INTRODUCTION TO MACHINE LEARNING, CSE 574

PROJECT 1.1

ALGORITHM BASED PROGRAM V/S DEEP LEARNING PROGRAM

- Vineet Suresh Kothari (#50291159)

vineetsu@buffalo.edu

Contents

1. Problem Definiton
2. Structure of the model and parameters
3. Development of the model
4. Results
5. Analysis (subject to different parameters)
6. References

Problem Definition

We consider the task of FizzBuzz. In this task an integer divisible by 3 is printed as Fizz, and integer divisible by 5 is printed as Buzz. An integer divisible by both 3 and 5 is printed as FizzBuzz. If an integer is not divisible by 3 or 5 or 15, it simply prints the input as output (for this last case, the input number can be classified as Other in Software 2.0.

Structure of the model

This deep learning model is sequential implying that it has layered structure of network where each layer contains different nodes with set of weights and input.

The best fit features 5 layers including the input and the output layer. The input layer has 10 nodes, for passing 10 converted binary bits for each input ranging from 1 to 999. Since 999 converted to binary is 0b1111100111, hence 10 input bits. Next three are the dense layers whose no. of nodes are userdefined. The last layer is output layer with 4 nodes, each for each class as 1 or 0.

Development of the model

Import/ Creation of dataset

In this project we create the dataset with help of software 1.0 where the input values ranges from 1 to 999 and output is either Fizz, Buzz, FizzBuzz or Other.

Pre Processing of the dataset

1. For learning and validating the model, we split the dataframe into “Train” and “Test”.
2. Process the input values in terms of binary digits and factorize the output as set of four classes.

Model Creation according to hyper-parameters

1. Initializing model parameters for tuning the model via an interactive real-time form. Following are the parameters:
    - No. of hidden layers. (Name: no_of_layers, Type: Integer, Range: N.A. (feasible for computation))
    - No. of nodes in each hidden layer (Name: nodes_in_layer[i], Type: Integer, Range: N.A. (feasible for computation))
    - Validation split of the dataset(in terms of per unit) (Name: validation_splitt, Type: Float, Range: (0,1))
    - Activation Function: (Name: Activation_function, Select: [tf.nn.relu, tf.nn.relu6, tf.nn.crelu, tf.nn.elu, tf.nn.selu, tf.nn.softplus, tf.nn.softsign, tf.nn.dropout, tf.nn.bias_add, tf.sigmoid, tf.tanh]) [2] [6]
    - Loss Function (Name: loss_function, Select: "categorical_crossentropy", "sparse_categorical_crossentropy", "binary_crossentropy","kullback_leibler_divergence","poisson","cosine_proximity")
    - Optimizer : (Name: optimizer_used, Select: "SGD", "RMSprop", "Adagrad", "Adadelta", "Adam", " Adamax", "Nadam", "TFOptimizer") [1]
2. Initializing Sequential() model in Keras. [1]

Sequential model creates a sequence of layers in the neural network. This also allows us to modify each layer differenly interms of layer type (Dense, Flatten, Reshape) , number of nodes, activation function and other factors. [5]

3. Append layers with number of nodes and activation function to the model using model.add()
4. Add output layer with number of output nodes with softmax activation function.
5. Compile the model by giving the loss function and metrics for comparison.

Model training, testing and validating.

The model is then trained by using model.fit() function with a validation split of 20%. After training the predicted ouput is calculated on the test input. [1]

Results

Accuracy, Loss and Time taken

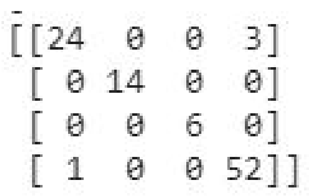
The model results an accuracy of 95.56 %, loss of 0.14 for validation set of split 20%.

It takes 185 seconds to train with 5000 epochs.

Confusion Matrix.

Classification percentges of individual classes out of 100 test samples:

1. Fizz: 24/27 = 88.8%
2. Buzz: 14/14 = 100%
3. FizzBuss 6/6 = 100%
4. Other: 52/58 =89.65%



```
[[24  0  0  3]
 [ 0 14  0  0]
 [ 0  0  6  0]
 [ 1  0  0 52]]
```

Configuration for the best fit

Best fit implies, models resulting best optimal balance of the metrics. For the above results, the model is configured with following parameters:

No. of hidden Layers	3
Nodes in each layer	[64,256,128]
Loss function	Crelu
Activation function	Kullback Leibler Divergence
Optimizer	SGD
No. of epochs	5000
Validation split	20%

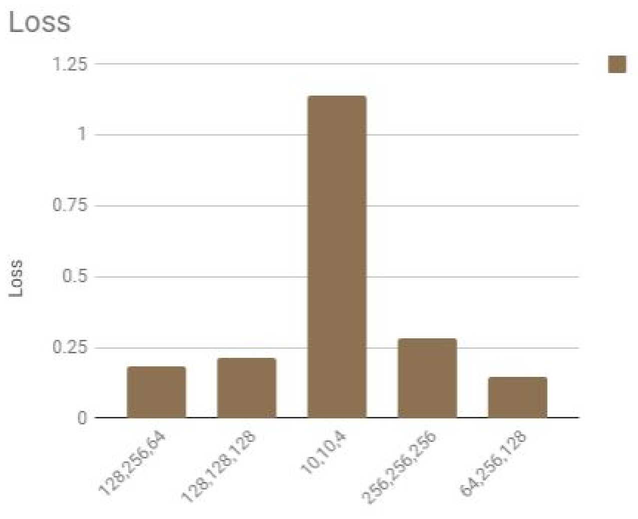
Analysis

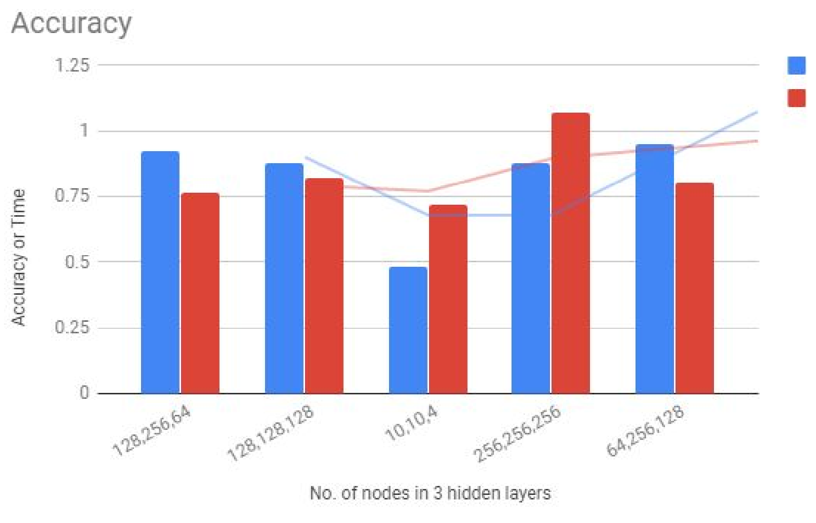
Link to referring the spreadsheet analysis

On analysis of the factors, other parameters are kept constant, and following constant applies to all test cases:

No. of hidden layers =3, Validation split= 0.2, Activation Function: relu, Loss Function: Categorical Crossentropy, Optimizer: SGD

Changing number of nodes in the hidden layer



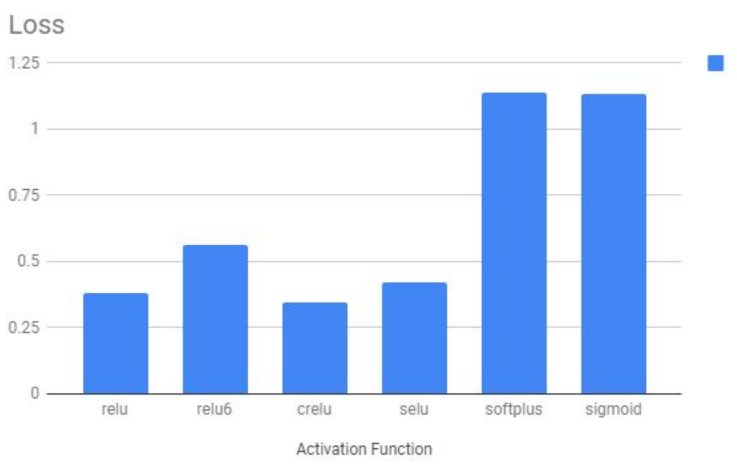


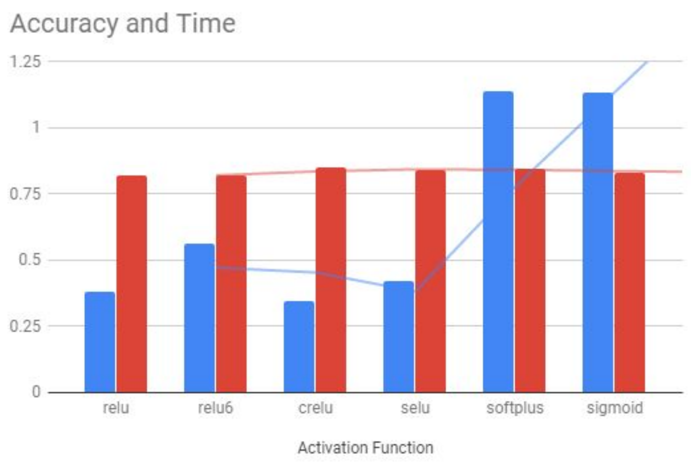
The analysis shows that a neural net with 3 hidden layers with no. of nodes as 64, 256, 128 performs with an accuracy of 95%, loss of 0.14 with time taken, 80.1 seconds.

Layers with nodes as 256, 256, 256 has the most time taken i.e. 1.06 mins. Which concludes, more the no. of nodesin each layer, more the computations, increases the time proportionally.

Changing activation function

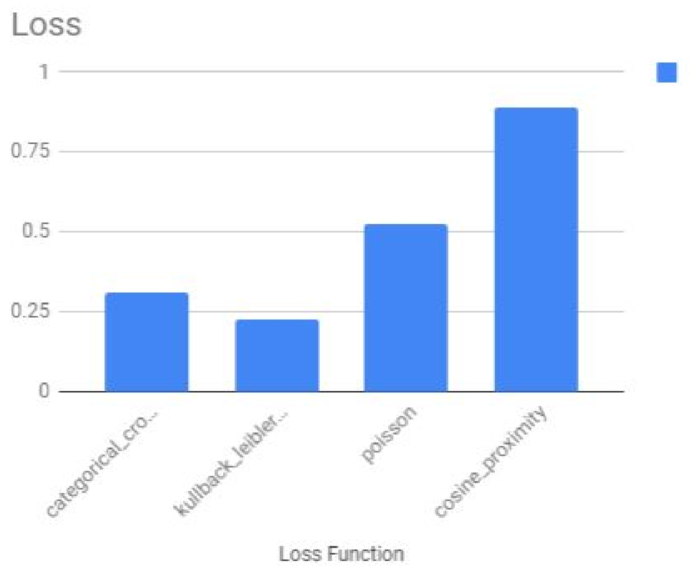
The model results an accuracy of 92 %, loss of 0.34 for validation set using 'Crelu' as an activation function. It takes 85 seconds to train with 2000 epochs.

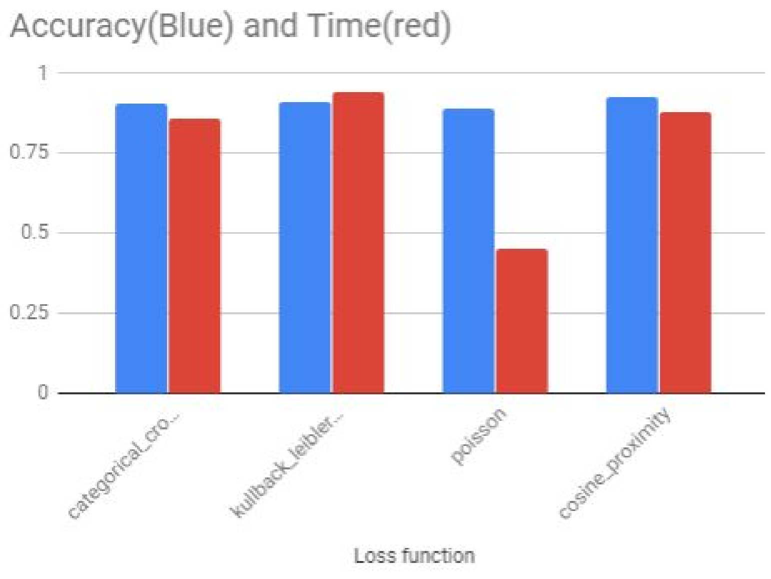




Changing loss function

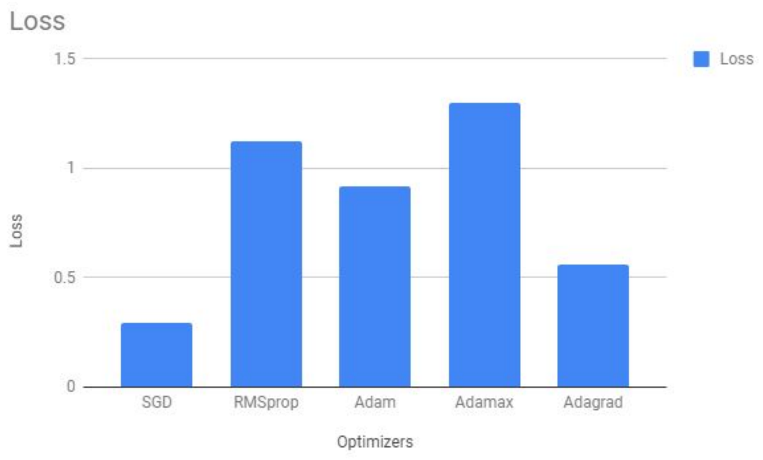
Accuracy of %, Loss of 0.66 for **Kullback Leibler divergence**. Time taken to train with 2000 epochs is 102 seconds.

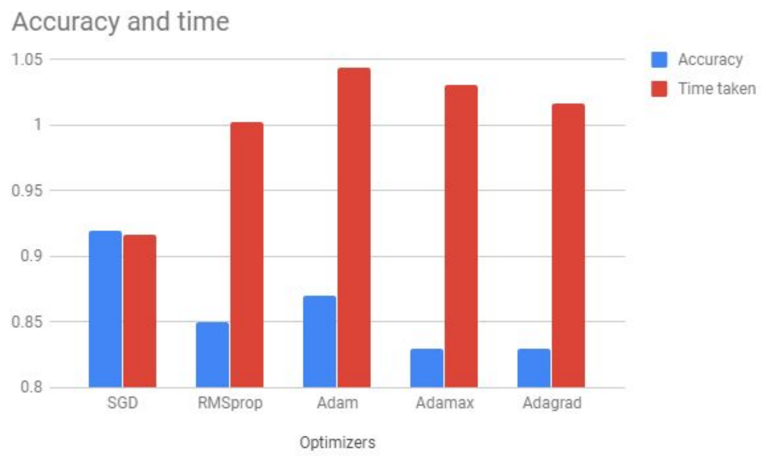




Changing optimizers

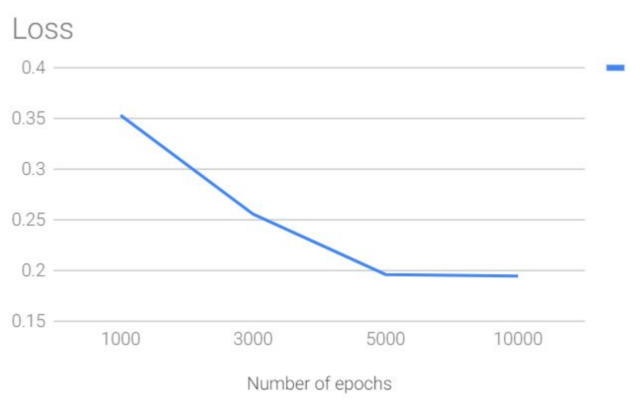
Accuracy of 92%, Loss of 0.29 for **Stochastic Gradient Descent** . Time taken to train with 2000 epochs is 92 seconds

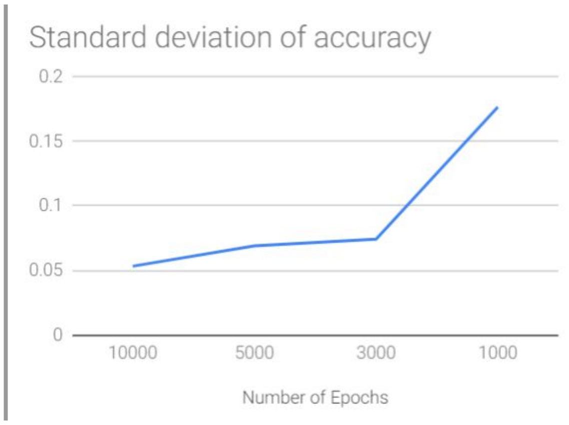




Changing no. of epochs

Epochs may add computations to the model training, but they create the model more stable considering loss and accuracy. Running too many no. of epochs may also lead to overfitting.





Accuracy: 92.40, Loss: 0.19, Time taken: 196.46 seconds, Std. Deviation of accuracy: 0.059 for **5000** Epochs.

Conclusions

1. The final best fit parameters, no. of hidden layers: 3 [64, 256, 128], optimizer: SGD, activation function: crelu, loss function: Kullback Leibler divergence with 5000 epochs gives

Accuracy of 95.56 % with loss of 0.14.

References

[1] https://keras.io, "Keras Documentation"
[2https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0, "Understanding Activation Functions in Neural Networks"
[3]https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d
[4]https://colab.research.google.com/notebooks/forms.ipynb,"Forms in Google colaboratory"
[5]https://www.cs.toronto.edu/~hinton/csc2515/notes/lec9timeseries.pdf, "Sequential models"
[6]https://www.tensorflow.org/api_guides/python/nn, "Activation functions in Neural network, tensorflow"