Ahmedabad University
School of Engineering and Applied Science
Ingenium – 2019

Two day workshop
On

Machine Learning using Python (Tensorflow and Keras)
[19 – 20 January, 2019]

1. Deep learning with Python:

There are two top numerical platforms for developing deep learning models, they are Theano developed by the University of Montreal and TensorFlow developed at Google. Both were developed for use in Python and both can be leveraged by the super simple to use Keras library. Keras wraps the numerical computing complexity of Theano and TensorFlow providing a concise API that we will use to develop our own neural network and deep learning models.

Theano is an open source project released under the BSD license and was developed by the LISA (now MILA) group at the University of Montreal, Quebec, Canada. At its heart Theano is a compiler for mathematical expressions in Python. It knows how to take your structures and turn them into very efficient code that uses NumPy, efficient native libraries like BLAS and native code to run as fast as possible on CPUs or GPUs. [Ref: https://en.wikipedia.org/wiki/Theano_(software)]

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team. [Ref: https://en.wikipedia.org/wiki/TensorFlow]

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) and its primary author and maintainer is François Chollet, a Google engineer. [Ref: https://en.wikipedia.org/wiki/Keras]

PyTorch is an open source machine learning library for Python, based on Torch, used for applications such as natural language processing. It is primarily developed by Facebook's artificial-intelligence research group and Uber's "Pyro" software for probabilistic programming is built on it. PyTorch is a python package that provides two high-level features: (i) Tensor computation (like numpy) with strong GPU acceleration, (ii) Deep Neural Networks built on a tape-based autodiff system. [Ref: https://en.wikipedia.org/wiki/PyTorch, https://pytorch.org/about/]

2. TensorFlow:

- Data in the form of Tensor
- Tensor flowing through computational graph

Two steps:
- Building computational graph
- Executing computational graph

2.1 Constant:
- For only constant values
- No external inputs are accepted
- Graphs cannot be modified

Script: 01.tf_constant.py
- Create node, graph
- Create session
- Run the session
- Close the session

Script: 02.tf_const_multiply.py
- Multiplication of two constants

Script: 03.tf_graph.py
- Visualization of computation graph
- Uses tensorboard
- Command: tensorboard –logdir = "graph"

2.2 Placeholder:

- Accepts the external values
- Cannot update the node value

Script: 04.tf_placeholder.py
- Give any external values

2.3 Variables:

- Accepts external value and update it

Script: 05.tf_variable.py
- Output: Loss with respect to fix values of 'w' and 'b'

Exercise – 1:
Change values of 'w' and 'b' to w = -1 and b = 1. Observe the loss value and compare with the previous value. Interpret the result for given linear model, input (x) and actual output (y).

Exercise – 2:
Implement Gradient Descent algorithm for automatically update of parameters 'w' and 'b' using TensorFlow variables. (Update Script: 05.tf_variable.py).

3. Multilayer Perceptron with Keras for multiclass classification problem:

Steps:

- Load data
- Data cleaning and Preprocessing, if required
- Define model
- Compile model
- Fit model
- Evaluate model
- Predict the class for test data for visualization, if required

MNIST Database:

MNIST (Modified National Institute of Standards and Technology) dataset is created for handwritten digit classification problem. Dataset is developed by Yann LeCun, Corinna Cortes and Christopher Burges.

MNIST Dataset

Total images	70,000
Training	60,000
Testing	10,000
Resolution	28 x 28
Classes	10 (0 – 9 digits)

Script: 06.mnist_dataset.py

- Download the MNIST dataset (first time only).
- Visualizing digit character.

Script: 07.mnist_mlp.py

Exercise – 3:
Implement the multilayer perceptron for MNIST dataset for different activation function, loss function, optimizer, epochs and batch size. Analyse the results obtained.

4. Convolutional Neural Network (CNN) with Keras and TensorFlow for multiclass classification problem:

Script: 08.mnist_cnn.py

- Observe the output and compare the result with multilayer perceptron output.

Exercise – 4:
Implement the CNN for large network compare to network in Script: 08.mnist_cnn.py. Analyse the results. Also change the various parameters and continue the analysis. Steps to create model are as follows:

1. Convolutional layer with 30 feature maps of size 5 × 5.
2. Pooling layer taking the max over 2 × 2 patches.
3. Convolutional layer with 15 feature maps of size 3 × 3.
4. Pooling layer taking the max over 2 × 2 patches.
5. Dropout layer with a probability of 20%.
6. Flatten layer.
7. Fully connected layer with 128 neurons and rectifier activation.
8. Fully connected layer with 50 neurons and rectifier activation.
9. Output layer.

Exercise – 5:
Implement the CNN example with TensorFlow only. (For, script: 08 and exercise – 4 both).

TFLearn:

TFlearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it. [Ref: http://tflearn.org/]

(Install TFLearn: pip3 install tflearn)

Exercise – 6:
Implement the CNN example with TFLearn. (For, script: 08 and exercise – 4 both).

Hope “Learning” was “Deep”