CSE 574 – Introduction to Machine Learning
Project 1.1 Software 1.0 versus Software 2.0
Report 1.0 – Fizz buzz Problem

Asish Kakumanu
UB Person No: 50288695
asishkak@buffalo.edu

Abstract

The Abstract of the Project is to compare two problem solving approaches to software development: The traditional logic-based approach (Software 1.0) and Machine Learning Based Approach (Software 2.0).

1 Software Development Approaches

As Discussed above, we are comparing two different approaches to solve fizz buzz problem.

1. Logic Based Approach.
2. Machine Learning Based Approach.

1.1 Logic Based Approach

In Logic Based Approach, the code is implemented for the first 100 integers where the pseudocode returns fizz when the number is divisible by 3, buzz when the number is divisible by 5, fizzbuzz when the integer is divisible by 15 and other when found any other number which doesn't satisfies the above conditions. Logic Based Approach provides 100% accuracy. This output is used as a result set with which we validate our output generated by Machine Learning Approach

```
def fizzbuzz(n)
    if n % 3 == 0 and n % 5 == 0:
        return 'fizzbuzz'
    elif n % 3 == 0:
        return 'fizz'
    elif n % 5 == 0:
        return 'buzz'
    else:
        return 'other'
```

1.2 Software 2.0

In this Approach, we can use any one of the Machine Learning Frameworks such as TensorFlow, Keras, Pytorch etc.,

For this Instance, we are currently using ***Keras*** Library for the software development. Keras is an open-source library designed to make the creation of new Deep Learning models easy. This high-level neural network API can run on top of deep learning frameworks like TensorFlow, Microsoft CNTK, etc.

Other Packages we use in the development are Pandas, NumPy etc.,

Pandas : pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series

NumPy : NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Step-By-Step Process :

1. Defining a Model
2. Training the Model using dataset
3. Testing using the Model for desired Output
4. Tuning Hyper Parameters, Activation Functions, Cross Entropy Loss Functions & Optimization technique

1.2.1 Creating Multidimensional Binary Vectors

Input is a Number, Output is a String like Fizz, Buzz or Fizz buzz.

We need to change every *Input Integer* into a *Binary Vector* by using *Bitwise Right Shift*.

dataInstance >> d & 1 for d in range(10)

*Which also represents dataInstance by 2**10*

We need to generate output for Integers between 1 to 100. So, training our Machine Learning Algorithm using Integers between 1 to 100 is cheating in Machine Learning terminology. So, we use Integers between 101 to 1000. So, the Number of Digits is provided as 10. So, the Max Number which we get is 1024 and the training set is less than the Max Number.

1.2.2 Mapping Input to Label

Label and the data are Mapped using the below code implementation.

```
if (labelInstance == "fizzbuzz"):
    processedLabel.append([3])
elif (labelInstance == "fizz"):
    processedLabel.append([1])
```

```
            elif (labelInstance == "buzz"):
                processedLabel.append([2])
        else:
            processedLabel.append([0])
```

1.2.3 Creation of a Model

The term ***Model*** refers to the model artifact that is created by the training process. The training data must contain the correct answer, which is known as a target or *target attribute*. The learning algorithm finds *patterns* in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns. You can use the ML model to get predictions on new data for which you do not know the target

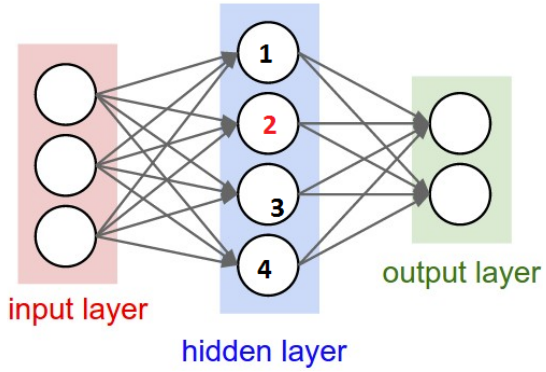
In this Instance we are using Sequential Model rather than using Non-Sequential Model.

Sequential Model: Sequential layers allows us to create model layer by layer. We create hidden layers with weights for the given input to get the desired output.

Non-sequential Model: Non-sequential is a complex network where one layer gives input to any layer such as Siamese and residual networks.

1.2.4 Creation of Layers

A Layer can be an Input layer, hidden layer, Activation etc., A layer has several nodes which are connecting to the nodes in the next layer. Finally, we arrive at the output layer, which is out desired result layer.



1.2.5 Activation Layers

Activation Layers decide whether a neuron should be activated or not. Whether the information that the neuron is receiving is relevant for the given information or should it be ignored.

$$Y = \text{Activation}(\Sigma(weight * input) + bias)$$

The activation function is the nonlinear transformation that we do over the input signal. This transformed output is then sent to the next layer of neurons as input.

We can use more than one activation Function in a given Model.

In this Implementation, we are using various Activation Layers:

1. Sigmoid
2. Tanh
3. Relu
4. SoftMax

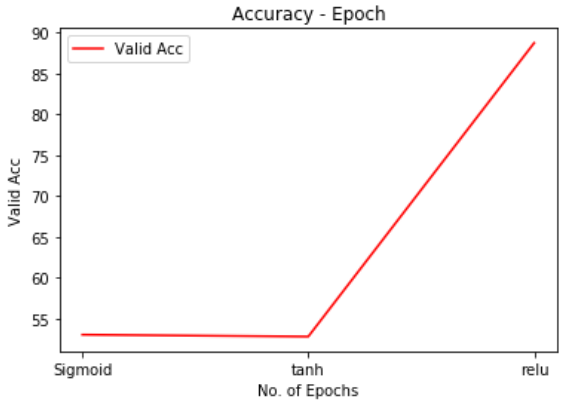
Let us compare each Activation Layer in a given condition where all the hyper parameters, loss Functions and Optimization techniques are same for an Instance.

Hyper Parameters set are

- **Dropout**: 0.2
- **First Layer Nodes**: 256
- **Second Layer Nodes**: 4
- **Num. of Epochs**: 10000

Optimization Technique is ***Rmsprop***

We have used two Activation Functions in this Implementation which are a Pair of two where SoftMax is common



SoftMax Activation:

If we use SoftMax layer as output layer. Exponential function will increase the probability of maximum value of the previous layer compared to other value. The SoftMax function squashes the outputs of each unit to be between 0 and 1, just like a sigmoid function. But it also divides each output such that the total sum of the outputs is equal to 1. Also, summation of all output will be equal to 1.0 always.

1.2.6 Loss Functions

Cross-entropy Loss Functions:

Categorical cross entropy is a loss function. It is one of the three parameters that we use to compile a model. As we have binary vectors as our data. So, we use Categorical Cross entropy. Here each integer value is represented as a binary vector that is all zeros except the index of the number.

1.2.7 Optimizer

There are various optimizers available which are

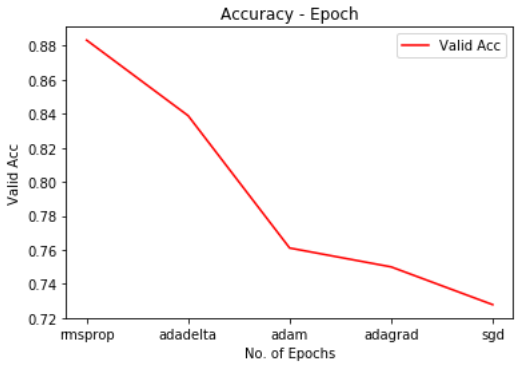
1. SGD
2. Adadelta
3. Adagrad
4. Rmsprop
5. Adam

These are compared in similar conditions

Hyper Parameters set are

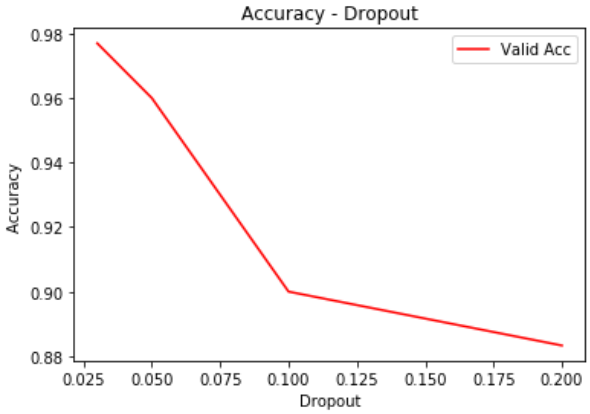
- **Dropout**: 0.2
- **First Layer Nodes**: 256
- **Second Layer Nodes**: 4
- **Num. of Epochs**: 10000

Accuracy calculated for various Activation Functions accordingly



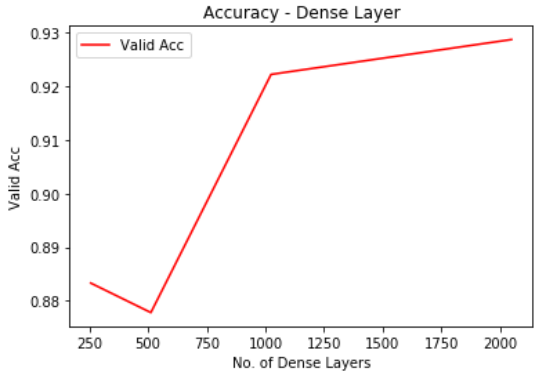
1.2.8 Dropout Layers

Comparison of Dropout and Accuracy is given in the graph below



1.2.9 Dense Layers

Comparison of No. of Dense Layers and Accuracy is given in the graph below



Best set of parameters

Model: *Sequential Model*
Optimization Layer*: Rmsprop*
Activation Function*: Relu & SoftMax*
Hyper Parameters*:*
Epoch Number*: 10000*
Dropout*: 1.0*
First Dense Layers: *2048*
Second Dense Layers: *4*

Validation Accuracy: 0.993
Accuracy: 100%

Conclusion

Thus, we can implement the development in Software 2.0, which we also have implemented in Software 1.0.