Documentation and User Manual

0 Contents:

1: Imports [1]

2: Classes  
 2.1: TipDataset [2]  
 2.2: FirstNet [2]  
 2.3: SecondNet [3]  
 2.4: ThirdNet [3]

3: Functions  
 3.1: dataldr\_make [4]  
 3.2: trn [5]  
 3.3: val [5]  
 3.4: execute [6]  
 3.5: output [6]  
 3.6: main [7]

4: Global Variables [7]

5: User Manual [7]

1 Imports:

from sklearn.metrics import explained\_variance\_score as evs  
from sklearn.model\_selection import ShuffleSplit

from livelossplot import PlotLosses  
import matplotlib.pyplot as plt

import numpy as np  
import math

import pandas as pdimport torch

import torch.nn as nn  
import torch.nn.functional as F  
from torch.utils.data import TensorDataset, DataLoader, Dataset

2 Classes:

2.1 TipDataset(Dataset)

Custom dataset to read and hold the tip data as data (features) and lbls (targets/labels)  
Applies divisor to SIF values to bring down to order 10s

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| csv\_file | File address for the csv containing the data | string |  |
| sif | Selects SIF element [1, 2, 3] for training | integer | 0 |
| divisor | Amount by which to divide the SIF elements | integer | 1e-5 |
|  |  |  |  |
| Parameters |  |  |  |
| sif\_idx | The actual index in the csv for the selected SIF | integer |  |
| select | Allows selection of non-contiguous features | array |  |
|  |  |  |  |
| Attributes |  |  |  |
| tips | The raw csv file read by pandas | pandas dataframe |  |
| data | The features from the csv | tensor |  |
| ndat | The number of features in the set | integer |  |
| lbls | The targets (labels) from the csv | tensor |  |

2.2 FirstNet(nn.Module)

A simple neural network for training  
Architecture: input -> 25(activation) -> 25(activation) -> 1

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| features | Number of training features | integer | 7 |
|  |  |  |  |
| Parameters |  |  |  |
| z\* | Hidden layers | tensor |  |
| a\* | Activation layers | tensor |  |
|  |  |  |  |
| Attributes |  |  |  |
| lin\_\* | Linear layer | nn module |  |
| activ | The activation layer | nn module |  |
|  |  |  |  |
| Returns |  |  |  |
| z3 | The final hidden layer | tensor |  |

2.3 SecondNet(nn.Module)

A simple neural network for training, but with an extra hidden layer  
Architecture: input -> 25(activation) -> 25(activation) -> 10(activation) -> 1

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| features | Number of training features | integer | 7 |
|  |  |  |  |
| Parameters |  |  |  |
| z\* | Hidden layers | tensor |  |
| a\* | Activation layers | tensor |  |
|  |  |  |  |
| Attributes |  |  |  |
| lin\_\* | Linear layer | nn module |  |
| activ | The activation layer | nn module |  |
|  |  |  |  |
| Returns |  |  |  |
| z4 | The final hidden layer | tensor |  |

2.4 ThirdNet(nn.Module)

A simple neural network for training, but with massively increased neurons  
Architecture: input -> 25(activation) -> 100(activation) -> 200(activation) -> 50(activation) -> 1

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| features | Number of training features | integer | 7 |
|  |  |  |  |
| Parameters |  |  |  |
| z\* | Hidden layers | tensor |  |
| a\* | Activation layers | tensor |  |
|  |  |  |  |
| Attributes |  |  |  |
| lin\_\* | Linear layer | nn module |  |
| activ | The activation layer | nn module |  |
|  |  |  |  |
| Returns |  |  |  |
| z5 | The final hidden layer | tensor |  |

3 Functions:

3.1 dataldr\_make

A routine to construct the dataloaders for training, validation and testing, as well as features and targets for testing  
Uses ShuffleSplit to create the sets

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| ICGT\_tips | Holds the total dataset | tensor dataset |  |
| trn\_batch\_size | The training batch size | integer |  |
| tst\_batch\_size | The test batch size | integer |  |
|  |  |  |  |
| Parameters |  |  |  |
| ss | The ShuffleSplit object | ShuffleSplit |  |
| idx | The indices returned by the shuffle split | [integer, integer] |  |
| f\_trn | The training features | tensor |  |
| t\_trn | The training targets | tensor |  |
| f\_vtt | The validation and testing features | tensor |  |
| t\_vtt | The validation and testing targets | tensor |  |
| f\_val | The validation features | tensor |  |
| t\_val | The validation targets | tensor |  |
| f\_tst | The testing features | tensor |  |
| t\_tst | The testing targets | tensor |  |
| ICGT\_tips\_trn | The training tensor dataset | tensor dataset |  |
| ICGT\_tips\_val | The validation tensor dataset | tensor dataset |  |
| ICGT\_tips\_tst | The testing tensor dataset | tensor dataset |  |
| trn\_ldr | The training dataloader | dataloader |  |
| val\_ldr | The validation dataloader | dataloader |  |
| tst\_ldr | The testing dataloader | dataloader |  |
|  |  |  |  |
| Returns |  |  |  |
| trn\_ldr | The training dataloader | dataloader |  |
| val\_ldr | The validation dataloader | dataloader |  |
| tst\_ldr | The testing dataloader | dataloader |  |
| f\_tst | The testing features | tensor |  |
| t\_tst | The testing targets | tensor |  |

3.2 trn

This routine handles the training loop

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| mdl | The model to be trained | nn.Module |  |
| opti | The optimiser object | optim object |  |
| crit | The criterion (loss) function | nn loss object |  |
| ldr | The dataloader for training | dataloader |  |
|  |  |  |  |
| Parameters |  |  |  |
| trn\_los | Tracks the training loss | float |  |
| trn\_acc | Tracks the training accuracy using evs | float |  |
| X | The feature set for the current datapoint | tensor |  |
| y | The target for the current datapoint | tensor |  |
| y\_pred | The predictions returned by the model | tensor |  |
| los | The calculated loss for the current datapoint | nn loss object |  |
|  |  |  |  |
| Returns |  |  |  |
| trn\_los\_avg | The averaged training loss | float |  |
| trn\_acc\_avg | The averaged training accuracy | float |  |

3.3 val

This routine handles the validation loop

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| mdl | The model to be validated | nn.Module |  |
| crit | The criterion (loss) function | nn loss object |  |
| ldr | The dataloader for validation | dataloader |  |
|  |  |  |  |
| Parameters |  |  |  |
| val\_los | Tracks the validation loss | float |  |
| val\_acc | Tracks the validation accuracy using evs | float |  |
| X | The feature set for the current datapoint | tensor |  |
| y | The target for the current datapoint | tensor |  |
| y\_pred | The predictions returned by the model | tensor |  |
| los | The calculated loss for the current datapoint | nn loss object |  |
|  |  |  |  |
| Returns |  |  |  |
| val\_los\_avg | The averaged validation loss | float |  |
| val\_acc\_avg | The averaged validation accuracy | float |  |

3.4 execute

This routine is responsible for the entire training process, and handles in-training plotting

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| model | The model to be trained | nn.Module |  |
| n\_epochs | The number of epochs the model should be trained for | integer |  |
| trn\_ldr | The training dataloader | dataloader |  |
| val\_ldr | The validation dataloader | dataloader |  |
| opti | The optimiser object | optim object |  |
| crit | The criterion (loss) function | nn loss object |  |
| plot | A flag denoting whether in-training plotting occurs | boolean |  |
|  |  |  |  |
| Parameters |  |  |  |
| liveloss | Responsible for in-training plotting, activated by plot | PlotLosses() object |  |
| epoch | The current epoch number | integer |  |
| logs | Holds the log data for the current epoch | dict |  |
| trn\_los | The training loss for the current epoch | float |  |
| trn\_acc | The training accuracy for the current epoch | float |  |
| val\_los | The validation loss for the current epoch | float |  |
| val\_acc | The validation accuracy for the current epoch | float |  |
|  |  |  |  |
| Returns |  |  |  |
| model | The final, trained model | nn.Module |  |

3.5 output

This routine handles the error analysis and output

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| model | The trained model to be analysed | nn.Module |  |
| f\_tst | The features for testing | tensor |  |
| t\_tst | The targets for testing | tensor |  |
|  |  |  |  |
| Parameters |  |  |  |
| output | The output from the model when given f\_tst | tensor |  |
| truth | The true values for f\_tst, from t\_tst | tensor |  |
| errors | A list of the absolute errors for each datapoint | [float] |  |
| outputs | A list version of output | [float] |  |
| truths | A list version of truth | [float] |  |
| avg\_error | The average absolute error | float |  |
| avg\_value | The average of the absolute targets | float |  |
| max\_error | The largest absolute error in the dataset | float |  |
| bad\_index | The index for the max\_error datapoint | integer |  |
| datalen | The length of the dataset, used for averaging | integer |  |
| i | Indexing for the loop | integer |  |
| error | The absolute error for the current datapoint | float |  |
| mini | The minimum value of both truths and outputs | float |  |
| maxi | The maximum value of both truths and outputs | float |  |

3.6 main

This is the core routine that handles everything else

|  |  |  |  |
| --- | --- | --- | --- |
| Arguments | Description | Datatype | Default |
| file\_name | The file address for the data csv | string |  |
| mdl | The ID for the model to be used | integer | 3 |
| n\_epochs | The number of epochs to run training for | integer | 500 |
| sif | Which stress intensity factor element to be predicted | integer | 0 |
| plot | Flag that determines whether to do mid-training plots | boolean | True |
| test | Flag for whether to evaluate the trained model | boolean | True |
| save | Flag for whether to save the trained model | boolean | False |
| mID | Model ID to be affixed to the save file’s name | to-string | 1 |
| load | Flags whether to train or load a model | boolean | False |
| ld\_mdl | Contains the required data for loading a model  All models are saved as model\*\_SIF\*\*\_\*\*\*.pt, where \* is the model architecture, \*\* is the SIF element and \*\*\* is the mID. When loading, ld\_mdl should be [\*,\*\*,\*\*\*] | list | [3,1,1] |
|  |  |  |  |
| Parameters |  |  |  |
| trn\_batch\_size | The batch size for training | integer |  |
| tst\_batch\_size | The batch size for testing and validation | integer |  |
| model | The selected model architecture | nn.Module |  |
| opti | The optimiser function | optim object |  |
| crut | The criterion function | nn loss object |  |
| ICGT\_tips | The custom dataset that reads and holds the input | Dataset |  |
| trn\_ldr | The dataloader for training | dataloader |  |
| val\_ldr | The dataloader for validation | dataloader |  |
| tst\_ldr | The dataloader for testing [vestigial] | dataloader |  |
| f\_tst | Input features for testing | tensor |  |
| t\_tst | Targets for testing | tensor |  |
| cancel | Flag set to true if load operation fails | boolean |  |

4 Global Variables:

device: Holds which execution device is to be used – CPU or CUDA GPU : string

5 User Manual:

Usage of this solution is trivially easy. If one desires to train a new model, simply write *main(\*file address of the data\*, \*any of the flags and arguments one wishes to choose, as described in Section 3.6\*, load=False, \*ld\_mdl is irrelevant when not loading\*)*, and the solution will train a new model and perform whatever other tasks the set flags dictate.

Contrarily, if one wishes to review a previously trained model, one must write *main(\*file address of the data with which to perform analysis\*, \*these arguments are now irrelevant\*, load=True, ld\_mdl=[\*model ID\*, \*SIF element\*, \*mID of desired model\*])*, where details of the ld\_mdl are again found in Section 3.6. This will result in the loading and analysis of the selected model.