Training the classifier & confidence-based review

In this file we demonstrate how to train the classifier, generate predictions, launch the confidence-based review, and export the project results. We do so as a continuation of the example in the NewProject.mlx script. In that project, we demonstrated how to create a project, import video, extract its frames and features, and launch the annotator, and we did so using a demonstration project with a small number of short, video without annotations. Here, we provide annotations and already extracted features corresponding to the full-length videos in the home-cage dataset, and use these annotations to run steps 6-11 of the project workflow (see below).

Table of Contents

Loading/generating the clip table

After the video features have been generated and a subset of the video annotated (the former we expedite by using the downloaded features, and the latter by using the dataset labels), we are ready to train the classifier.

To do so, we first contruct a DeepAction project by passing the folder where our project is stored to the DeepActionProjectClass.

```
projectFolder = '/Users/harriscaw/Documents/Behavior classification/Projects/example_p
project = DeepActionProject(projectFolder);
```

```
BehaviorTable(project)
```

```
project = project.GetClassifierData();
```

```
Creating clips and loading clip data to train the classifier...
  Dividing project annotations & features into clips
    Collecting annotation status for 12 videos...complete
  Collecting feature status for 12 videos  100.0% |████████████████████████| 12/12 [00:21.6 < 00:00.0] (0
  - Created 638 clips with a target length of  (48 frames on avg. with FPS of 48.0)
  - Created 58 clips with a target length of 01:00 (1798 frames on avg. with FPS of 30.0)
    - 615 are human-annotated
    - 23 do not have human annotations
  Loading clip data...
    Loading features and reducing dimensionality  100.0% |████████████████████████| 12/12 [00:39.3 < 00:0
  - Loading annotations... complete
```

```
% project = MarkClipsAsIncomplete(project, 150);
```

Splitting into train/validation/test/unlabeled sets

After we have generated clips for the project, we select clips for use in training, validation, and testing. The training clips will be used directly to train the classifier. The validation clips will be used to tune the classifier training and train the confidence scorer. And the test clips will be used to evaluate both the classifier and the confidence scorer prior to the confidence-based review.

The proportion of labeled data in each of these sets is specified in the [Evaluation] section of the config.txt file. The TrainProportion, ValidationProportion, and TestProportion parameters govern the proportion of clips in each set. By default, 60 percent of data is used in training, and 20 percent in validation and testing.

To generate these sets, we run the SplitClipData method on our project:

```
project = project.SplitClipData();
```

```
Loading network data...
  - Splitting into train/validate/test splits
Clip data split into sets:
    Train: 431 clips (70%)
    Validation: 92 clips (15%)
    Test: 92 clips (15%)
```

Setting up the classifier

After the clip data have been split into sets, we create the classifier to use in training. To do so, we first specify the sequence-to-sequence LSTM via the parameters in the [Classifier] section of the configuration file. We also specify the length of the sequences to further divide the clips into. The full set of training parameters is specified using the following configuration file parameters:

▼ `SequenceLength=450`

Length (in frames) of sequences to be input into RNN.

Default: `450`

▼ `NumberHiddenUnits=64`

Number of hidden units in each layer of the BiLSTM.

Default: `64`

▼ `NumberLayers=2`

Number of BiLSTM layers.

Default: `2`

▼ `DropoutRatio=0.5`

Dropout probability of dropout layers located after each BiLSTM layer.

Default: `0.5`

▼ `ClassificationLayer=cross-entropy`

Classification loss function to use.

- Options:
  - `cross-entropy` : standard cross-entropy loss function
  - `weighted cross-entropy` : cross-entropy loss, where loss is inversely proportional to the incidence of the class

Default: `cross-entropy`

```
project = project.SetUpClassifier('showplots', true);
```

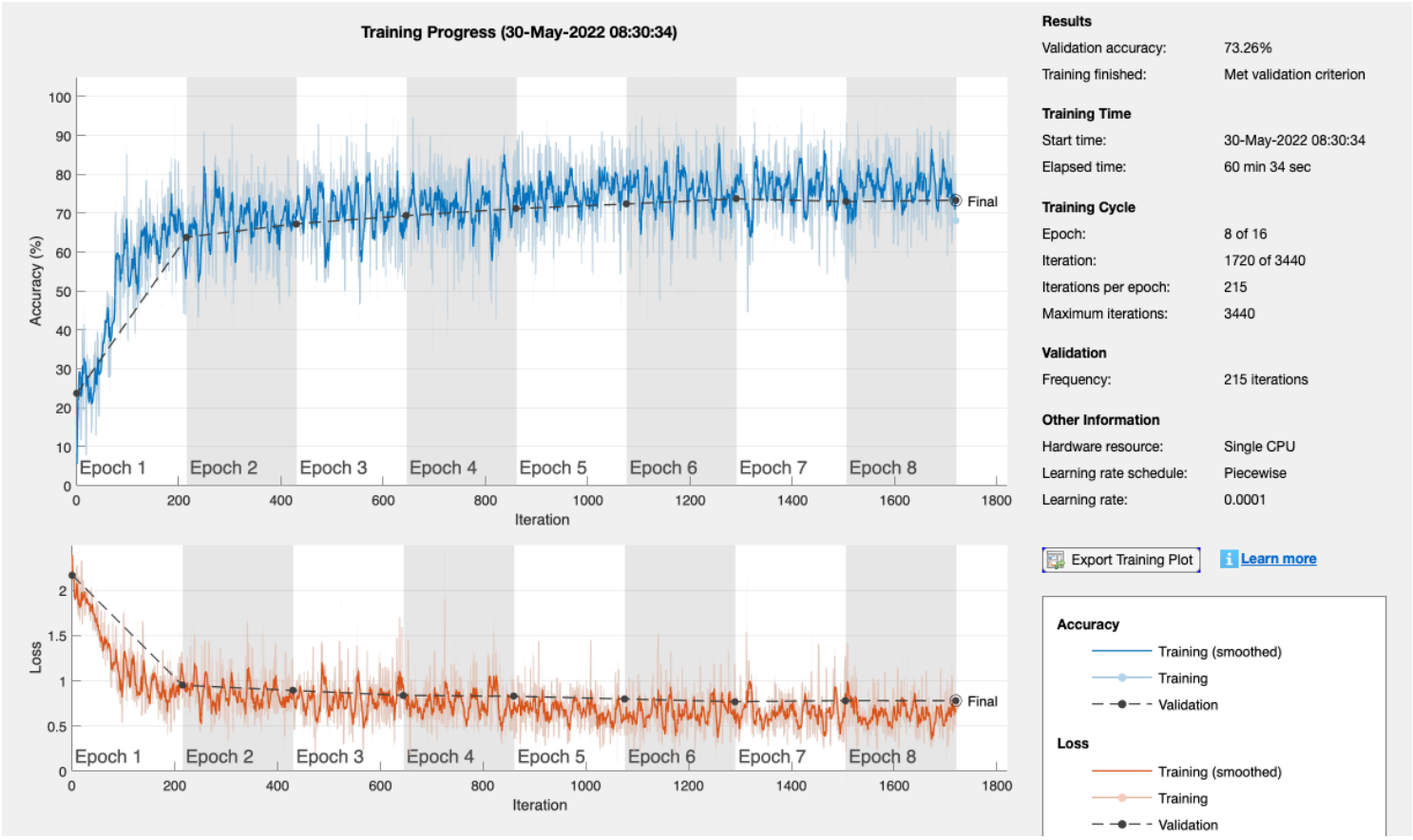
```
Setting up network...
  Training options
    - MiniBatchSize: 8
    - InitialLearnRate: 0.001000
    - LearnRateDropPeriod: 4
    - LearnRateDropFactor: 0.100
    - MaxEpochs: 16
  Network options
    - Classification layer: cross-entropy
    - NumberLayers: 2
    - NumberHiddenUnits: 64
```

```
    - DropoutRatio: 0.500
  Network input
    - NumItersEpoch: 215
    - NumberFeatures: 512
    - NumberClasses: 9
```

Training the classifier

```
% project = project.TrainNetwork()
project = project.TrainClassifier();
```

```
Training on single CPU.
|=========================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Ba
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |
|=========================================================================================================================|
|       1 |           1 |       00:00:23 |        5.52% |       23.68% |       2.3871 |       2.1645 |
|       1 |         215 |       00:08:27 |       68.02% |       63.82% |       0.8060 |       0.9482 |
|       2 |         430 |       00:15:48 |       72.54% |       67.26% |       0.8249 |       0.8896 |
|       3 |         645 |       00:22:57 |       63.11% |       69.36% |       1.0240 |       0.8349 |
|       4 |         860 |       00:30:12 |       61.50% |       71.15% |       0.9328 |       0.8258 |
|       5 |        1075 |       00:37:28 |       76.84% |       72.36% |       0.5979 |       0.7954 |
|       6 |        1290 |       00:44:56 |       57.87% |       73.68% |       1.1824 |       0.7660 |
|       7 |        1505 |       00:52:39 |       53.31% |       72.98% |       1.2596 |       0.7778 |
|       8 |        1720 |       01:00:09 |       68.10% |       73.31% |       0.7128 |       0.7771 |
|=========================================================================================================================|
Training finished: Met validation criterion.
```




```
Network training completed
  - Total training time:       01:00:44
  - Final validation accuracy: 73.3%
  - Max validation accuracy:   73.3%
```

```
Generating clip predictions... complete
Evaluating network...
  - Generating train set predictions
  - Generating validation set predictions
  - Generating test set predictions
       Overall performance
 ==============================
 |          | Accuracy | F1    |
 +----------+----------+-------+
 | Train    | 0.781    | 0.688 |
 | Validate | 0.738    | 0.653 |
 | Test     | 0.729    | 0.653 |
 +----------+----------+-------+

                            Behavior results (test set)
 ================================================================================
 |               | Precision | Recall | TruePositiveRate | FalsePositiveRate | F1    |
 +---------------+-----------+--------+------------------+-------------------+-------+
 | drink         | NaN       | 0.000  | 0.000            | 0.000             | NaN   |
 | eat           | 0.810     | 0.642  | 0.642            | 0.010             | 0.717 |
 | groom         | 0.660     | 0.802  | 0.802            | 0.122             | 0.724 |
 | hang          | 0.908     | 0.857  | 0.857            | 0.007             | 0.882 |
 | micromovement | 0.655     | 0.611  | 0.611            | 0.125             | 0.632 |
 | rear          | 0.757     | 0.731  | 0.731            | 0.030             | 0.744 |
 | rest          | 0.854     | 0.948  | 0.948            | 0.027             | 0.898 |
 | walk          | 0.726     | 0.557  | 0.557            | 0.024             | 0.630 |
 +---------------+-----------+--------+------------------+-------------------+-------+
```

Training the confidence-based review

```
% project = project.GenerateConfidenceScores();
```

```
Intializing confidence scorer
Training confidence scorer using TemperatureScaling
  - Sequence calibrator trained
  - TemperatureScaling scorer
    - Training set
    - Test set

         Confidence score performance
 ============================================
 | Set   | MSD    | MAE   | ReviewEfficiency |
 +-------+--------+-------+------------------+
 | Train | 0.035  | 0.093 | 0.784            |
 | Test  | -0.007 | 0.101 | 0.814            |
 +-------+--------+-------+------------------+
```

Running the confidence-based review

```
project = project.LaunchAnnotator()
```

Final checks & data export

```
project.CreateLabeledClips('PlaybackSpeed', 3, 'Scale', 1)
```

```
project.ExportAnnotations()
```