Soumitra Alate

Department of Computer Science

University at Buffalo

Buffalo, NY 14260

smalate@buffalo.edu

Project 1.1: Software 1.0 Versus Software 2.0

In Software 1.0 the task of FizzBuzz is implemented in Python using the if-else statement and modulo arithmetic. In Software 2.0 the same task is implemented using the machine learning approach.

1. Libraries used in project:

Keras: It is a Python deep learning library. It is a high level neural network api written in python and capable of running on top of TensorFlow.

Pandas: It is a software library written for the python programming language for data manipulation and analysis.

Matplotlib: Matplotlib is a python library used to create 2D graphs and plots by using python scripts.

Numpy: It is a library for multidimensional arrays and matrices. It also contains mathematical functions to operate on the arrays.

TensorFlow: It is a open source software library used for machine learning applications like neural network.

2. Terms used in project:

1) **Dense Layer:** It is a fully connected neural network layer in which each input node is connected to each output node.
2) **Activation Function:** It is a function which converts a input signal of a node in a neural network to an output signal. It converts a linear function to a nonlinear function. If activation function is not used then output will be a linear function and linear function has limited power and does not perform good most of the times. Examples of activation functions are sigmoid, tanh, relu, softmax, leakyrelu.
3) **Loss Function**: It's a method of evaluating how well your algorithm models your dataset. If your predictions are totally off, your loss function will output a higher number. If they're pretty good, it'll output a lower one.(i.e they calculate the difference between output and target variable) Following is the last layer and activation function combination

Problem Type	**Last Layer Activation**	**Loss Function**	**Example**
Binary Multi Label Classification	Sigmoid	Binary_crossentropy	Dog vs Cat/ 1 vs 0
Multi Class, Single label classification	Softmax	Categorical_crossentropy	4 label (Fizz,buzz,fizzbuzz, other)

4) **DropOut:** Number of nodes to be eliminated from hidden layer to check the performance is known as dropout. Dropout is used to prevent overfitting.
5) **Overfitting:** When the model doesn't generalize well from training data to unseen data is called as overfitting. Eg: If a model is trained to recognize image of cat and if its accuracy is good then

model is perfectly trained. However , if you give an image of dog ,then the model will classify as cat which is an impact of overfitting.

6) **Optimizer:** Optimizers calculate the gradient i.e. the partial derivative of loss function with respect to weights, and the weights are modified in the opposite direction of the calculated gradient. This cycle is repeated until we reach the minima of loss function. Examples of optimizers are rmsprop, adam, SGD, Adagrad, Adadelta.
7) **Validation Set:** A set of examples used to tune the parameters of a classifier, for example to choose the number of hidden units in a neural network.
8) **Epochs:** One epoch is when an entire dataset is passed forward and backward through neural network only once. In every iterations the weights get updated.
9) **Batch Size**: Total number of training examples present in single batch is called as batch size. The entire set of input cannot be passed to neural network at once so the data is divided into batches.
10) **Early Stopping**: Early stopping is a technique for controlling overfitting in machine learning models, especially neural networks, by stopping training before the weights have converged.

3. Working:

In Software 1.0 we are creating a simple logic based fizzbuzz implementation in python using if-else and modulo operator. In Software 2.0 a machine learning based fizz-buzz is implemented using keras. In 2.0 approach initially we need to process the data in which we are converting a decimal number into 10 bit binary because using decimal numbers directly will be considered as single feature i.e single input neuron which is not effective to train the neural network. The neural network is effective in high dimensional data so it is converted into 10 bit binary (to represent 1-1000 , 10 bit binary is required). The input neurons are multiplied by weight and added with the bias and then fed to the hidden layer which consist of 256 neurons. In the first dense layer ‘relu’ activation function is used which converts the

linearity into non-linearity because nonlinear functions are capable of solving complex computations. The output of first dense layer is then fed to second dense layer consisting of 4 nodes in which ‘softmax’ activation function is used which predicts the label from the hot-vector(i.e the bucket with labels fizz, buzz, fizzbuzz, other). The model learns to predict the label by a feedback process called as backpropagation. This involves comparing the output a network produces with the output it was meant to produce, and difference is used to modify the weights. This process is performed using a optimizer. The feedback process causes the model to learn, reducing the difference between intended and predicted output.

4. Model Calculation:

```
model = get_model()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 256)               2816
_________________________________________________________________
activation_12 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_14 (Dense)             (None, 4)                 1028
_________________________________________________________________
activation_13 (Activation)   (None, 4)                 0
=================================================================
Total params: 3,844
Trainable params: 3,844
Non-trainable params: 0
```

Calculations:

We are using 256 nodes in first dense layer and for these nodes the value of bias is 256 and there are 10 nodes in input layer. For the second dense layer there are 4 nodes and for these nodes the bias is 4 and the input nodes are 256.

Number of Parameters for dense layer 1: 256*10 + 256 = 2816

Number of Parameters for dense layer 2: 4*256 + 4 = 1028

Total Parameters = 2816+1028 = 3844

5. Network Settings:

Figure	First_Dense_Layer_Nodes	Drop_out	Accuracy
Fig 1	512	0.1	94
Fig 2	128	03	70

Table 1 :*Keeping all other parameters unchanged*

Figure	Activation Function	Optimizer	Accuracy
Fig 3	Tanh	Adam	53
Fig 4	Leakyrelu	SGD	94

Table 2: *Keeping all other parameters unchanged*

Figure	Number of Epochs	Batch Size	Accuracy
Fig 5	20000	128	87
Fig 6	40000	64	90

Table 3: *Keeping all other parameters unchanged*

Accuracy and Loss Graph:

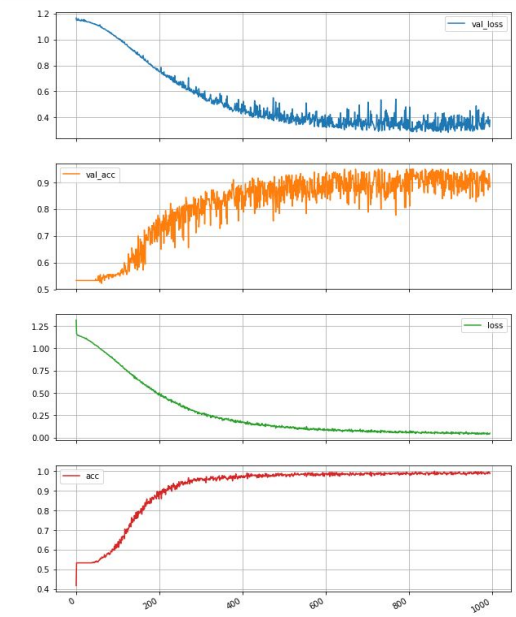

Fig 1

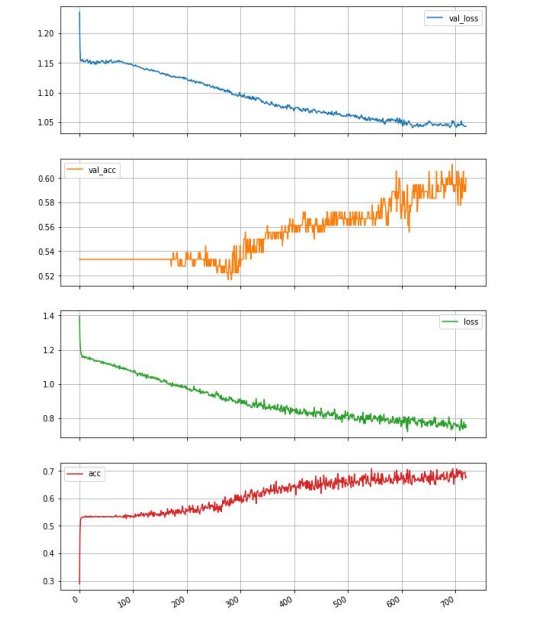

Fig 2

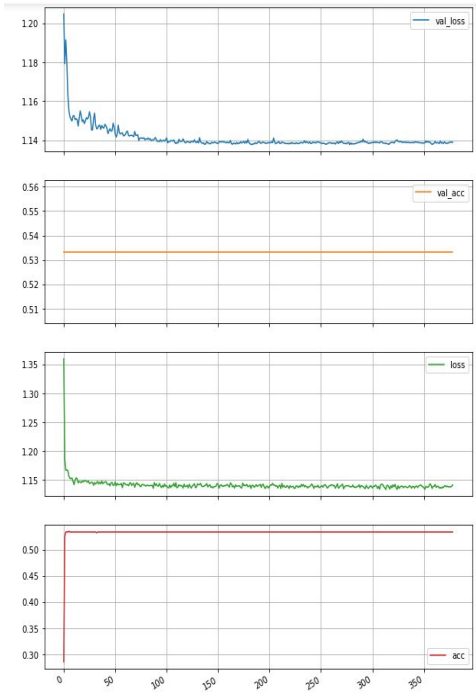

Fig 3

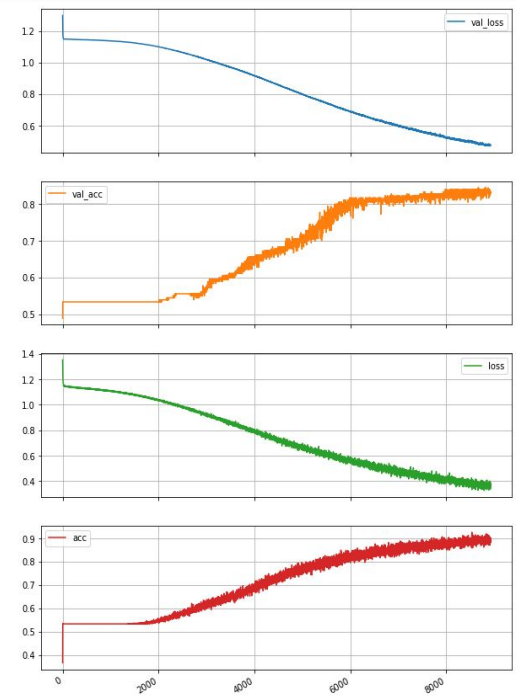

Fig 4

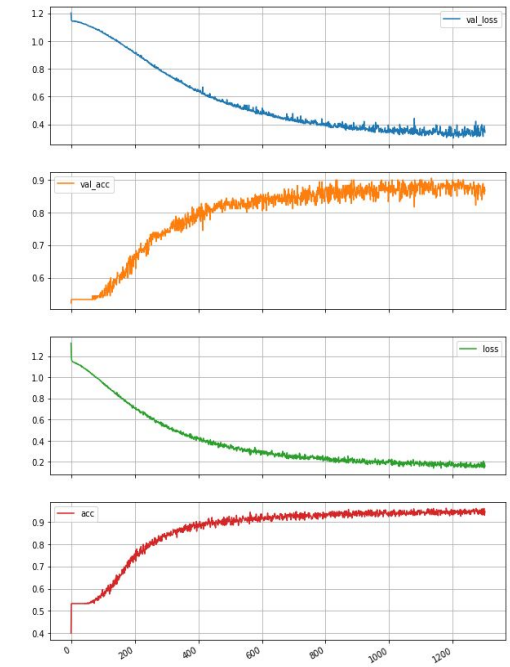

Fig 5

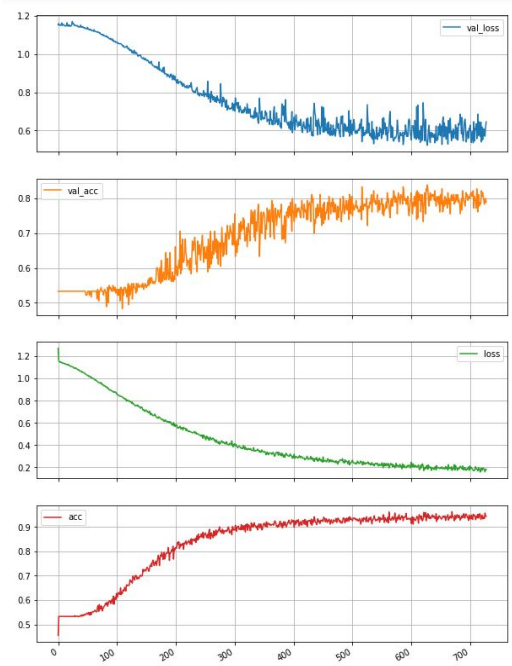

Fig 6

6. Works Cited

1. https://keras.io/
2. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0
3. https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c
4. https://machinelearningmastery.com
5. https://www.explainthatstuff.com/introduction-to-neural-networks.html