QuickStart Guide: Echo LVH Pipeline

Contents

Installation

This repo can be obtained by directly cloning the existing git repository. Create a directory in which to store the downloaded package and run:

```
# git clone https://github.com/jamessli/ECHO_LVH_PACKAGE/ my_directory
```

In the package directory, locate the requirements.txt file. This contains all of the necessary dependencies utilized by the package. Install everything by running:

```
# pip install -r requirements.txt
```

Directory Structure

The package will enable users to designate the directory(s) in which the input data and output data will be read from and written to respectively. The input data must be in the form of video files contained within a set directory hierarchy.

Set up the directory by creating a directory and within, set up a structure as follows:

```
my_directory/
    studies/
        test/
            Amyloidosis/
            HCM/
            HTN/
        train/
            Amyloidosis/
            HCM/
            HTN/
```

Within the individual disease labeled directories in the training directory, upload all of the training files in the form of:

```
study_name/
    video.mp4
```

These videos within the training data set will be automatically processed into a training/validation set split at approximately 85:15 percentage. While initially, 15% of the total studies will be relocated to the test set automatically.

Training

Training the models on the given training and test data sets can be done with a single command as follows:

```
#python main.py --train my_dir --device my_device
```

In this case, my_dir represents the directory containing the studies as outlined in the previous section my_device is the name of any inference device/devices in the server in the form GPU:0, CPU:0…

Entering a single device as input will train on a single device. Entering multiple devices will attempt to create a parallel training environment with those devices. Entering nothing will default to CPU:0. This however, will likely fail with memory insufficiency unless a small batch size is provided.

In addition to training with the default commands as show, the package can also consume a hyperparameters.txt file that contains hyperparameter values that overwrites the native ones provided in the script. This file is of the format:

```
batch_size = int
epochs = int
weight_decay = float
learnable_layers = float
optimizer = string
learning_rate = float
learning_rate_decay = int
dropout_rate = float
```

Where:

batch_size - the batch size

epochs - the number of epochs of training

weight decay - the severity of l2 regularization

learnable_layers - the number of layers with tunable weights

optimizer - the deep learning optimizer

learning_rate - the initial learning rate of the optimizer

learning_rate_decay – the number of epochs until the learning rate is halved

dropout_rate – the severity of dropout chance after each layer.

Testing

Testing the models on the given training and test data sets can be done with a single command as follows:

```
#python main.py --test my_dir --device my_device
```

Where my_dir represents the location of the studies directory where the test and train directories are located.

Demo

A visual demonstration is available through streamlit that shows the inferencing and video generation in real time on a single case study. This can be done through the following command.

```
# streamlit run video_demo.py
```

At the top, enter the path to the case study and select the inference device from the drop-down menu.