DTE2502 – Neural Networks :: Graded Assignment 01

Tasks in the assignment

- You are expected to convert a TF code to PyTorch.
- You are provided with a reference code on the course `gitHub` page (https://github.com/krayyalasomayajula/DTE2502). You are expected to insert code in few places as indicated in the code base.
- You are not expected to know what the model is doing to complete the assignment. However, if you are interested to know more about the architecture and the problem the model is trying to solve you may read the `conf_paper.pdf` provided in `gitHub`.

Directory structure of the code base

Files	Description

`./modules/models.py`
`./modules/loss.py`
`./modules/dataset.py`
	The main coding exercises for the assignment will be limited to these files.

`./modules/pyramidpooling.py` `./modules/engine.py`	Supporting Pytorch code for model building

`./utils/phoc_generator.py`
`./utils/__init__.py`
`./utils/map.py`
`./utils/phos_generator.py`
`./utils/Alphabet.csv`
	Supporting utility code for feature building and feature mapping.

`./completeEnv.yaml` `./main.py`	Environment replication and `main(args)` function to start the training

You are expected to code (fill up the code in the indicated places)

- Here you will reproduce the code in *Pytorch*. The *Tensorflow* code for the same is available in the file `train_phoscnet.py` at (https://github.com/anuj-rai-23/PHOSC-Zero-Shot-Word-Recognition)
- Dataloader (`./modules/dataset.py`) :
  - You will be coding `__init__(self)` in `class phosc_dataset(Dataset):`
  - When dealing with a new problem, most of your time will be dedicated to data cleansing and data handling. The data for this assignment has images and feature encoding as well. So you will have to handle the images and csv file parsing in the dataset class. Once you have created the class you can use it for handling training, validation and test datasets.
  - You may adapt the approach presented in the course `GitHub` (`ml_frameworks/pytorch.ipynb`) to debug your implementation.
- Model (`./modules/models.py`) :
  - You will be coding in `__init__(self)` in `class PHOSCnet(nn.Module):`
  - However, one difference is that you will using `TemporalPyramidPooling`, already coded in the `class PHOSCnet(nn.Module)` at line
    - `self.temporal_pool = TemporalPyramidPooling([1, 2, 5])`

    So do not change the above line.
- Loss function (`./modules/loss.py`):
  - You will be coding `forward(self, y: dict, targets: torch.Tensor)` in `class PHOSCLoss(nn.Module):`

- The loss weights are currently multiplied by ‘0’. Replace ‘0’ with the correct type of loss following the *Tensorflow* code.

Dataset to be used:

Directory structure of the datasets

Files	Description

`./dte2502_ga01/train.csv`
`./dte2502_ga01/valid.csv`
`./dte2502_ga01/test_seen.csv`
`./dte2502_ga01/test_unseen.csv`
	These are files in comma separated values format that contain the file names of word images and their corresponding labels in text. You may ignore the author information of this assignment

`./dte2502_ga01/train`
`./dte2502_ga01/valid`
`./dte2502_ga01/test_seen`
`./dte2502_ga01/test_unseen`
	These are the directories that contain the actual word images.

- The datasets are available here (https://drive.google.com/drive/folders/1CPnYT-Sprdguzf6r7ORsIegpVQO4qLLo?usp=sharing).
- Use the small dataset to run your experiments and debugging quickly, once you are confident with your implementation use the larger dataset.

Expected outcome

If you have coded everything correctly the model should train without any problem.

- Use the following command for training:
```
python main.py --mode train \
--model PHOSCnet_temporalpooling \
--train_csv <ds_dir>/dte2502_ga01/train.csv \
--train_folder <ds_dir>/dte2502_ga01/train \
--valid_csv <ds_dir>/dte2502_ga01/valid.csv \
--valid_folder <ds_dir>/dte2502_ga01/valid
```
where `<ds_dir>` is the base directory (`dte2502_ga01_small` if you are using the smaller dataset) where the datasets were extracted.

- The per batch training loss in an epoch will be displayed in the terminal as:
```
loss: 155.11575317382812, step progression: 1/1253, epoch: 3
loss: 139.1514434814453, step progression: 2/1253, epoch: 3
loss: 143.59945678710938, step progression: 3/1253, epoch: 3
loss: 154.96688842773438, step progression: 4/1253, epoch: 3
```

- The validation accuracy per epoch is logged in the `PHOSCnet_temporalpooling/log.csv` this accuracy will be increasing till the model has reached convergence after which it will fluctuate around **0.72**.

- Once the training is complete, use the following command for testing:
```
python main.py --mode test \
--model PHOSCnet_temporalpooling \
--pretrained_weights PHOSCnet_temporalpooling/epochxx.pt \
--test_csv_seen <ds_dir>/dte2502_ga01/test_seen.csv \
--test_folder_seen <ds_dir>/dte2502_ga01/test_seen \
--test_csv_unseen <ds_dir>/dte2502_ga01/test_unseen.csv \
--test_folder_unseen <ds_dir>/dte2502_ga01/test_unseen
```
where `epochxx.pt` is the best model saved as per the validation accuracy

- The output for testing will be something like
```
accuracies of model: PHOSCnet_temporalpooling
Seen accuracies: 0.7249936692833628
Unseen accuracies: 0.7265130412762725
```

Final words

- This assignment is a scaled up version of the `ml_frameworks/pytorch.ipynb` example in the `github` presented in lecture `DTE-2502_wk04lec01_ML_frameworks.`
- The libraries and modules given in the `conda` environment is only provide for your guidance. Feel free to create new `conda` environments with more libraries for your study and experimentation. Try out and environment with Keras + TF as suggested in the original GitHub to understand the original TF code if it helps.
- However the final code submitted will be tested and graded in the `conda` environment provided in the `yaml` file and as such should not have any more dependencies.
- The model takes slightly < 6GB when running on the GPU. In case this does not fit into your GPU. Here are some ways to build a smaller model (in `./modules/models.py`) first then scale it up later
  - Remove the following layers in `self.conv` block:
    - Four conv2d layers `Conv2D(256, (3, 3), padding='same', activation='relu')`
    - And two conv2d layers `Conv2D(512, (3, 3), padding='same', activation='relu')`
  - Remove the following layers in `self.phos, self.phoc` blocks:
    - Two Linear layers: `Dense(4096, activation='relu')`, `Dropout` that follows them
  - Use the `--batch_size 2` option in the training.
  - These steps will reduce the model parameters and the input batch fed to the model. Thus making the model smaller.
- In case you want to use `pdb` to step into your code when training use `--num_workers 0` option to stop multi-processing on CPU side.

Deadline

- The deadline for submission is 15-Oct, 23:59.
- After the deadline, groups having trouble with the GA01 can ask for hints and possible extension.
- However, depending on the progress made, help received and the extension agreed upon there will a cap on the final grade to be received for this assignment.
- The cap will apply to the score of GA01 only and not for any other module.