

# CSA 250 Deep Learning

## Assignment 1

---

### Python Dependencies and Requirements

- python (v3.7)
- tensorflow (v2.1)
- Other libraries used: argparse, numpy, os

### Instructions to run

The 'main.py' is the interface to the program. It is programmed to run in two modes - train mode and test mode. The 'main.py', when executed without any arguments, enters into training the deep network. The 'main.py' when executed with the (optional argument) '--test-data <test\_file>' enters into test mode and produces the 'Software1.0.txt' and 'Software2.0.txt'.

- Train mode : python main.py
- Test mode : python main.py --test-data <test\_file>

NB: <test\_file> has to be in the same directory as 'main.py'

### Program details

This program is an implementation of FizzBuzz using traditional methods (Software1.0) and using deep learning (Software2.0). The FizzBuzz is a simple task of classifying numbers into four categories - fizz (numbers divisible 3), buzz (numbers divisible by 5), fizzbuzz (numbers divisible by 15) and other numbers as such. The program has three major components:

- ◆ 'main.py' : This python file acts as a client file for the user to interact
- ◆ software\_1\_0 : This is a python package consisting of files for implementation using traditional modulo arithmetic.
- ◆ software\_2\_0 : This is a python package consisting of files for implementation using modern deep learning.

#### *'main.py'*

The 'main.py' is programmed to run in two modes - train mode and test mode. The 'main.py', when executed without any arguments, enters into training mode and when executed with the optional argument test mode.

#### *'software\_1\_0'*

The software\_1\_0 package consists of files used for implementing FizzBuzz using modulo arithmetic. This package consists of two files.

- fizzbuzz.py
- accuracy.py

'fizzbuzz.py' module has a fizzbuzz() function that takes a filename (<test\_file>) as input. The function then opens the input file, reads it line by line to obtain the number and classifies it into one of the four categories. It then writes the classes to a text file 'Software1.txt' in the same directory as that of 'main.py'.

'accuracy.py' module has an accuracy() function that compares the 'Software1.txt' outputted by 'fizzbuzz.py' with 'test\_output.txt' file present in the same directory as that of 'main.py'. It finally calculates and prints the accuracy of classification.

### **'software\_2\_0'**

The software\_2\_0 package consists of files used for implementing FizzBuzz using deep learning. This package consists of five files.

- generate\_train\_data.py
- model\_parameters.py
- model\_train.py
- model\_test.py
- model\_accuracy.py

'generate\_train\_data' module has three functions of which generate\_train\_data() generates the training data set depending on the arguments passed such as TRAIN\_BEGIN, TRAIN\_END etc. Each number between TRAIN\_BEGIN and TRAIN\_END is converted to a numpy array with entries according to the binary sequence of that number. Also for each number, one hot encoded (according to the class) categorical numpy array is generated. These two arrays for the training data (train\_X and train\_Y).

'model\_parameters.py' has model hyperparameter variables such as epochs, learning rate, batch size etc initialized with values.

'model\_train.py' module has the model\_train() function, which trains the deep network using the training data generated with tensorflow backend. After the model is trained, it is stored in a './model' directory.

'model\_test.py' module has model\_test() function that takes a filename (<test\_file>) as input. The function opens and loads the saved model from './model' directory. It then opens the input file, reads it line by line to obtain the number and classifies it into one of the four categories depending on the output of the network model. It then writes the classes to a text file 'Software2.txt' in the same directory as that of 'main.py'.

'model\_accuracy.py' has a model\_accuracy() function that compares the 'Software2.txt' outputted by 'model\_test.py' with 'test\_output.txt' file present in the same directory as that of 'main.py'. It finally calculates and prints the accuracy of classification.

## **Model architecture**

The model is defined using Keras API within Tensorflow. The architecture has two Dense layers with 512 units each. Relu activation is used for both layers for activating the outputs. A dropout layer is added after these dense layers to regularize and prevent overfitting. The loss function used was 'categorical\_crossentropy', and the optimizer used was 'RMSprop()'. The output layer of the network is a dense layer with four categories, with softmax activation. The model was trained for 50 epochs with batch size of 32.

## **Results**

The FizzBuzz task was implemented using the traditional modulo arithmetic method as well as using the deep learning method. The modulo arithmetic method (Software1.0) produced an accuracy of 100%. On the other hand, the deep learning method (Software2.0) produced an accuracy of 98% on the test data, which is quite good.

The Software1.0 implementation is bound to produce 100% accuracy due to the nature of its logic. The Software2.0 implementation uses a deep network that learns the different characteristics of the four classes from the training data and then uses it to predict the classes for test data. The performance of the network varies with different parameters, different optimizers and different structures. The structure used in this project was obtained by trying many of such combinations.