Activity Recognition using Deep Learning (Tensorflow)

J.Vikranth Jeyakumar

Setting Up Local Tensorflow Environment

- Use Anaconda to install Python
- Identify your machine config (Note AMD Graphic cards don’t work)
- Create a virtual environment with the required python version
- Activate the virtual environment
- Conda install tensorflow-gpu

This eliminates the hassle of installing Cuda!

Easy to update

- If you need the nightly build or the latest version:
  - Conda install cuda toolkit
  - Pip install tensorflow_version

Data Preprocessing for Human Activity Recognition

- Split data into windows
- Hyperparameters:
  - Window size
  - Stride
- Final shape of data should be:
  - (n_samples, window length, features)
- Labels - One hot encoded

Tensorflow and Keras

- I’m using Tensorflow version 2.2
- Keras API
  - Sequential Models
  - Functional API’s

Layers

- Dense
- Convolutional
- LSTM
- Activation
- BatchNorm
- Dropout
- MaxPooling

Model Architectures

- MLP
- CNN
- LSTM
- Convolutional LSTM

Tensorboard

- Monitor different runs
- Tune Hyperparameters
- Logs training process
- Plots

Checkpoints

- Can restore training process
- Save models after every epoch
- Save only the best models

Classification vs Regression

Two main Changes

- Output layer Activation function
  - Softmax - Classification
  - ReLu - Regression
- Loss Function
  - Cross Entropy - Classification
  - MSE, MAE - Regression

Tensorflow Lite

- For Android apps
- Post Training Quantization:
  - https://www.tensorflow.org/lite/performance/post_training_quantization
  - Model size decreases by 4x
- Limited support for LSTM
  - https://github.com/tensorflow/tensorflow/blob/master/tensorflow/lite/experimental/examples/lstm/TensorFlowLite_LSTM_Keras_Tutorial.ipynb
  - https://www.tensorflow.org/lite/convert/rnn

Key Takeaways

- Identify if you are doing a classification or regression task
- Use only CNNs for Real Time processing
- Use Batchnorm and Dropout layers
- Optimizer : Adam