Project 1.1: Software 1.0 Versus Software 2.0
Introduction to Machine Learning

Yash Narendra Saraf
UB Person No. 50290453
MS Computer Science and Engineering
School of Engineering and Applied Science
ysaraf@buffalo.edu

Abstract

The project has been used to compare the characterstics and features of software 1.0 and software 2.0. It uses a simple problem of fizz buzz to study various hyper-parameters associated with neural networks and understand the keras and tensor-flow libraries.

1 Software 1.0 and Software 2.0

The terms were coined by professor Andrej Karpathy which compared the conventional ways of solving a problem to the machine leaaring approach. He said "Software 2.0 will become increasingly prevalent in any domain where repeated evaluation is possible and cheap, and where the algorithm itself is difficult to design explicitly."[1] in one of his articles which shows the importance of software 2.0 in the coming time.

2 Fizz Buzz Problem

The fizz buzz is a simple problem where,

- If number is divisible by 3, then print "fizz"
- If number is divisible by 5, then print "buzz"
- If number is divisible both by 3 and 5 , then print "fizzbuzz"
- If number is not divisible both by 3 and 5 , then print "others"

3 Software 1.0

Software 1.0 is a simple logical construct created using if-else loop using the above defined scenario. Since the dataset has been created using a particular set of equations mentioned above. We get perfect results on the dataset too.

4 Software 2.0

Software 2.0 is a method of creating a mathematical model based on traing using datapoints and testing on some different data. Since we need to classify each and every term into either of the four buckets i.e. fizz, buzz, fizzbuzz, or others. We can say that it is a classification problem.

Table 1: Model Description

Layer	Number of nodes	Weights associated or application
Input Layer	10 nodes	None
dense_1 (Dense)	256 nodes	(10+1)*256=2816
activation_1 (Activation)	256 nodes	Relu applied to all dense_1 outputs
dropout_1 (Dropout)	256 nodes	Some percentage of nodes nullified to 0
dense_2 (Dense)	4 nodes	(256+1)*4=1028
activation_2 (Activation)	4 nodes	softmax applied to find probablity

4.1 Neural Network Model

I have referred to the keras code where we have used a sequential model i.e. every layer connected to the next one via weights. Here a particular example for all parameters has been used to explain the model.

The input layer consists of 10 input nodes for binary input of any number from 1 to 1000. Then the inputs has been passed to the dense layer 1 or the hidden layer. The number of weights associated between these layers has been shown in the table. The 1 is added to compensate for the bias.

Then the activation function has been applied, in our case a relu function so that the range of values can be constrained to a particular limit.

Then to avoid overfitting a dropout layer has been added so a few neuron outputs are dropped to 0 randomly before transmitting to furthur layers.

Then a dense layer 2 has been added which has 4 nodes i.e. the number of classes. The number of weights associated is shown in Table 1.

The softmax activation is applied to find the probablity of the input being in each class to make the final guess.

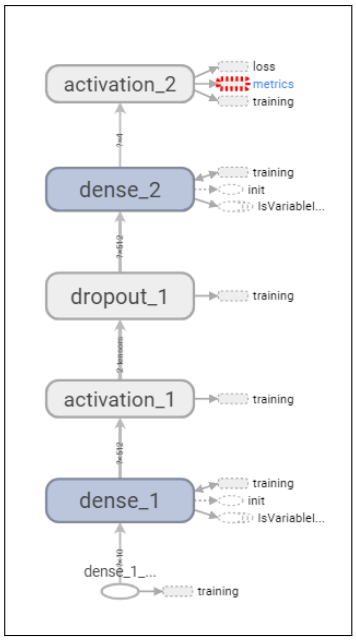

Figure 1: Model

5 Experimentations

Various hyperparameters like Number of hidden nodes, type of optimizer, type of activation at the hidden layer, batch size, dropout and validation data split have been varied and the response has been noted. Used the keras documentation to study different hyperparameters.[2]

5.1 Number of hidden nodes-"first_dense_layer_nodes"

The number of nodes has been between 64 to 1024. It is observed that for lower values the neural network is not able to model the given pattern and it is reflected in the testing accuracy. But as the number of nodes is made sufficiently high i.e. 1024 the model starts to accurately fit the given problem giving a good value for testing accuracy.

5.2 Dropout-"dropout"

The dropout has been added after the dense hidden layer and activation, so it basically nullifies some of the nodes before passing them to the dense layer 2. This is done to ensure the data does'nt

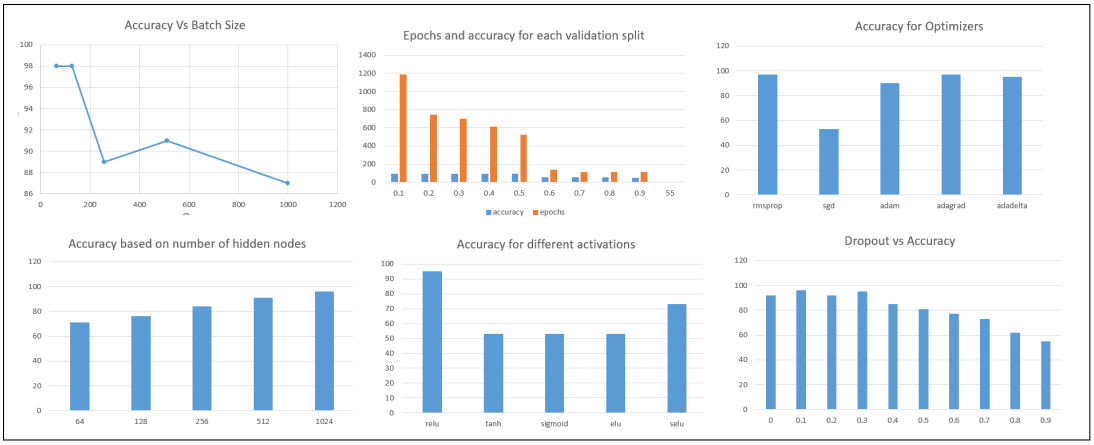

Figure 2: Summary - a-Accuracy plot for various batch sizes, b-Accuracy plot and epochs plot for various validation splits, c-Accuracy plot for various Optimizers, d-Accuracy plot for various number of Hidden node , e-Accuracy plot for various activation functions, f-Accuracy plot for various dropout ratios

start to overfir the training data. This can be verified by simultanously checking the performance of the validation set. Thee dropout has been varried between 0.0 to 0.9 with the intervals of 0.1. The dropout has to be carefully decided as the droput 0.0 means no nde from the hidden layer has been dropped. Also higher values of droput means a lot of significant features may get neglected. So over here we have observed a good accuracy for 0.1 and 0.2 dropout.

5.3 Type of Activation Function-"activation"

Different activation functions for the hidden layer has been used i.e. relu, elu, selu, sigmoid, and tanh and the testing accuracy for each of them has been observed. For all the activation functions except the relu we have noticed poor performance of the model. All the activations except relu has some positive multiplicative value attached to it for the negative input whereas the relu directly nullifies any negative input.

5.4 Type of Optimizer-"optimizer"

Except for Stochastic Gradient Descent all other optimizers used i.e. adam, adagrad, adadelta, and rmsprop has significantly good performance. As these are adaptive learning rate methoda the learing rate is also adjusted based loss function and various other parameters. So using these gives us a significantly better performance than SGD.

5.5 Batch Size-"mode_batch_size"

Batch Size defines the set of datapoints fed to the neural network at one time. So the updates to the weights are performed after each batch rather than each datapoint. This is generally used to make the computations faster. But there is a limit to the batch size which is defined by the amount of data which can be held by the memory of the system. So using a very large batch size may also damage the accuracy as the weights are not being updated frequently also having very low batch size may increase the computation time of the model. So appropriate batch size should be used. In the experimentation the batch size has been varied from 64 to 1000. The best performance has been observed for least batch size but the amount of time to reach the result was also that high.

5.6 Validation Data Split-"validation_data_split"

Validation refers to the amount of data seperated from the training data set to test it during the training phase. This is done so that the model is being tested on the different data than it is being

trained on during training. The training accuracy is always going to increase as we are minimizing the training loss function. But after a point the model will have a better generalization of the given datapoints and the validation accuracy will start to decrease as the model is now starting to overfit the traing dataset. This is the proper time to stop the training cycle. There is a method known as cross validation which shuffles the validation set for each epoch so that the data can be used at its maximum capablities.

It is also clear that as the validation is increased to an unreasonably high values the number of epochs after which it settles has become less, this is because we have set the patience parameter as 100 i.e. the number of epochs to observe the validation error to increase. The validation error is going to increase as the model has not been trained in the first few epochs as the amount of training data has been drastically reduced.

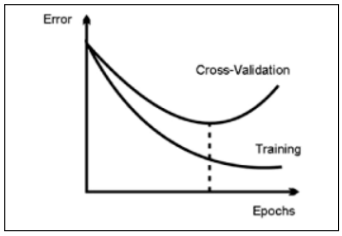

Figure 3: Validation error and Training error [3]

5.7 Output Layer Activation-"Activation"

This layer has been added to find the probablity of each class so that the prediction can be made. Softmax takes the dense layer 2 output and finds out the output probablity for each bucket. So the one with the maximum probablity the is the model's solution to the input datapoint.

5.8 Cross Entropy

The cross entropy is used while training to find the deviation of the obtained output probablities to the actual class. This method is used for classification problems to find out the error function so that it can be minimized.

Conclusion

In this project we have been introduced to the concept of Software-1.0 and how it works. A simple single layer sequential neural network has been studied for a simple fizz-buzz problem by varying all the different hyperparameters. After varying all the hyperparameters the best result has been observed for the following adjusted values of hyperparameters.

INPUT_SIZE = 10	EARLY STOPPING NUM_EPOCHS = 615
DROP_OUT = 0.1	MODEL_BATCH_SIZE = 128
FIRST_DENSE_LAYER_NODES = 1024	OPTIMIZER = `rmsprop`
SECOND_DENSE_LAYER_NODES = 4	EARLY_PATIENCE = 100
VALIDATION_DATA_SPLIT = 0.2	Activation Function: `relu`

The graph for accuracy measurement is as follows:

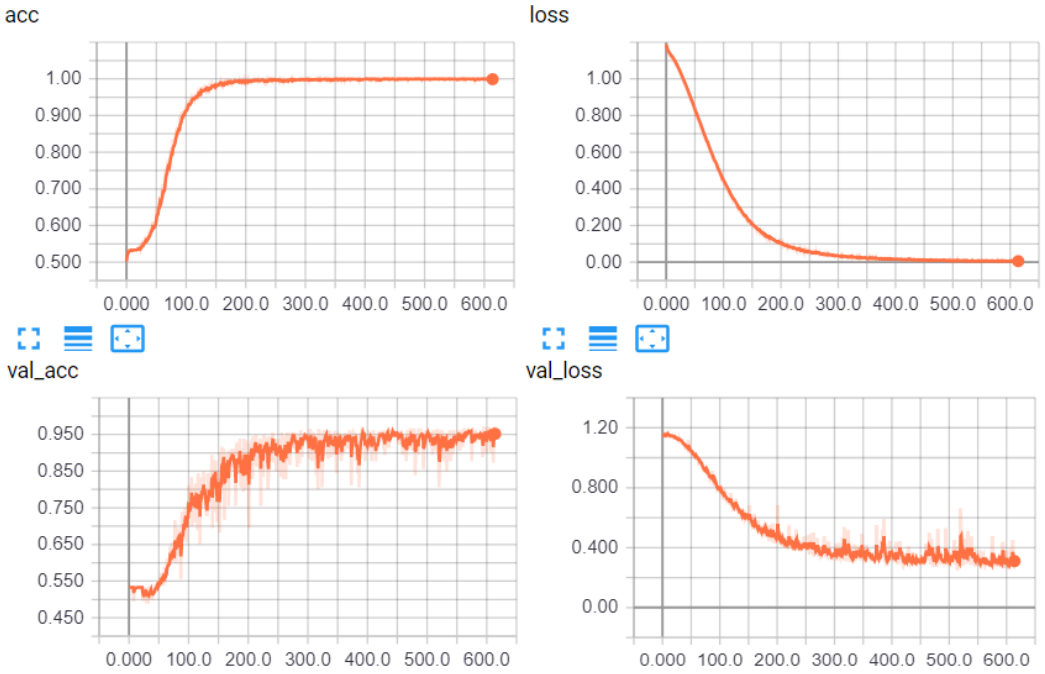

Figure 4: Ideal case scenario observed while varying hyperparameters-Accuracy-98 %

References

[1] Andrey Karpathy, *Software 2.0. medium.com* 2017

[2] Keras Documentation, *https://keras.io/* 2017

[3] Google Image Search *Training Vs Validation error*.