**Name: Modukuri Sumanth Kumar**

**S/R No.: 17025**

**Tools**

* Tensorflow 2.x
* Keras
* Python 3.x

**Observations**

This is a basic classification problem of FizzBuzz with 4 classes, training on [101, 1000] & testing on [1, 100].

Here when I talk about “**BASE**”, it implies the Number base in which the encoding for the input was done. I wanted to test which BASE works best for this problem, and which

* Model Configurations

**2 Layer Model**

tf.keras.layers.Dense(512, activation='relu'),  
tf.keras.layers.Dropout(0.2),  
tf.keras.layers.Dense(1024, activation='relu'),  
tf.keras.layers.Dropout(0.2),  
tf.keras.layers.Dense(CLASS\_SIZE, activation='softmax')

Result:

|  |  |  |
| --- | --- | --- |
| BASE (encoding) | EPOCHS | ACCURACY in % (on test data) |
| 2 | 100 | 95 |
| 2 | 200 | 97 |
| 2 | 300 | 97 |
| 3 | 100 | 94 |
| 3 | 200 | 96 |
| 3 | 300 | 96 |

We see that 2 layers with Dropout is performing very good with few hundred EPOCHS. But I wanted to test if it is better to use multiple layers, as in is it helping me get better results than 1 layer model.

**1 Layer Model**

tf.keras.layers.Dense(1024, activation='relu'),   
tf.keras.layers.Dense(CLASS\_SIZE, activation='softmax')

Result:

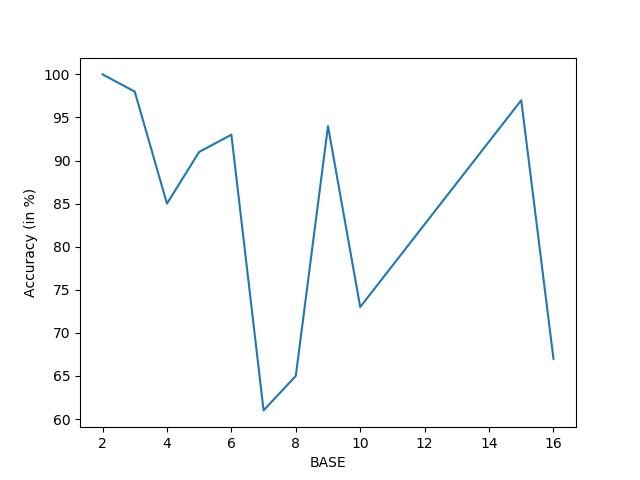
|  |  |  |
| --- | --- | --- |
| BASE (encoding) | EPOCHS | ACCURACY inn % (on test data) |
| 2 | 100 | 97 |
| 2 | 300 | 100 |
| 3 | 100 | 85 |
| 3 | 300 | 96 |

With Increasing EPOCHs we get equivalent if not better results in 1-layer than in the 2-layer Model. So continuing with 1-layer Model.

* BASE finding

For this I ran 1000 EPOCHs for each BASE to get the best one.

Result:



Inferences:

* + BASE = 2 performs the best, gives 100% accuracy on test data
  + All BASE values that are multiples of 3 perform better than others (except 2) because data has more multiples of 3 to label
  + All BASE values that are multiples of 5 perform better than prime numbers (except 2 & 3) because data has more multiples of 5 to label and prime numbers don’t have a correlation with the data
  + Not using DROPOUT helps in this case as it actually learns the pattern of the data and that is what is required to get perfect results
* Optimizer

I observed that amongst “Adam”, “SGD” & “RMSprop”, “RMSprop” worked the best for this data set.

* Loss

As we are solving a classification problem here, it is recommended to use “categorical\_crossentropy”

* Activation

For Hidden layers I observed that “relu” gave better performance over other activation functions like “softmax” & “tanh”

And for Output Layer, the best activation function was “softmax”

Henceforth We can conclude that to get the best results on the test data we can use the following:

* EPOCHS = 1000
* BASE = 2
* Model: 1-Layer No Dropout Model
* Optimizer: “RMSprop”
* Loss: “categorical\_crossentropy”
* Hidden layer activation: ”relu”
* Output Layer activation: “softmax”

model.compile(optimizer='RMSprop',

                loss='categorical\_crossentropy',

                metrics=['accuracy'])

Points:

* Model with Encoding Base = 2 gives better accuracy than Encoding Base = 3
* Adding Dropout in the model gave worse results
* Accuracy with 1000 Epochs is 100 % on test data
* Increasing the number of levels in the model did not achieve 100 % accuracy on the test data