CSE 574 – INTRODUCTION TO MACHINE LEARNING – PROJECT 1.1

Submitted By: Keerthana Baskaran

UB# 50288944

Given Project:

The given project was a Fizz Buzz problem implemented using Sequential model with one input layer, one hidden layer and one output layer. The activation function used for the first hidden layer is ReLU and the activation function used for the second hidden layer (output layer) is softmax. The optimizer used in this model is rmsprop, the loss function is determined by categorical cross entropy and the metrics is based on the accuracy of the model. 10000 epochs are used with 128 as batch size. For regularization, early stopping limit is for 100 epochs and the dropout rate is 0.2. The accuracy of the model is 84% (min) to 91%(max).

Optimizing the given Project:

1. I have used the same sequential model with one input layer, two hidden layers and with one output layer. The input layer has 10 neurons as there are 10 inputs (when we convert the value to binary) the intermediate_layer_1 has 230 neurons and the intermediate_layer_2 has 100 neurons. The output layer has 4 neurons as there is only 4 possibility in this categorical problem.

2. For the first and second hidden layer, I have implemented Tanh activation and for the last layer (output layer) I have used Sigmoid activation function.

3. For regularization, I have used the dropout rate as 0.2 (It is always recommended to use the dropout rate from 20 to 50%) and early stopping limit as 100 epochs.

4. The optimizer that I used here is the Adam. Loss is determined by categorical cross entropy and metrics is measured by accuracy.

5. For the same number of epochs and batch size, the model has reached an accuracy of 99%

Conceptual explanation for the above changes:

1. A model is a set of neural network layers. The simpler one is the Sequential model provided by Keras. The more complex one is Keras functional API (which is used for non -sequential)

   (i) At first, we instantiate the model
   (ii) Add layers to the model
   (iii) Compile the model with the loss function, optimizer and the metrics
   (iv) Fit the model using data
   (v) At last we will evaluate the model with test data

   There is no solid way to determine the number of hidden layers and the number of hidden neurons. Few books and articles have asked to keep it based on the size of input and output, or (no. of input + no. of output) * 2/3 or depending on the training set and so on. But it is recommended to start from one hidden layer and minimum neuron in that layer and then increase based on the performance and accuracy that we get.

   I have used 2 hidden layers in the optimized model.

2. There are few common activation functions like Sigmoid, Tanh, ReLU, Softmax, Leaky Relu, Maxout. Sigmoid has range from 0 and 1 and it has convergence issue. So, Tanh is used which ranges from -1 to 1 and easy to converge to 0. Sigmoid can be used for probability problems and it will be efficient because range is 0 to 1. Tanh is used for classification between two class problem. Both sigmoid and Tanh has the vanishing gradient issue and ReLU overcomes it. But the limitation of ReLU is that it can be used only within the hidden layers. Hence for output layers we can use Softmax. The Softmax can be used highly for multiclass classification problem like Fizz Buzz. Another issue with ReLU is that sometimes the gradient will be fragile and will lead to dead neurons. In this case the weights will be updated. So Leaky ReLU is used which will keep the updates active.

3. Sometimes the hypothesis curve is too much fitting (overfitting) the data that the model does not work accurately, so to stop this we use regularization. We use dropout rate and early stopping as a part of regularization. The dropout rate can be roughly between 20% to 50% where 50% will give maximum amount of regularization. Earlystopping is the number of epochs which has no improvement for which the model will run before it stops. Here we set the earlystopping as 100 epochs.

4. Optimizers are used to minimize the loss function (error function). There are many optimizers available however Adam is considered most effective. Adam is different to the stochastic gradient descent which maintains single learning rate and does not change during the training. Adam is combination of Adaptive Gradient Algorithm and root mean square propagation. Adagrad is specific to individual features where some weights in dataset will have different learning rate than others. Adagrad can be used when lot of inputs are missing. RMSProp is recommended for recurrent neural networks.

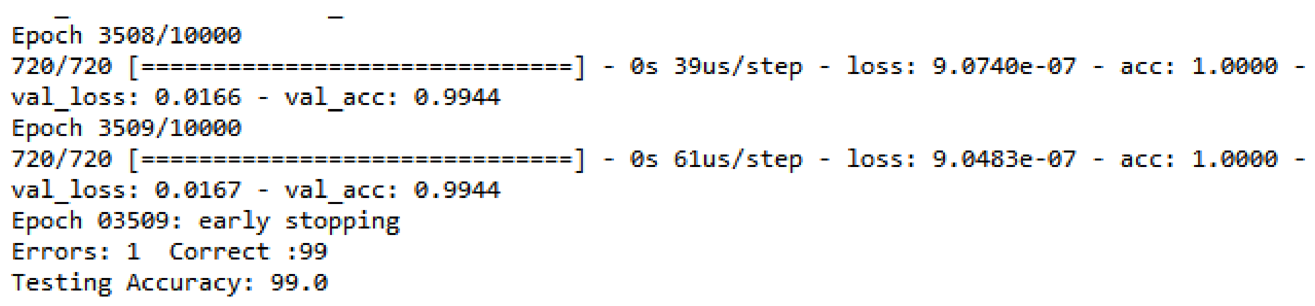
Optimized model:

Network Setting:

model=sequential model, Input size=10, dropout=0.2, num_of_hidden_layer=2, First_dense_layer=230, Second_dense_layer=100, Third_dense_layer=4, dropout=0.2, Activation function: Tanh for first two layers and Sigmoid for last layer, optimizer: Adam, model_batch_size=128, tensorboard_batch_size=32, early_patience=100, metrics=accuracy.

Accuracy achieved: 99%

Output of the model:



```
Epoch 3508/10000
720/720 [==============================] - 0s 39us/step - loss: 9.0740e-07 - acc: 1.0000 -
val_loss: 0.0166 - val_acc: 0.9944
Epoch 3509/10000
720/720 [==============================] - 0s 61us/step - loss: 9.0483e-07 - acc: 1.0000 -
val_loss: 0.0167 - val_acc: 0.9944
Epoch 03509: early stopping
Errors: 1  Correct :99
Testing Accuracy: 99.0
```

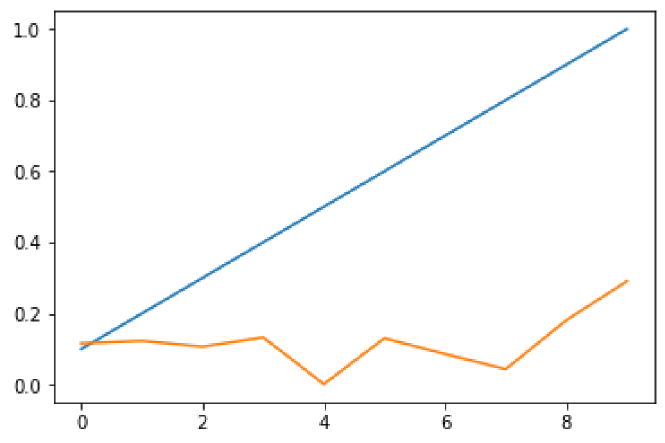


Fig1.1 Cross entropy vs dropout rate(x,y)

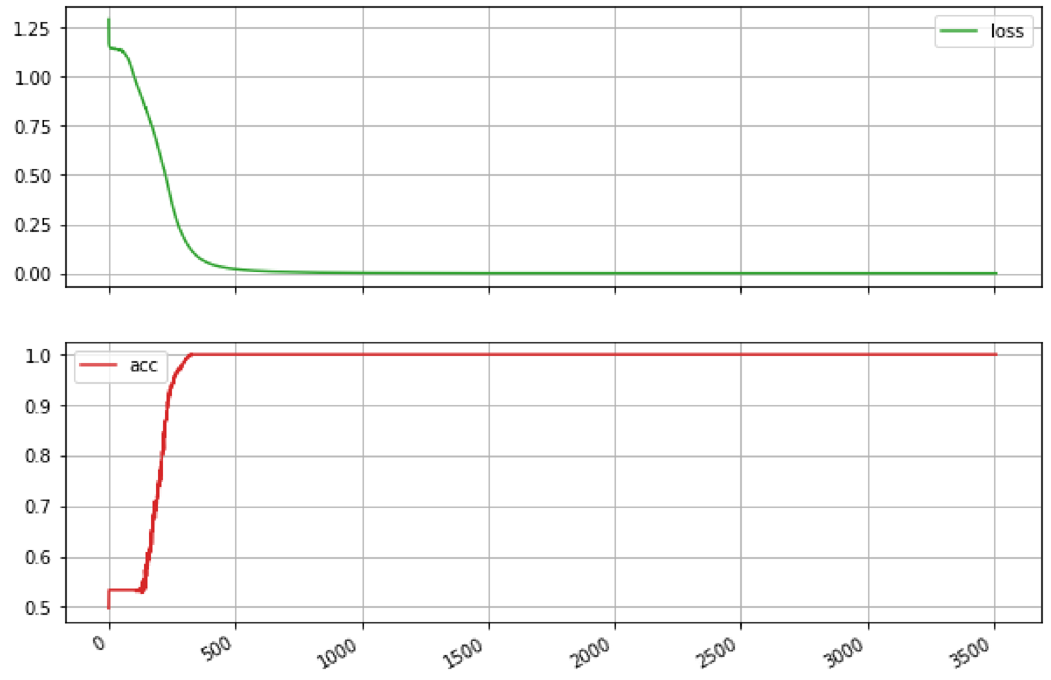


Fig1.2 Model loss and accuracy graph

Results observed:

By using two hidden layers with enough neuron and by using the optimizer as Adam the model could give a higher accuracy even if the activation function used is not highly accurate.

Other experiments:

Standard condition (hyperparameters from the given project):

Input size=10; dropout=0.2; first_dense_layer=256; second_dense_layer=100; validation_data_split=0.2; epoch_count=10000; model_batch_size=128; tensorboard_batch_size=32; early_patience=100, activation function=Relu, softmax; optimizer=rmsprop; metrics=accuracy.

Observations by changing the hyperparameters:

The one mentioned in 'changes' are the values changed for each experiment. Other hyperparameter is maintained as in standard condition for these experiments.

1. **Changes:** Epoch=100
   Observation: The accuracy is around 53%. This is because the model learns less with a smaller number of epochs. If the epoch value is given too high, then the model might overfit and fail to learn the data. So, a moderate epoch value with early stopping will help the model get high accuracy.

2. **Changes:** Epoch=100, Validation_data_split=0.5, dropout=0.5
   Observation: the accuracy is 64%. This is because when the dropout ratio is set to 50%, more regularization occurs. Since the training and validation set Is equally split, the model will learn more efficiently, and the weights will get updated. However the Epoch is too less for the model to become accurate.

3. **Changes:** Validation_data_split=0.5, dropout=0.5, first_dense_layer=1 neuron
   Observation: The accuracy reached here is 52% since the model could not learn accurately with one hidden layer having one active neuron. We must start with one hidden layer with one neuron but keep increasing the neurons and layers till the model reaches a good accuracy.

4. **Changes:** Dropout=0.2, first_dense_layer=230, second_dense_layer=100, third_dense_layer=4
   Observation: The accuracy achieved here is 89%. Because of the greater number of neurons, the weights got updated after each epoch and the model has gained higher accuracy.

5. **Changes:** No early stopping, no Dropout
   Observation: Without early stopping the model has learnt with all the epochs and the accuracy achieved is 92%. However, overfitting did not occur with such large epoch value.

6. **Changes:** Epoch=100000, no early stopping, no dropout
   Observation: The Epoch size is increased 10 times more and still the model performs good and accuracy reaches 90% without early stopping and dropout.

7. **Changes:** First_dense_layer=230, Second_dense_layer=100, Third_dense_layer=4, Activation function: Sigmoid for all 3 layers, optimizer: Adam
   Observation: The accuracy is 53%. The same network setup by changing activation function to tanh and sigmoid gave an accuracy of 99% in the optimized model but in this case, it is 53%. This is because the sigmoid function has many disadvantages that it sometimes misguides the model in learning.

8. **Changes:** First_dense_layer=230, Second_dense_layer=100, Third_dense_layer=4, Activation function: Tanh, Tanh, Sigmoid, optimizer: rmsprop
   Observation: The accuracy is 97%. With the same setting by changing optimizer to Adam the model gave an accuracy to 99%. Adam optimizes better than rmsprop.

Experiments using same network setup but changing one hyperparameter:

1. **Network Setup:** Input size=10; dropout=0.2; first_dense_layer=256; second_dense_layer=100; validation_data_split=0.2; epoch_count=10000; model_batch_size=128; tensorboard_batch_size=32; early_patience=100

Activation function in layer 1	Activation function in layer 2	Accuracy
Sigmoid	Tanh	37%
Tanh	Tanh	24%
Tanh	Sigmoid	53%
Sigmoid	Sigmoid	53%
Relu	Softmax	83%
Tanh	Softmax	89%

Table 1.1 Different activation function vs Accuracy

Observation: For the same network setup, using Softmax in last layer gives the higher accuracy irrespective of the activation function used in other hidden layers.

2. **Network setup:** Input size=10; dropout=0.2; validation_data_split=0.2; First_dense_layer=230, Second_dense_layer=100, Third_dense_layer=4, dropout=0.2, Activation function: Tanh, Tanh, Sigmoid

Optimizer	Accuracy
Rmsprop	97%
Adam	99%
Adagrad	90%
Adadelta	86%

Table 1.2 Different Optimizers vs Accuracy

Observation: From the above experiment, it is said that for the same network setting Adam gives higher accuracy than anyother optimizer.

3. **Network Setup:** Input size=10; dropout=0.2; validation_data_split=0.2; epoch_count=10000; model_batch_size=128; tensorboard_batch_size=32; early_patience=100, Activation: Relu, Relu, Relu Softmax, optimizer=adam

First dense layer	Second dense layer	Third Dense layer	Fourth dense layer	Accuracy
5	6	9	4	53%
20	40	70	4	48%
100	100	100	4	74%
140	180	120	4	75%

Table 1.3 Different neurons in each layer vs Accuracy

Observation: As the number of neurons in each layer increases, the accuracy is also increasing for the same network setup. Additionally, the epoch processed is getting reduced as the number increases.

4. **Network Setup:** Input size=10; dropout=0.2; validation_data_split=0.2; epoch_count=10000; model_batch_size=128; tensorboard_batch_size=32; early_patience=100, Activation: Relu for all layers except last layer(softmax) optimizer=adam

Number of hidden layers	Number of neurons in each layer	Accuracy
3	230,250,260,4	86%
4	230, 250, 260, 4	80%
5	230, 250, 260, 230, 4	69%
6	230, 250, 260, 230, 240, 4	48%

Table 1.4 Number of hidden layer vs Accuracy

Observation: It is noticed that for the same network setting, as the number of layers is increasing the accuracy is decreasing. This must be due to the overfitting issue.

5. **Network setup:** Input size=10; dropout=0.2; validation_data_split=0.2; First_dense_layer=230, Second_dense_layer=100, Third_dense_layer=4, dropout=0.2, Activation function: Tanh, Tanh, Sigmoid

Dropout	Cross entropy loss
0.1	0.1162
0.2	0.1233
0.3	0.1067
0.4	0.1329
0.5	0.0013
0.6	0.1305
0.7	0.0859
0.8	0.0435
0.9	0.1812
1.0	0.2909

Table 1.5 Dropout rate vs Cross-entropy loss

Learning from these experiments:

1. Both training and validation accuracy increases as the number of epochs increase. More information is learned in each epoch.
2. Using Adam optimizer increases the training time.
3. Increasing the batch size decreases the training time but reduces the rate of learning.
4. As the number of epochs increase, more information is learned. The training as well as validation accuracy increases and then stabilizes.
5. By adding more hidden layers, training time as well as information learned in each epoch increases.
6. As the number of neurons in each layer increases the accuracy increases.
7. As the number of layers are increasing the accuracy varies (sometime decreases as it goes high).