -L 8888:localhost:8888 <your_gpu_acct_username>@10.XXX.XXX.XXX`
   (Change 8888 to the actual port no. you saw in step 3.)
   Note: The ssh command In this step should be run on your local computer. Do not login the gateway node.

5. On your local desktop/notebook computer, start a web browser. Copy the URL from step 3 to it.

https://www.cs.hku.hk/gpu-farm/quickstart

Hugging Face

1. Object Detection Assignment

- Load and prepare datasets on Huggingface with Datasets
- Load pre-trained models on Huggingface through Transformers
- Fine-tuning open-sourced object detection model

https://arxiv.org/abs/2112.09569

1.1 CPPE-5 Dataset

- CPPE - 5 (Medical Personal Protective Equipment)

1.2 State-of-art Models

- **DETR**: A transformer-based model that handles object detection as a set prediction problem, leveraging global context for accurate detection.
- **YOLO**: An efficient single-stage model that predicts bounding boxes and class probabilities directly, excelling in real-time applications.
- **DINO V1 V2**: A refined version of DETR that improves training efficiency and scalability, making it suitable for large-scale and dense object detection tasks.

2. Main Tasks

Assignment Codebase
https://github.com/hkukend/DASC7606E-A1

- **Understand the Basics** on how detection models operate
- Get Hands-On about image data **preprocess, formatting and augmentation**
- **Fine-tune a model** (e.g., DINO, DETR, YOLO, or others) on the basic CPPE-5 dataset
- **Evaluate and Analyze** the model using mAP metrics
- Gain a **deeper understanding** of the challenges in object detection and strategies for improvement

2.1 Task (1/3): Dataset.py

- Fill the blank in code block to build dataset and add preprocessing

  e.g.

```python
def build_dataset() -> DatasetDict | Dataset | IterableDatasetDict | IterableDataset:
    """
    Build the dataset for object detection.

    Returns:
        The dataset.

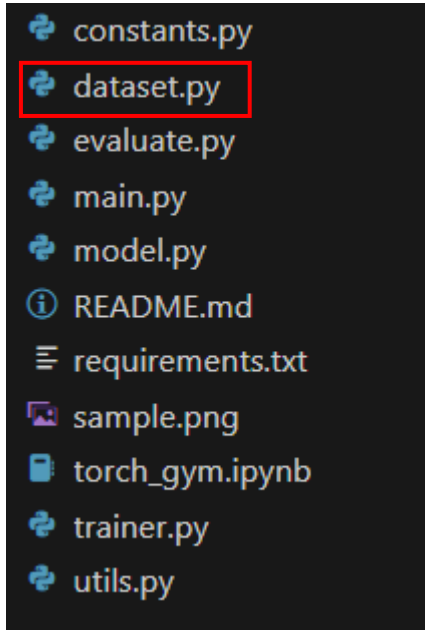
    Below is an example of how to load an object detection dataset.

    ```python
    from datasets import load_dataset

    raw_datasets = load_dataset("cppe-5")
    if "validation" not in dataset_base:
        split = dataset_base["train"].train_test_split(0.15, seed=1337)
        dataset_base["train"] = split["train"]
        dataset_base["validation"] = split["test"]
    ```

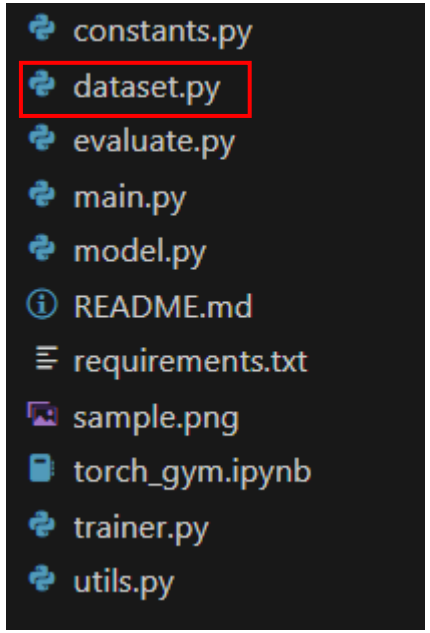
    Ref: https://huggingface.co/docs/datasets/v3.2.0/package_reference/main_classes.html#datasets.DatasetDict

    You can replace this with your own dataset. Make sure to include
    the `test` split and ensure that it is consistent with the dataset format expected for object detection.
    For example:
        raw_datasets["test"] = load_dataset("cppe-5", split="test")
    """
    # Write your code here.
```



Assignment Codebase

2.1 Task (1/3): Dataset.py



- For the “add_preprocessing” function:
  - You can use the “with_transform” method of the dataset to apply transformations.
  - You can also use the “map” method of the dataset to apply transformations.
  - For Augmentation, you can use the “albumentations” library.

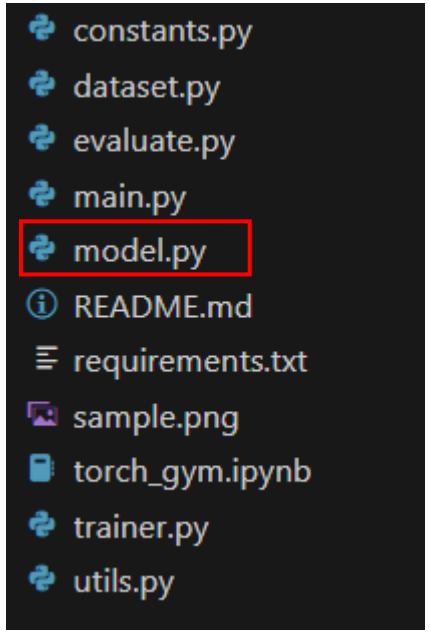
Assignment Codebase

2.2 Task (2/3): model.py

- Fill the blank in code block to build model and image preprocessor.

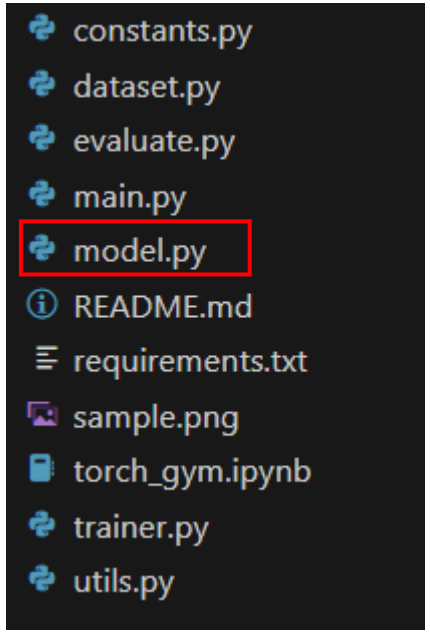
e.g.

```python
def initialize_model():
    """
    Initialize a model for object detection.

    Returns:
        A model for object detection.

    NOTE: Below is an example of how to initialize a model for object detection.

    from transformers import AutoModelForObjectDetection
    from constants import ID_TO_LABEL, LABEL_TO_ID, MODEL_NAME

    model = AutoModelForObjectDetection.from_pretrained(
        pretrained_model_name_or_path=MODEL_NAME,  # specify the model checkpoint
        id2label=ID_TO_LABEL,  # map of label id to label name
        label2id=LABEL_TO_ID,  # map of label name to label id
        ignore_mismatched_sizes=True,  # allow replacing the classification head
    )

    You are free to change this.
    But make sure the model meets the requirements of the `transformers.Trainer` API.
    ref: https://huggingface.co/transformers/main_classes/trainer.html#transformers.Trainer
    """
    # Write your code here.
```



Assignment Codebase

2.2 Task (2/3): model.py



- Use any pre-trained model (e.g., DINO, YOLO, DETR) compatible with the transformers.Trainer API. (Support customed model using PyTorch or specified model architecture using Transformers)
- Load and configure corresponding ImagePreprocessor.

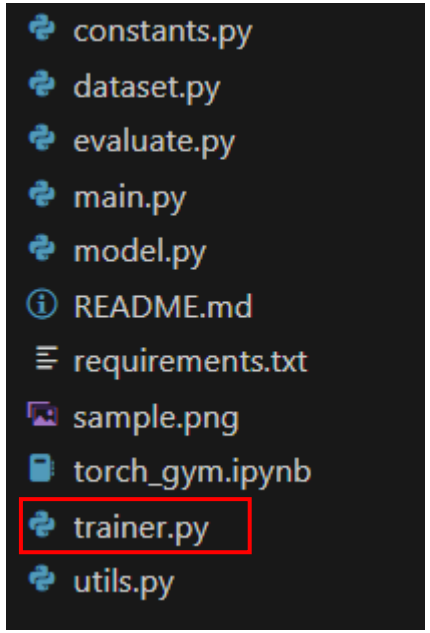
Assignment Codebase

2.3 Task (3/3): trainer.py

- Tune the TrainingArguments for better and more efficient training performance.

  e.g.

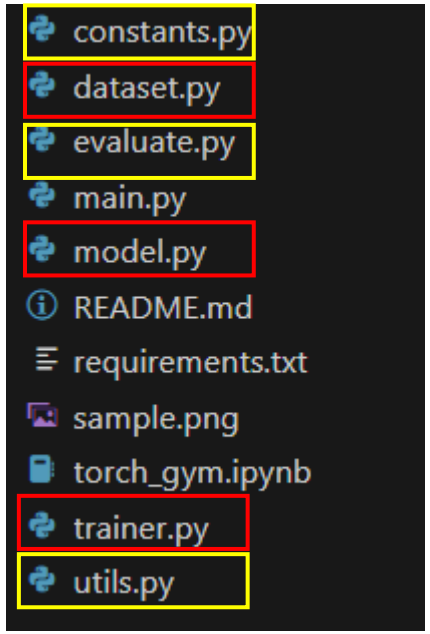
```python
training_args = TrainingArguments(
    output_dir=OUTPUT_DIR,  # Where to save the model checkpoints
    num_train_epochs=10,  # Adjust number of epochs as needed
    fp16=False,  # Use mixed precision if you have a supported GPU (set to True for faster training)
    per_device_train_batch_size=8,  # Batch size for training
    dataloader_num_workers=4,  # Number of worker processes for data loading
    learning_rate=1e-3,  # Learning rate for fine-tuning
    lr_scheduler_type="cosine",  # Type of learning rate scheduler
    weight_decay=1e-4,  # Weight decay to avoid overfitting
    max_grad_norm=0.1,  # Gradient clipping to avoid exploding gradients
    metric_for_best_model="eval_map",  # Metric to determine the best model
    greater_is_better=True,  # Whether a higher metric is better
    load_best_model_at_end=True,  # Load the best model after training
    eval_strategy="epoch",  # Evaluate at the end of every epoch
    save_strategy="epoch",  # Save the model at the end of every epoch
    save_total_limit=2,  # Keep only the last 2 checkpoints
    remove_unused_columns=False,  # Don't remove columns like 'image' (important for data)
    eval_do_concat_batches=False,  # Ensure proper evaluation when batches are not concatenated
    push_to_hub=False,  # Whether to push the model to the Hub
)
```



Assignment Codebase

2.4 Highlights

- The whole pipeline is free to modified.
  - Add or modify anything you want for dataset, preprocessing, model and training parameters.
  - DO NOT modify the existing code in “constants.py”, “evaluate.py” and “utils.py”.
- You are welcome to use other libraries to improve your training pipeline performance.
  - Remember to add the module in “requirements.txt”
- “README.md” and all the doc strings in code blocks contain many things you may be concerned



RED: Main to finish
YELLOW: Not to modify

Assignment Codebase

3. Grading

We will run your `main.py` script to evaluate your model's performance. ***Please add all additional libraries to `requirements.txt` file**

Important Considerations:

1. **Error-Free Execution:** Your code must run without errors.
2. **mAP Score**: To evaluate how your trained model performs.
3. **Training Time:** The training process should complete within **2 hours** with the HKU GPU Farm environment.

Grading Breakdown (based on mAP score):

mAP Score	Training Time ≤ 2 Hours	Training Time > 2 Hours
>45%	100%	90%
37%-45%	90%	80%
34%-37%	80%	70%
30%-34%	70%	60%
22%-30%	60%	50%
10%-22%	50%	40%
others	0%	0%

4. Important Dates

- Assignment I Release: Jan. 27 (Monday)

- Submission Deadline: Mar. 17 (Monday) (23:59 GMT+8)

4.1 Late Submission Deduction

- 10% for late assignments submitted within 1 day late.
- 20% for late assignments submitted within 2 days late.
- 50% for late assignments submitted within 7 days (Mar 27) late.
- 100% for late assignments submitted after 7 days (Mar 27) late.

Questions!

If any more questions, please contact kendwj@hku.hk
or schen59@hku.hk