CS6203

Dipika Singhania
A0195129X
dipika16@comp.nus.edu.sg

November 2019

1 Task details:

Our task for this assignment is to try to learn the relationship between datasets and the models that performs best on them. We took the an approach that we thought might work and we present the details of the implementation and commands to reproduce the results. The github repository of the project is https://github.com/dipika-singhania/model_selection

2 Project folder structure:

The project folder structure is created to systematically handle different levels of files that this project requires. The structure is as below:

```
Base/ .......................................... Main project folder
├─ dataset_loaders/ ............................ Directory for pytorch dataloaders
│  └─ image_classification/
│     ├─ data_config.py ........................ Script that includes data dictionary
│     ├─ utils.py
│     ├─ cars.py ............................... Example dataloder definition of cars dataset
│     └─ ...
├─ dataset_pool/ ............................... Location of downloaded datasets
│  ├─ cifar10/ ................................. Example downloaded data folder
│  └─ ...
├─ glove ....................................... Directory containing word embeddings
├─ logs ........................................ Distilled model dataset attributes from model runs
├─ meta_learning_features/ ..................... Directory for meta features
│  ├─ target_embeddings/ ....................... Directory for dataset level target features
│  │  └─ cars_target_embedding.pkl ............. Example of cars dataset target feature
│  ├─ dataset.csv .............................. Example of csv file containing features/attributes
│  └─ ...
├─ meta_model_bin/ ............................. Scripts to generate meta features/files
│  ├─ get_labels_to_vec.py ..................... Example of script to generate meta features
│  └─ ...
├─ meta_model_pool/ ............................ Directory to save meta model files
├─ model_bin/ .................................. Directory containing scripts to run models on dataset
│  └─ image_classification/
│     └─ models_pool.py ........................ Script containing model dictionary
├─ model_pool/ ................................. Directory to save model checkpoints run on datasets
│  └─ resnet18_flower.pth.tar .................. Example of a saved checkpoint
├─ outputs_logs/ ............................... Runtime logs can be saved at this directory
└─ run_scripts/ ................................ Scripts that user needs to run
```

3 Important scripts and folder for user:

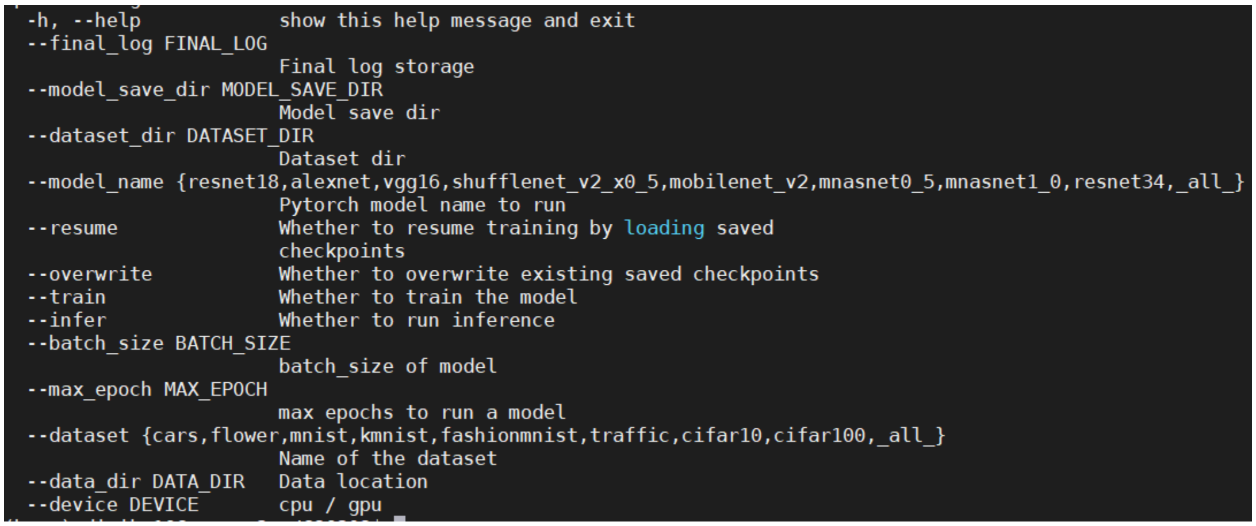
Out of all the described folders and scripts there are only hand-full of them that are really necessary for an user. We are listing them below.

3.1 run_scripts/run_model_on_dataset.py

This script is used to run actual models (e.g. ***Alexnet, ResNet18***) on a dataset (e.g. ***Cifar10, Mnist***) to get results. The arguments of the code can be seen in this screenshot and can be obtained by running

```
python run_scripts/run_model_on_dataset.py --h
```

The example output of this command lists all the argument descriptions



```
  -h, --help            show this help message and exit
  --final_log FINAL_LOG
                        Final log storage
  --model_save_dir MODEL_SAVE_DIR
                        Model save dir
  --dataset_dir DATASET_DIR
                        Dataset dir
  --model_name {resnet18,alexnet,vgg16,shufflenet_v2_x0_5,mobilenet_v2,mnasnet0_5,mnasnet1_0,resnet34,_all_}
                        Pytorch model name to run
  --resume              Whether to resume training by loading saved
                        checkpoints
  --overwrite           Whether to overwrite existing saved checkpoints
  --train               Whether to train the model
  --infer               Whether to run inference
  --batch_size BATCH_SIZE
                        batch_size of model
  --max_epoch MAX_EPOCH
                        max epochs to run a model
  --dataset {cars,flower,mnist,kmnist,fashionmnist,traffic,cifar10,cifar100,_all_}
                        Name of the dataset
  --data_dir DATA_DIR   Data location
  --device DEVICE       cpu / gpu
```

The only arguments that are necessary (default will be used for the rest of the arguments)

- model_name: Either use one of the models from the model list e.g. ***alexnet*** or use ***_all_*** to train on all available models sequentially.
- dataset: Either use one of the datasets from the dataset list e.g. ***cifar10*** or use ***_all_*** to train on all available datasets sequentially.
- train: Whether to perform training
- resume: Whether to resume training from already available model checkpoints
- overwrite: Whether to overwrite already saved checkpoints while training from scratch.
- infer: Whether to perfrom inference on test dataset.
- max_epoch: set epoch count for training

Example command is

```
CUDA_VISIBLE_DEVICES=0 python run_scripts/run_model_on_dataset.py
--model_name=_all_ --train --overwrite --infer --max_epoch=20
--dataset=cars
```

The above command performs training sequentially from scratch for all models on the dataset cars and performs inference after training.

```
CUDA_VISIBLE_DEVICES=0 python run_scripts/run_model_on_dataset.py
--model_name=resnet18 --infer --max_epoch=20
--dataset=cars
```

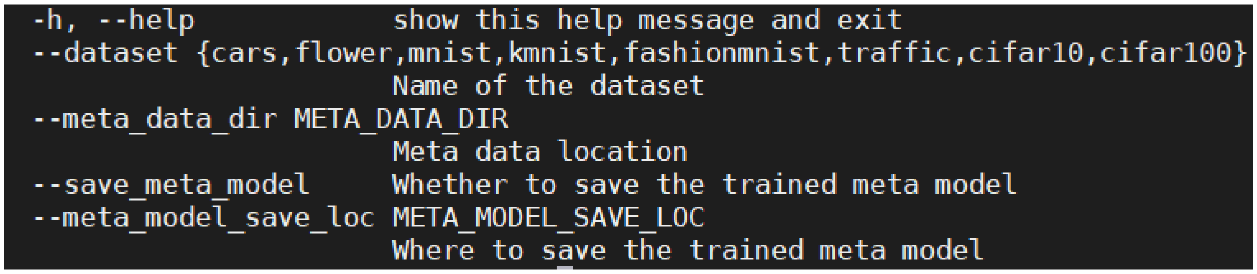
The above command performs just inference for the model resnet18 on the dataset cars, by loading the respective saved checkpoint.

3.2 run_scripts/suggest_model.py

This script is used to leverage the meta-model learned from previous runs of models to suggest a model that should be run on a given dataset. If this is run with a specific dataset name, then this program suggests a model based on a model created based on other runs that does not include the dataset provided by the user. To get a description of what the arguments are of this code one can run.

```
python run_scripts/suggest_model.py --h
```

which produces the output

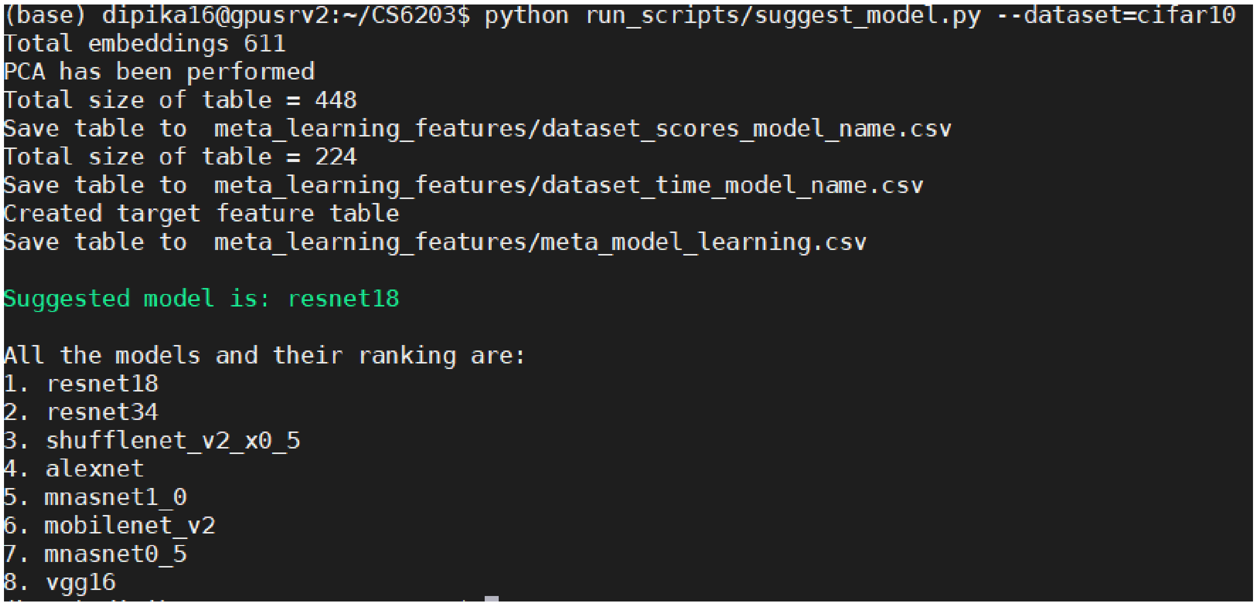


```
  -h, --help            show this help message and exit
  --dataset {cars,flower,mnist,kmnist,fashionmnist,traffic,cifar10,cifar100}
                        Name of the dataset
  --meta_data_dir META_DATA_DIR
                        Meta data location
  --save_meta_model     Whether to save the trained meta model
  --meta_model_save_loc META_MODEL_SAVE_LOC
                        Where to save the trained meta model
```

One example command required for this script is

```
python run_scripts/suggest_model.py --dataset=cifar10
```

which gives the suggested models obtained from consulting the meta model. It also lists all the models with there respective ranking so that next best options are also visible to the user.



```
(base) dipika16@gpusrv2:~/CS6203$ python run_scripts/suggest_model.py --dataset=cifar10
Total embeddings 611
PCA has been performed
Total size of table = 448
Save table to  meta_learning_features/dataset_scores_model_name.csv
Total size of table = 224
Save table to  meta_learning_features/dataset_time_model_name.csv
Created target feature table
Save table to  meta_learning_features/meta_model_learning.csv

Suggested model is: resnet18

All the models and their ranking are:
1. resnet18
2. resnet34
3. shufflenet_v2_x0_5
4. alexnet
5. mnasnet1_0
6. mobilenet_v2
7. mnasnet0_5
8. vgg16
```

4 Important steps for the user to run the necessary scripts:

4.1 Setting PYTHONPATH:

Before running any scripts it is necessary to set the pythonpath. The user must run the following commands which first goes to the root project directory and then sets **PYTHONPATH** to the required current working directory.

```
cd project_base
export PYTHONPATH=$PWD
```

4.2 Adding dataset/models to required files:

If the user wants to make use of any more datasets or models than the ones available, then the user must add the required definitions to the respective directory.

4.2.1 Dataset:

If a new dataset is required, then a function returning the dataloaders (train, validation, test) must be written inside a *.py* file in the directory ***dataset_loaders/image_classification/*** and the respective function along with the dataset name must be included in the dictionary inside ***dataset_loaders/image_classification/data_config.py***. Also the dataset needs to be added into ***meta_learning_features/dataset.csv***

4.2.2 Model:

Additional models than the already added ones and which are available in pytorch can be added into the dictionary residing in ***dataset_loaders/image_classification/models_pools.py***

4.3 Running training/inference with newly added models on newly added dataset:

This step is not needed if the user is working with already existing datasets and models as the existing models has already been run on the existing datasets and results have been saved. If required then the user needs to ***run_scripts/suggest_model.py*** Details has been given above.

4.4 Use meta-model suggestion for a dataset:

In this step user needs to run the script ***run_scripts/suggest_model.py*** Details of this script has been described above.