Project3

Computer Vision (CSI4116-01)

Spring, 2022

Due 19th June, 23:59

Project3-1

Back Propagation

Overview

- Let's implement **back-propagation** without deep learning framework

- You will be given a skeleton code, **project 3_1.py**.

- All codes for training a Multi-layer Perceptron are implemented in the code, except for some functions. You should complete below **4** functions.
  - **__init__()**: Initialize weights and biases here
  - **forward(self, x)**: function for forward propagation
  - **backward(self, x, y_onehot)**: function for back-propagation
  - **step(self)**: function for updating weights and biases

Details

- MLP we’re going to implement consists of an input layer, a hidden layer, and an output layer.

- Hidden layer has the dimension of 128.

- If you run the code after completing the functions, you will be able to see the training progress and the accuracy of your model.

- Weight initialization is quite important when training from scratch. Try various techniques if you want to enhance the performance.

Dataset

- MNIST is a dataset containing images of hand-written digits.

- We will use MNIST for training MLP, so you should design layers suitable for processing MNIST dataset.

- Code for downloading and loading the dataset is set in the skeleton code, so you don’t have to take care of it.

Cautions

- Do not import libraries other than those already imported.
- When completing the functions,
  - Do not use 3rd-party libraries except for **Numpy.random** and **Numpy.dot**
  - Do not change name and number of arguments
- Do not touch functions other than 4 functions mentioned.
- If you violates above cautions, you will get 0 point.

Project3-2

Training a deep neural network

Overview

- We'd like to build a high-performance **deep learning model** for image classification.

- You are provided with a skeleton code('project 3_2.py'), a train set, a validation set, an answer file, and a sample submission file.
  - Validation set will be provided through Kaggle Competition page.

- You should put all your codes for running your model in the skeleton code.

Skeleton Code Details

- **class MyModel(nn.Module)** : Your own model for image classification
- **class MyDataset(Dataset)** : Custom dataset to load data
- **def train()** : Function for training your model
  - After training, you should save the trained model parameter as '**model.pth**'. (Refer to line 36)
- **def main()** : Design main function at your convenience.
- **def test()** : Function for testing the performance of your model.
  - Do not touch the lines for loading trained model(51~56) and saving results(93~100). We will use the same structure when running your model.

Training Phase

- Train set consists of 40,000 images.
- You should train a model to output correct class number for the images.
- Ground truth labels(class numbers) are provided in ‘answer.json’.
- You must save your final model’s parameter as ‘model.pth’ and submit it by 6/19 23:59pm.
- Download: https://drive.google.com/file/d/1Iq4OYPyPjpJ4oF4bp9qCJeeo0zmCbzL1/view?usp=sharing

Testing Phase

- Test set will be open on LearnUs at 6/21 11:00am.

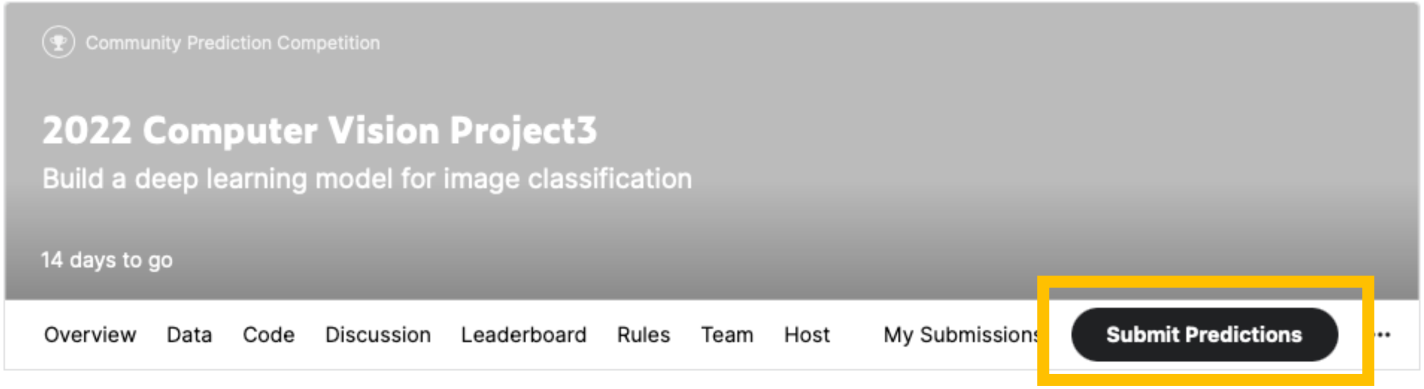
- When you run test(), it should save the test result as ‘result.csv’. Submit the csv file to LearnUs by 6/21 11:29am.
    - Refer to ‘sample.csv’ and the skeleton code for the format of csv file.

- You must test using the model loaded with model parameter you submitted by 6/19. If not, you will get 0 point.

- You can practice this process with validation set.

Validation Phase

- With the validation set, you can check the performance of your model by submitting the result to Kaggle competition page.



Community Prediction Competition

2022 Computer Vision Project3

Build a deep learning model for image classification

14 days to go

- Test set is not the same distribution as the validation set, so you should take the ranking just for a reference.

F1-score

- We will use F1-score to evaluate the performance of your code.

$$\text{F1 Score} = \frac{2 \text{ x } (\text{Precision x Recall})}{\text{Precision + Recall}}$$

- F1-score measures accuracy using precision and recall.
  - Precision: the ratio of true positives to all predicted positives.
  - Recall: the ratio of true positives to all actual positives.

Report

- Explain your implementation for each function with screenshots of the code.
- Explain how you improved the performance of your model.
- Explain how to run your code.

Cautions

- You can’t use predefined models or pretrained weights.

- You can’t use other datasets.

- Model parameter used when testing should be the same as the model parameter you submitted by 6/19. You should load the exact same ‘model.pth’ (submitted one). If the result is not the same, you will get 0 point.

Submission

- ~6/19 23:59pm
  - Submit '{STUDENT_ID}.zip' containing **project3_1.py, project3_2.py, report.pdf,** and **model.pth**.
  - Not following the zip file structure will result in 0 point.
  - Delay Policy - 50% deduction until 6/20 23:59pm

- 6/21 11:00~11:29am
  - **result.csv**
  - Late submission is not allowed, so be prepared for the testing phase.
  - Submitting result.csv without submitting model.pth in advance will be regarded as cheating, resulting in 0 point

Grading Policy

- Total 150 pts
  - Project 3-1 50 pts
    - Implementation 40 pts
    - Training accuracy 10 pts
  - Project 3-2 100 pts
    - Implementation 50 pts
      - Without submitting the result, you can’t get implementation score.
    - Competition result 40 pts
    - Report 10 pts