Text Predictions of LSTM RNN Performance Analysis using TensorFlow GPU – Deep Adaptive Learning

[1]T. Tritva Jyothi Kiran, [2]Dr. Dhanraj Varma
[1]Assistant Professor, [2]Professor
Computer Science Department
Dr. A.P.J. Abdul Kalam University, Indore, India.
{tritvajkiran@gmail.com, dhanrajmtech@gmail.com}

ABSTRACT: *A Recurrent Neural Network (RNN) is a type of Artificial Neural Networks (ANN) that is designed to take temporal dimension into consideration by having a memory (internal state) (feedback loop). LSTM networks work better compared to RNN since they overcome vanishing gradient problem. In practice, RNN fail to establish long term dependencies Feed Forward Neural Networks (vanilla networks) that map a fixed size input (such as image) to a fixed size output (classes or probabilities). But a drawback in Feed Forward Networks is that they do not have any time dependency or memory effect. RNN allow us to work with a sequence of vectors: Sequence in inputs, Sequence in outputs, Sequence in both. In this work I have used Text dataset which contains over 1115394 characters, from which sequence of different combinations of unique characters and required selected sentences extracted from text dataset and also make the effect of Dictionary. All the results are very efficient and accurate. The LSTM RNN proved its efficiency of learning 100% on Text Classification and Multi Task Learning in my Performance analysis work.*

KeyWords: *RNN (Recurrent Neural Networks), LSTM (Long Short-Term Memory), Feed Forward Neural Networks (FF-NN), Text Classification, Gradient descent, loss.*

I. INTRODUCTION

This work presenting an amazing type of artificial neural networks known as the recurrent neural networks (RNN). The base reference for RNN is the feed forward artificial network. RNN is a type of artificial neural network that is designed to take important dimensions which means it's going to take time right now into effect as well or into consideration by having a memory or internal state. As shown in figure (1) Feedforward Neural Networks (vanilla networks) that map a fixed size input (such as image) to a fixed size output (classes or probabilities). But a drawback in Feedforward networks is that they do not have any time dependency or memory effect. A RNN is a type of ANN that is designed to take temporal dimension into consideration by having a memory (internal state) (feedback loop) [1][2][3].

RNN takes the output in the previous time step and we feed it back again as an input to the neuron which means that this neuron in the next time step will rely on the input and its previous state now it has some memory effect and that's the beauty of recurrent neural networks (RNN). By using RNN it performs a text for example classification or for text prediction and they are being widely used in tons of applications in natural language processing even in image recognition. And a little bit deeper into how to train them what are the drawbacks of them and how do we overcome these drawbacks using a certain type of recorded neural networks known as long short-term memory or LST networks [2][3].

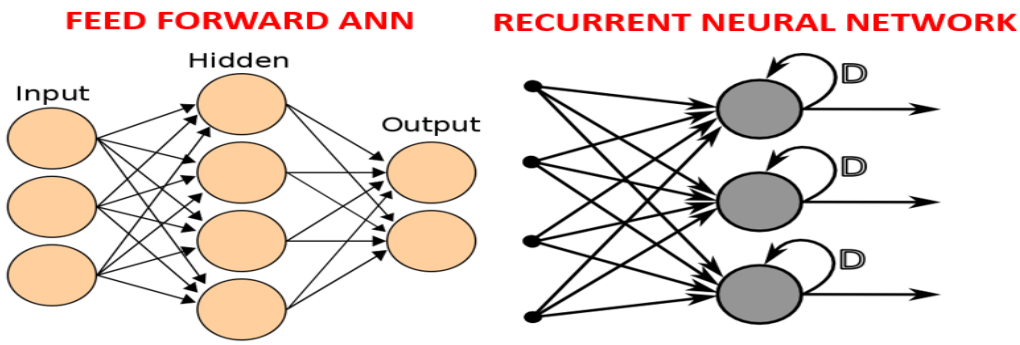

Figure (1): Model of RNN vs FF-ANN

.

A. RNN ARCHITECTURE

As shown in figure (1) and in figure (2), the RNN contains a temporal loop in which the hidden layer not only gives an output but it feeds itself as well. [1][2] An extra dimension is added which is time. RNN can ability to remember what happened in the previous time step so it works great with sequence

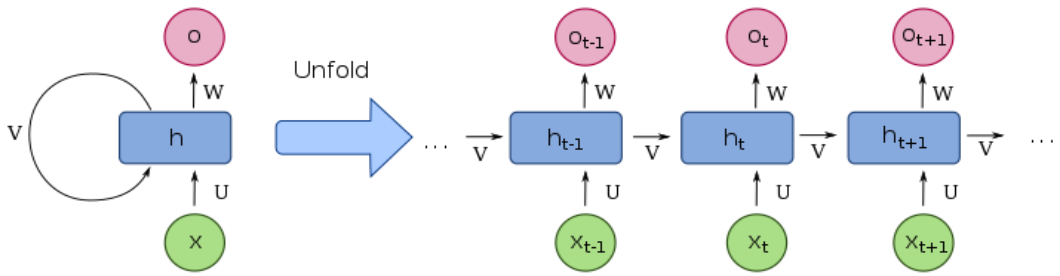

Figure (2): RNN Architecture.

of text. The trained RNNs to engender text character by character and consider the question "how is that even possible?" We found that recorded neural networks is a certain type of artificial neural networks in which we have some portal dimension or time to take time into consideration we started to use a kind of memory effect or internal state by feeding in the output of some neuron and feeding it back as an input to our neuron. The RNN architecture uses important loop in which a hidden layer not only gives an output but it feeds itself as well. Here our input is x that's again the output and it uses different weights (w) here with these are the weights connecting the input to hidden state each. What happened is that internal state or kind of memory effect we're going to feed itself back as an input, and the output here does not only rely on the input x which is basically if you hide the section if you ignore the section that will be a simple feed forward artificial neural networks in which an output relies only on the input. And here what happened in the past and that's where the time dependency comes into play [1][2][3].

So as mentioned before is an extra dimension that has been added to recurrent neural networks which is simply our time and we couldn't neuro networks would be able to recall what happened in the previous time step. So, again the work or it works great with sequence of text for example, language translation. To know like if the subject is a male or female or for example that the subject is like singular or plural there are so many different parameters that you need to kind of form an overall picture. By this kind of network, you can visualize it in a different way that make it a little bit easier for us to visualize.
The element when we unfolded as shown in figure (2) we have time dimension. That's why we have time (T) here [1][2][3].

Because it taking H_{t-1} which is happened in the previous time step multiplying it by V and I will feed it in here to input as well and in this time step XY will be fed as well as an input to future time step H_{t+1} and keep doing that's where the actual output not only rely on input but it will also rely on what happened in previous time step keep doing that over and over again.

Do it in math using an advanced version of recurrent neural networks known as LSTM or a long short-term memory.

B. WHAT MAKES RNN SO SPECIAL

This section covers the important features of recurrent neural networks (RNN) and why are they so special. Feedforward ANNs are so constrained with their fixed number of input and outputs suppose like a CNN will have fixed size image (28x28) and produces a fixed output (class or probabilities). Feedforward ANN have a fixed configuration, i.e.: same number of hidden layers and weights.

Recurrent Neural Networks offer huge advantage over feedforward ANN. RNN allow us to work with a sequence of vectors: Sequence in inputs, Sequence in outputs, Sequence in both.

C. RNN MATH

A RNN accepts an input x and generate an output o.
The output o does not depend on the input x alone, however, it depends on the entire history of the inputs that have been fed to the network in previous time steps. Two equations that govern the RNN are as follows:

INTERNAL STATE UPDATE:

$$h_t = \tanh(X_t * U + h_{t-1} * V)$$

OUTPUT UPDATE:

$$o_t = softmax(W * h_t)$$

To learn an advanced form of recurrent neural networks known as LSTM or a long short-term memory.

D. VANISHING GRADIENT PROBLEM

LSTM networks work much better compared to vanilla RNN since they overcome the vanishing gradient problem. The error has to propagate through all the previous layers resulting in a vanishing gradient. As the gradient goes smaller, the network weights are no longer updated. When we add more layers to the network, the gradients of the loss function slant to zero, making the network hard to train. Each layer depends on the output from the previous layers, the "v" is multiplied several times resulting in vanishing gradient, (ex: 0.1 * 0.1 * 0.1 *…. *0.1 =1e-10). ANN gradients are calculated during backpropagation. In backpropagation, we calculate the derivatives of the network by moving from the outermost layer (close to output) back to the initial layers (close to inputs). The chain rule is used during this calculation in which the derivatives from the final layers are multiplied by the derivatives from early layers. The gradients keep diminishing exponentially and therefore the weights and biases are no longer being updated. each layer depends on the output from the previous layers, the "v" is multiplied several times resulting in vanishing gradient, (ex: 0.1 * 0.1 * 0.1 *.....*0.1 =1e-10)

Current Weight = Old Weight – Learning rate * gradient.

9.09999 = 10.1 – 1 * 0.001

[1][2][3] Gradient descent is an optimization algorithm. Which helps to obtain the optimized network weight and bias values. It is implemented to work as by iteratively trying to minimize the cost function. It works by calculating the gradient of the cost function and affecting in the adverse direction until the local/global minimum is achieved. local/global maximum is achieved when we take the positive of the gradient.

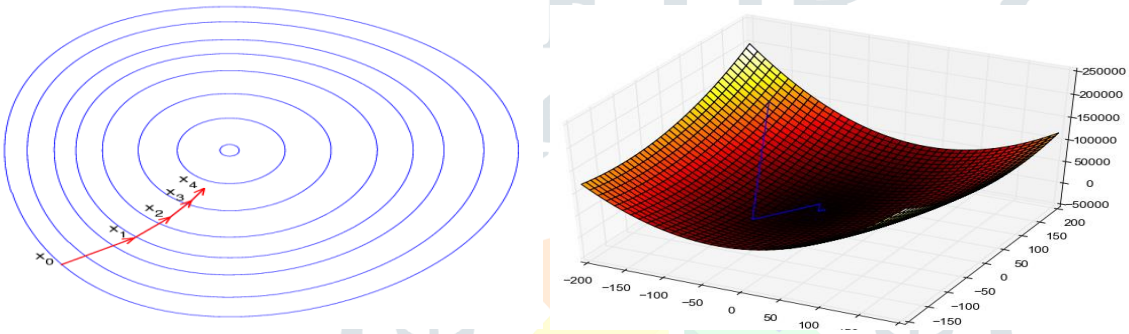

Figure (3): vanishing gradient problem

As shown in figure (3), as more layers are added the gradients of the lost function starts to approach to 0 making the network extremely hard to train. The problem in some of these techniques get stuck in we call it local minimum but we're not using that because we need you know other strategies like regularization and other strategies to avoid being stuck in a local minimum problem. If the positive of the gradient is taken local or global maximum is achieved. That is gradient ascent.

The learning rate is the size of the steps taken. If learning rate increases, the range covered within the search space will increase so we reach global minimum faster. However, we can overshoot the target. For small learning rates, training will take for much longer to succeed in optimized weight values. If learning rate increases, we increase the learning rate the area covered in the search space will increase. So, we might reach global minimum faster, we call it adaptive learning rate.

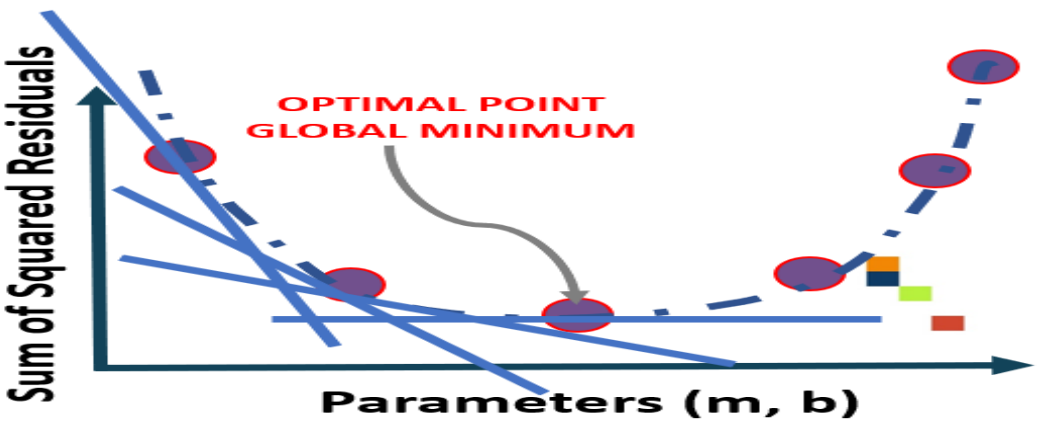

Figure (4): Gradient descent

As per results of figure (4), Gradient descent works as follows:

1. Calculate the derivative (gradient) of the Loss function

2. Pick random values for parameters m, b and substitute
3. Compute the step size (how much are we going to update the parameters?)

$$Step\ size\ =\ Slope\ *\ learning\ rate$$

4. Update the parameters and repeat

E. LONG SHORT-TERM MEMORY (LSTM)

[1][2][3] LSTM networks work better compared to vanilla RNN since they overcome vanishing gradient problem. In practice, RNN fail to establish long term dependencies, we can see it in figure (5).

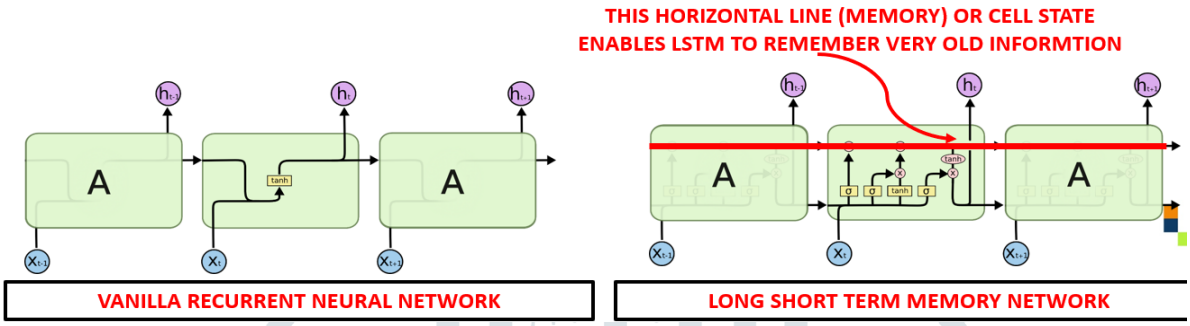

Figure (5): LSTM vs RNN

LSTM networks are the improved advanced version of RNN that are designed to remember long term dependencies by default. LSTM can remember and recall information for a prolonged period of time. Recall that each line represents a full vector. As per Figure (6), LSTM contains gates that can allow or block information from passing by. [3] Gateways entail of a sigmoid neural net layer along with a pointwise multiplication operation. Sigmoid output ranges from 0 to 1.
0 = Don't allow any data to flow.
1 = Allow everything to flow!

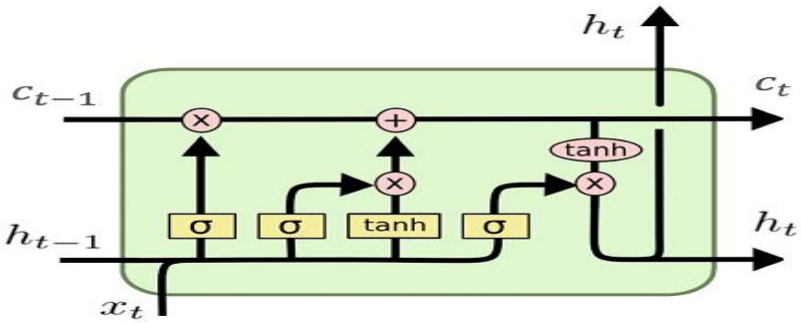

Figure (6): LSTM math

II. IMPLEMENTATION

To map text to numbers and then to create training samples and benches, meant for data preparation, the challenging part is to get the training data first and to make that training data ready to perform the training. And then afterwards we feed in testing dataset to start to generate text based on train model.
Train a long short-term memory which is a type of record known as a network and what's record on your network simply contains loop in which the header layer not solitary bounces an output but it feeds itself as well. From bunch of inputs bunch of outputs develop a relationship between them. Then by adding the time we are adding basically an extra dimension to our equation and then we'll be able to predict let's say anything related to time sequence.

A simple vanilla feed forward artificial network actually uses a little bit more complex architecture and that's why here we're using recurrent neural network for the next generation. This work using basic Text datasets to train our LSTM network to perform predictions and the objective is to train network to predict the text the next character in a sequence of characters. First Import libraries and the load data sets. That's the training data that is to feed in LSTM and network to try predict. I wanted to obtain basically the unique characters I want to know how many unique characters I have here. Take vocab which is simply vocabulary. Now, need to do an encoding, actually need to convert these basial letters into a bunch of numbers. And once we have that then we can take these numbers kind of an encoded version of it at the ticket and then train our LSTM network. And once I train it then I can take these numbers back a kind of decode it back again to
bunch of characters and then can create text or sentence again.

Next step is to map text to numbers. First, we're creating a map from unique characters to indices. The vocab creates a bunch of numbers in here. Create one index character that would be kind of the translation in the other direction because we need both. That's simply convert dataset from characters to bunch of integers. And that's the objective. Now, took the dataset underscores text and converted it to a bunch of numbers This is dictionary. Take every single character convert it to a bunch of numbers. Now, all text training data represented in an integer format. Perform predictions over LSTM network. Use a batch method which can easily convert these individual characters into sequence of the desired size.

To duplicate the sequence, shift it by one character. By doing that then we have created our input and our output data. To shuffle data, it just tunes the order of it and that will improve our generalization capability of our network and specify all parameters forward. Then create an embedding layer. That's how you build our model. Next, create elastic layer simply called TAF that carries the layers that Elysium specify how many layers you're looking for. Now, create a dense network at the end. And that will be our architecture of network. Then we can change the dimensions as you want. Now use an optimizer in this case.

III. RESULTS AND COMPARISIONS

```
Actual Data set Text input:

-------------------------------
ALONSO:
What, all so soon asleep! I wish mine eyes
Would, with themselves, shut up my thoughts: I find
They are inclined to do so.

SEBASTIAN:
Please you, sir,
Do not omit the heavy offer of it:
It seldom visits sorrow; when it doth,
It is a comforter.

ANTONIO:
We two, my lord,
Will guard your person while you take your rest,
And watch your safety.

ALONSO:
Thank you. Wondrous heavy.

SEBASTIAN:
What a strange drowsiness possesses them!

ANTONIO:
It is the quality o' the climate.

SEBASTIAN:
Why
Doth it not then our eyelids sink? I find not
Myself disposed to sleep.

ANTONIO:
Nor I; my spirits are nimble.
They fell together all, as by consent;
They dropp'd, as by a thunder-stroke. What might,
Worthy Sebastian? O, what might? --No more: --
And yet me thinks I see it in thy face,
What thou shouldst be: the occasion speaks thee, and
My strong imagination sees a crown
Dropping upon thy head.

SEBASTIAN:
What, art thou waking?

ANTONIO:
Do you not hear me speak?

SEBASTIAN:
```

```
I do; and surely
It is a sleepy language and thou speak'st
Out of thy sleep. What is it thou didst say?
This is a strange repose, to be asleep
With eyes wide open; standing, speaking, moving,
And yet so fast asleep.

ANTONIO:
Noble Sebastian,
Thou let'st thy fortune sleep--die, rather; wink'st
                            Whiles thou art waking.
```

Total Length of Data Set: 1115394

First Run for unique characters:

---

65 unique characters

Unique Characters OUTPUT

```
"'", ',', '-', '.', '3', ':', ';', '?', 'A', 'B', 'C', 'D',
 'E', 'F', 'G', 'H', 'I', 'J',
 'K', 'L', 'M', 'N', 'O', 'P',
 'Q', 'R', 'S', 'T', 'U', 'V',
 'W', 'X', 'Y', 'Z', 'a', 'b',
 'c', 'd', 'e', 'f', 'g', 'h',
 'i', 'j', 'k', 'l', 'm', 'n',
 'o', 'p', 'q', 'r', 's', 't',
 'u', 'v', 'w', 'x', 'y', 'z'
```

Integers for unique characters OUTPUT

```
"'": 5, ',': 6, '-': 7, '.': 8,
 '3': 9, ':': 10, ';': 11, '?': 12, 'A': 13, 'B': 14, 'C': 15,
 'D': 16, 'E': 17, 'F': 18, 'G': 19, 'H': 20, 'I': 21, 'J': 22,
 'K': 23, 'L': 24, 'M': 25, 'N': 26, 'O': 27, 'P': 28, 'Q': 29,
 'R': 30, 'S': 31, 'T': 32, 'U': 33, 'V': 34, 'W': 35, 'X': 36,
 'Y': 37, 'Z': 38, 'a': 39, 'b': 40, 'c': 41, 'd': 42, 'e': 43,
 'f': 44, 'g': 45, 'h': 46,
 'i': 47, 'j': 48, 'k': 49,
 'l': 50, 'm': 51, 'n': 52,
 'o': 53, 'p': 54, 'q': 55,
 'r': 56, 's': 57, 't': 58,
 'u': 59, 'v': 60, 'w': 61,
 'x': 62, 'y': 63, 'z': 64
```

Array arrangement OUTPUT

```
array(['\n', ' ', '!', '$', '&', "'", ',', '-', '.', '3', ':', ';', '?',
       'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
       'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
       'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
       'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'],
                                    dtype='<U1')
```

Selected values OUTPUT

```
array[18, 47, 56, ..., 45,  8,  0])

'First Citizen' ---- characters mapped to int ---- > [18 47 56 57 58 1 15 47 58
47 64 43 52]

Total Characters = 1115394

Selected characters = 11153
```

OUTPUT

A l l : R e s o l v e d .

R e s o l v e d .

F I r s t C I t I z e n :

F I r s t , y o u

Batch and vocab size OUTPUT

```
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
(64, 100, 65) # (batch_size, sequence_length, vocab_size)
```

Parameters used

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (64, None, 256)           16640
_________________________________________________________________
gru (GRU)                    (64, None, 1024)          3938304
_________________________________________________________________
dense (Dense)                (64, None, 65)            66625
=================================================================
Total params: 4,021,569
Trainable params: 4,021,569
Non-trainable params: 0
```

Prediction's output

```
array([43, 27,  1, 23, 13, 60, 57, 45, 34, 58, 23,  1, 19, 34, 13, 12, 56,
       42, 48,  8, 12, 33, 23, 51, 63, 58,  6, 38,  7, 16, 31, 14,  9, 20,
       18, 11, 47, 14,  4, 12, 32, 34, 25,  3, 61,  2,  3, 36, 28, 56, 34,
       54, 36, 38, 10, 56,  7, 26, 61,  3,  2, 50, 18,  2, 60, 47, 44, 26,
       55, 28, 23, 58, 27, 59,  6,  8, 26, 14, 54, 60, 50, 48, 32, 23, 46,
             49, 39, 58, 13,  0, 19, 32, 56, 50, 58, 53, 31, 30, 23, 11])
```

Input:

```
 "ous queen of heaven, that kiss\nI carried from thee, dear; and my true
lip\nHath virgin'd it e'er sinc"
```

Next Char Predictions:

```
'eO KAvsgVtK GVA?rdj.?UKmyt,Z-DSB3HF;iB&?TVM$w!$XPrVpXZ:r-
Nw$!lF!vifNqPKtOu,.NBpvljTKhkatA\nGTrltoSRK;'
```

Prediction shape: (64, 100, 65)
(batch_size, sequence_length, vocab_size)
scalar_loss: 4.174824

Each Epoch Loss output

```
Epoch 1/10
172/172 [==============================] - 27s 137ms/step - loss: 3.2144
Epoch 2/10
172/172 [==============================] - 25s 136ms/step - loss: 2.0480
Epoch 3/10
172/172 [==============================] - 25s 136ms/step - loss: 1.7410
Epoch 4/10
172/172 [==============================] - 25s 136ms/step - loss: 1.5700
Epoch 5/10
172/172 [==============================] - 25s 136ms/step - loss: 1.4684
Epoch 6/10
172/172 [==============================] - 25s 136ms/step - loss: 1.3992
Epoch 7/10
172/172 [==============================] - 25s 136ms/step - loss: 1.3540
Epoch 8/10
172/172 [==============================] - 25s 136ms/step - loss: 1.3105
Epoch 9/10
172/172 [==============================] - 25s 136ms/step - loss: 1.2726
Epoch 10/10
    172/172 [==============================] - 25s 136ms/step - loss: 1.2347
```

After Loss decreased Parameters OUTPUT

./training_checkpoints/ckpt_10

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (1, None, 256)            16640
_________________________________________________________________
gru_1 (GRU)                  (1, None, 1024)           3938304
_________________________________________________________________
dense_1 (Dense)              (1, None, 65)             66625
=================================================================
Total params: 4,021,569
Trainable params: 4,021,569
Non-trainable params: 0
```

Final Prediction OUTPUT

```
ROMEO: my lordship.
DRKEN a husband speak.

Citizens:
Very figniful troop.

RICHARD:
Bads villain, destains ught!
Therefore, I must be consumes I do past their offices;
```

```
Her humbly in your expectance is but the glory of Hermione of love,
Suffixite their awful heart,
To been in parent and myself;
She was to court or, bring from Dadlassing.

LEONTES:
What talks up on warrow of wing,
To the year to run his horse that dreams have lef
The heaven the late was much about thee; and bring you than all the heads,
The adver yielded, or else, To your bones
Ky cursed sthere of it already.


HENRY BELINGBROKE:
My Lord, love Every dark obedience,
With any stroke, shall make them on:
'Tis arouts with such a fool,
Have your infection of a minstre whole hands, he entreaty that fearful bed
And all the name puts for our couss, by his haste you will do,
Ere the rest in these fool is bustle,
With smales it as my cold fully.

BENVOLIO:
Then let it be possible;
            And that at swellish-brother, and well, when ye no monsh
```

IV. CONCLUSION

All the results are very efficient and accurate. The LSTM RNN proved its efficiency of learning 100% on Text Classification and Multi Task Learning in my Performance analysis work. A Recurrent Neural Network (RNN) is a type of Artificial Neural Networks (ANN) that is designed to take temporal dimension into consideration by having a memory (internal state) (feedback loop). LSTM networks work better compared to RNN since they overcome vanishing gradient problem. In practice, RNN fail to establish long term dependencies Feed Forward Neural Networks (vanilla networks) that map a fixed size input (such as image) to a fixed size output (classes or probabilities). But a drawback in Feed Forward Networks is that they do not have any time dependency or memory effect. RNN allow us to work with a sequence of vectors: Sequence in inputs, Sequence in outputs, Sequence in both. In this work I have used Text dataset which contains over 1115394 characters, from which sequence of different combinations of unique characters and required selected sentences extracted from text dataset and also make the effect of Dictionary. All the results are very efficient and accurate.

Future Work: This work done using GPU, I recommend the extension using TPU.

V. REFERENCES

[1] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee, Recent Advances in Recurrent Neural Networks, arXiv, 2018.

[2] Ilya Sutskever, James Martens, Geoffrey Hinton, Generating Text with Recurrent Neural Networks, International Conference on Machine Learning, Bellevue, WA, USA, 2011.

[3] Alex Sherstinsky, Fundamentals of Recurrent Neural Network (RNN) and Long- STM (LSTM) Network, arXiv, 2021.

Author:

I am Mrs. T. Tritva Jyothi Kiran with 10 years of work as Assistant Professor in Computer Science Department. Previously I have completed AICTE funding Project on IEEE802.11e in JNTUH and published in IEEE. I have been awarded Two times for the National Award for Excellence "Adarsh Vidya Saraswathi Rastriya Puraskar" from Glacier Global Management in 2020. And Received "Women Researcher" Award from 9th international conference organized by VDGOOD Professional Association. Extant I am working in the research domain Deep Learning using TensorFlow and Swarm Intelligence. You can find my Lectures during COVID in my Blog is tritvajyothikiran.blogspot.com and in Tritva Jyothi YouTube Channel. This work is my part of the Phd research work under the supervisor Dr. Dhanraj Varma, Principal & Research Dean in Dr. A.P.J. ABDUL KALAM UNIVERSITY,INDORE, MADHYA PRADESH, INDIA.