Multi-Modal Signal Classifier - README

Landon Buell

October 2020

Abstract

The *Signal Classifier* program uses a hybrid neural network architecture to attempt to classify time-series signals to a set of pre-determined classes (defined by the input labels). For the purpose of the algorithm's development, input samples were assumed to be .wav files of musical instruments performing one note, or several shorter notes in succession- totaling at around 100,000 - 200,000 samples each, sampled at 44.1 kHz.

1 Getting the Signal Classifier on Your Machine

The *Signal Classifier* program is stored in a GiHub Repository. As of Fall 2020, the repo can be found at

https://github.com/landonbuell/Buell-Senior-Thesis

This repository can be pulled to your machine any number of ways, reccommended via github. (Git Clone, GitHub Desktop, etc.) Doing so will create new local directory on your machine. The "MAIN" python program itself is a standard *.py* file located at:

.../Buell-Senior-Thesis/SignalClassifier/ClassifierHybrid/ClassifierHybridMain.py

2 Command Line Arguments

The Classifier requires 6 command-line arguments to function properly. These are:

2.1 readPath (str/path)

The local directory path where "read" data files are stored. Each file in this path is a a particularly formatted csv file. The csv/excel files are formatted as follows:

Full (local) directory path to .wav file	Integer Class name	String Class Name

The folder should otherwise be empty because CSV Files formatted any other way will not be handled, and the program will crash. As an example, a CSV file may contain a first few lines that resemble:

Fullpath	Target Int	Target Str
C:/Users/.../GUITAR.C5.wav	16	GUITAR
C:/Users/.../BASSCLARINET.As4.wav	4	BASSCLARINET
C:/Users/.../CLARINET.Cs5.wav	8	BBCLARINET
C:/Users/.../VIOLIN.A4.wav	34	VIOLIN

The repository directory

.../Buell-Senior-Thesis/SignalClassifier/Target-Data

is an example of a valid directory, which each file being well-formatted. There are two a python scripts in:

.../Buell-Senior-Thesis/SignalClassifier-Preprocessing/FileLabelEncoder

That will help with creating these files. See the "MAIN" file for arguments.

2.2 exportPath (str/path)

The local directory path where progress data is exported to. When finished, this path will contain *.csv* files detailing the training history of the model at each epoch as well ads the the label predictions of each sample.

2.3 modelPath (str/path)

The local directory oath where the TensorFlow models are exported to. Each model will be saved in it's own sub-folder. This is where the program will look for the model to re-load it for future train/predict sessions.

2.4 ProgramMode (str)

Indicates which mode the classifier program is executed as. Must be in ["train","predict","train-predict"].

In *train* mode, all collected samples will be used to train the classifier model. A "TRAININGHISTORY" file will be exported to "exportPath". In *predict* mode, all collected samples will be used to run predictions with the classifier model. A "PREDICTIONS" file will be exported to "exportPath". In *train-predict* mode, a 90% training, and 10% testing split is made pseudo-randomly (This ratio can be changed, by adjusting a hard coded parameter in the train/test mode class). The Classifier then runs *train* mode and *test* mode with the respective data subsets.

2.5 modelName (str)

String representation used to name the classifier model in a human-readable format. Another Tensorflow/Keras script can be used to reload this model in it's trained state for future or external use.

2.6 newModel (bool)

Determines weather or not to create a NEW model of the given name. If *true*, then any existing model of the same name is overwritten (cannot be recovered easily!). If *False*, the program looks for an existing model of the name to resume training/predictions given last saved state.

2.7 When Running

(I still have to finish this...)

2.8 What to do with the Output

(And this...)