iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** IV **Month of publication:** April 2020

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue IV Apr 2020- Available at www.ijraset.com

Analysis of Machine Learning Algorithms for Predicting Confirmed Cases of Covid-19 in India

Abhishek Porwal[1], Shruti Jain[2]
[1]*B.Tech Scholar,* [2]*Assistant Professor, Dept. of Computer Science and Engineering, Indore Institute of Science and Technology, Indore M.P., India*

Abstract: ***There are many models and algorithms are available in Machine Learning for generating prediction which can be applied to wide range of cases. Good predictions can help us prepare better for the future. This paper is a practice to implement those application on the CVID-19 data set to predict confirmed cases of COVID-19 in India. The data set is available open on Government of India dashboard but we have used dataset from Kaggle and tried to see which Machine Learning Models will give good predictions. In this, we had analyzed Simple Artificial Neural Network, Deep Neural Network and Recurrent Neural Network. We had run the same experiment with different setups where we enhance number of hidden layers to see how model performance will change. At the end we had given a comparison of all the models for their performance.***
Keywords: Deep learning, hidden layers, COVID-19, Recurrent Neural Network, Deep Neural Network, Kaggle.

I. INTRODUCTION

This Paper is about comparing Machine Learning algorithms to predict confirmed cases of COVID-19 in India. So First of all it is important to understand about COVID-19 and the Machine Learning Algorithms that we are going to use.

A.COVID-19

In India, the outbreak of novel corona virus has caused approximately 5500+ people infected and 100+ deaths. At the time of working on the prediction, confirmed cases were just 749. To control the spread of disease, screening large numbers of suspected cases for appropriate quarantine and treatment measures is a priority.

B.Machine Learning Models

Following Machine Learning Models can be used data for time-series analysis. As the data, we are working on is temporal, so these models have been selected for analysis. [1] gives us the motivation for such analysis.

1) *Artificial Neural Network*: Artificial Neural Network are computing system inspired by the Human Brain. ANNs functions computing and are similar to biological neural network in basic characteristics. There are many inputs to neurons, each input is given a relative weight which affects the processing of input. Several types of ANNs have been developed in past years, but can be broadly classified on the way of learning process i.e. 'Supervised learning' and 'Unsupervised learning'. The ANNs have been applied to many fields including automotive, banking, financial, medical, speech and many more. The work on ANNs have only increased since 1990's and is progressing rapidly.[2]

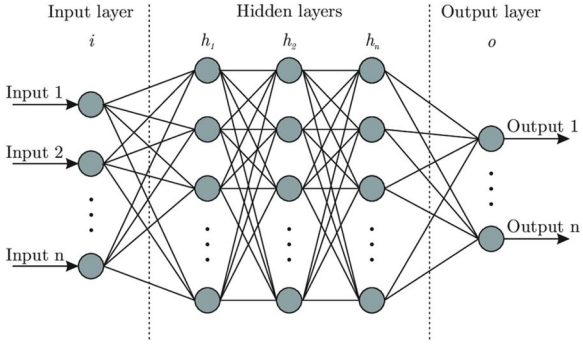


Fig: 1 Architecture Of Deep Neural Network

2) *Deep learning for Prediction:* There are many machine learning algorithms which have been working well from long time. Deep learning algorithms are good and have been around for many decades but highlighted in recent years only. This is because of scaling in data all thanks to digitalization and development in hardware like GPU devoted to computational power. Large scale neural net have astonishing performance with increase in data scaling. Even small neural net performs well with huge data comparing to traditional algorithms. [3]represents the approach using Deep Neural Network.

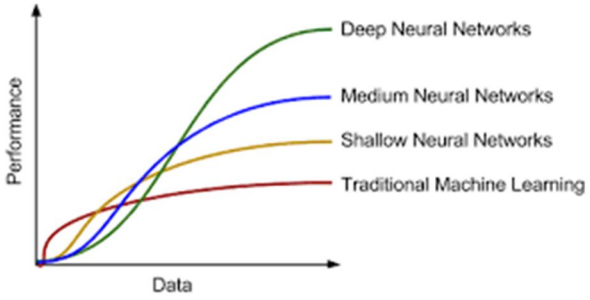


Fig: 2 Performance Of Data Scaling

3) *Deep Neural Network:* A Deep Neural Network [4] (DNN) is an Artificial Neural Network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship.

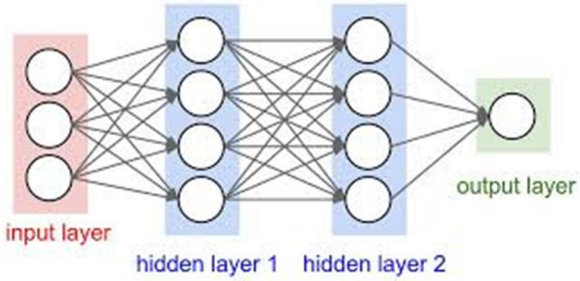


Fig. 3 ARCHITECTURE OF DEEP NEURAL NETWORK

4) *Recurrent Neural Network:* A Recurrent Neural Network[4] (RNN) is a class of Artificial Neural Network where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. There are more advanced Recurrent Neural Network which are known as LSTM and GRU. This paper does not include LSTM and GRU but can be applied on the same.

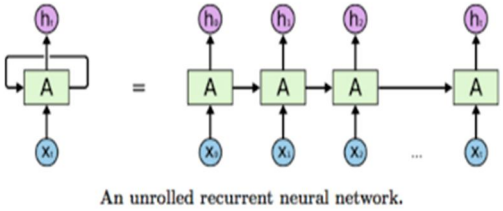


Fig. 4 Architecture of Recurrent Neural Network

II. EXPERIMENTAL SETUP

A. *Dataset*

Kaggle dataset is one of the best place to find reliable and quality datasets. In the predictions used for this paper we used dataset from covid-19 (week 3) competition on kaggle. The dataset consist of columns namely Id, Province_State, Country_Region, Date, ConfirmedCases, and Fatalities. The dataset contain data from entire world with 22950 entries. But the paper is only focused on the India so we pre-processed the data to be used for training the models. We don't need the entire set so we will be keeping data relavent to India and remove all the unnecessary columns too.

B. Pre-Processing

Pre-processing of the data to feed the neural networks has been done to get maximum performance from deep learning models. From going through data, it has been observed that the data have dates as a column, which needs to be replaced by serial number therefore a new column named “days” has been created. As we were working on Indian data only so we had removed all the rows which does not belongs to the "Country_Region" as India. We didn’t needed the Province_State column and Id column therefore it is also useless for our experimental setup so we had dropped them from the dataset. Having very less data makes our task much more difficult as the data is highly influenced from past data. This makes it difficult to predict due to involvement of seasoning, trends and noise. Finally we had only ‘days’ as “time” and ‘ConfirmedCases’ as “series” from the original dataset. The reduced dataset had only 75 rows. We used windowed dataset with window_size equal to 5 for implementation of Deep Neural Networks.

III. IMPLEMENTATION

A. Single Neuron

We have used single neuron for our first deep learning model. We used Sequential single layer model which is very important since it will be used as baseline. The single layer model don’t need any activation function and is having input_size of 5.

1) Code Snippet:
```
layer0 = keras.layers.Dense(1,input_shape=[window_size])
model = keras.models.Sequential([layer0])
model.compile(loss = 'MeanAbsoluteError', optimizer = 'Adam',metrics=['accuracy'])
```
The model is compile with loss function “MeanAbsoluteError” and “Adam” as optimizer. MeanAbsoluteError helps us know about the error during predictions. We used accuracy metrics to keep account of accuracy. We fit the model with reduced dataset and epochs equal to 1000, so we get a better understanding about models behavior.

2) Code Snippet:
```
history = model.fit(dataset,epochs=1000)
```
Now the model is ready and trained with the dataset. We can predict the ConfirmedCases with this model.

3) Code Snippet:
```
model.predict(series[time:time + window_size][np.newaxis])
```

B. Deep Neural Network

Deep Neural Network have input layer, hidden layers and output layer. For the prediction we had used 3 Deep Neural Networks (termed as Model 1, 2 and 3 respectively) which will help us know how no. of layers and no. of neurons affect the prediction and loss of the models.

1) Model 1: The first Deep Neural Network model consist of 3 layers having (10,10,1) neurons respectively. We used “Relu” as activation function and input_size same in all the models. Having same input_size, help us compare the models more effectively.

a) Code Snippet
```
#Input layer
keras.layers.Dense(10, activation = 'relu',input_shape=[window_size])
#hidden layer
keras.layers.Dense(10,activation = 'relu')
#output layer
keras.layers.Dense(1)
```
We used the same loss function in all the models. We compile the model with loss function as “MeanAbsoluteError” and “Adam” optimizer.

b) Code Snippet
```
model.compile(loss = 'MeanAbsoluteError', optimizer = 'Adam',metrics=['accuracy'])
```
We then compile the model with dataset and epochs equals to 1000.

c) Code Snippet
```
history = model.fit(dataset, epochs=1000)
```
The model is compiled and trained with our dataset. The model has been used for prediction of ConfirmedCases.

d) Code Snippet
```
model.predict(series[time:time + window_size][np.newaxis])
```

2) *Model 2:* The second model also have 3 layers similar to the model 1. But the no. of neurons in the input layer have been increased from 10 to 30. This will help us understand the dependency of the model on the neurons. The no. of neurons in layers are (30,10,1) respectively. We used the same activation function "Relu" in the input and hidden layer.

a) *Code Snippet*

```
#Input layer
keras.layers.Dense(30, activation = 'relu',input_shape=[window_size])

#hidden layer
keras.layers.Dense(10,activation = 'relu')

#output layer
keras.layers.Dense(1)
```

We used the same loss function and optimizer.

b) *Code Snippet*

```
model.compile(loss = 'MeanAbsoluteError', optimizer = 'Adam',metrics=['accuracy'])
```

We had compiled the model with dataset and epochs equals to 1000and the predictions has been drawn.

c) *Code Snippet*

```
history = model.fit(dataset, epochs=1000)
model.predict(series[time:time + window_size][np.newaxis])
```

3) *Model 3:* The third model have two hidden layers and will help us understand effects of increasing the layers in the deep neural networks. The layers have neurons (30,20,10,1) respectively. We have used the same activation function "Relu" and with the same loss function and optimizer model has been compiled and predictions has been drawn.

a) *Code Snippet*

```
#Input layer
keras.layers.Dense(30, activation = 'relu',input_shape=[window_size])

#hidden layer
keras.layers.Dense(20,activation = 'relu')
keras.layers.Dense(10,activation = 'relu')

#output layer
keras.layers.Dense(1)
model.compile(loss = 'MeanAbsoluteError', optimizer = 'Adam',metrics=['accuracy'])
history = model.fit(dataset, epochs=1000)
model. predict(series[time:time + window_size][np.newaxis])
```

C. Recurrent Neural Network

Recurrent neutral network are able to store previous output and make decision based on previous output. We used two different models with different no. RNN layers which help us understand behavior of model more effectively. We had also used Lambda layers which allows us to include processing data on the fly to the neural network.

1) *Model 1:* The first model have 1 input layer, 4 hidden layers and 1 output layer. The layers have 2 lambda layers one each at input and output and 3 RNN layers in hidden layers. The output lambda layer is used for scaling, all the outputs are multiplied by 100 so it does not change relative to other outputs.

a) *Code Snippet*

```
#Input layer
tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
input_shape=[None])
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue IV Apr 2020- Available at www.ijraset.com

```
#Hidden layers
tf.keras.layers.SimpleRNN(20, return_sequences=True,activation = 'relu'),
tf.keras.layers.SimpleRNN(10,activation = 'relu',return_sequences = True),
tf.keras.layers.SimpleRNN(10,activation = 'relu'),
tf.keras.layers.Dense(1),

#output layer
tf.keras.layers.Lambda(lambda x: x * 100.0)
```

The model is compiled with "MeanAbsoluteError" for loss function and "Adam" for the optimizer with accuracy metrices.

b) Code Snippet

```
model.compile(loss = 'mae', optimizer = 'Adam',metrics=['accuracy'])
```

We had trained the model with dataset and epochs equals to 1000.

c) Code Snippet

```
history = model.fit(dataset, epochs=1000)
```

The model is compiled and trained with the dataset and predicted ConfirmedCases

d) Code Snippet

```
model.predict(series[time:time + window_size][np.newaxis])
```

2) *Model 2:* The second model have 5 RNN layers whereas the previous model had only 3 RNN layers. The neurons in 6 layers are (30, 30, 20, 10, 10, 1) respectively in hidden layers. We still used lambda layers for input and output. We need to use return_sequence parameter in all the layers which feed into other RNN layers.

a) Code Snippet

```
#Input layer
tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                       input_shape=[None])
#hidden layers
    tf.keras.layers.SimpleRNN(30, return_sequences=True,activation = 'relu'),
    tf.keras.layers.SimpleRNN(30, return_sequences=True,activation = 'relu'),
    tf.keras.layers.SimpleRNN(20, return_sequences=True,activation = 'relu'),
    tf.keras.layers.SimpleRNN(10,activation = 'relu',return_sequences = True),
    tf.keras.layers.SimpleRNN(10,activation = 'relu'),
    tf.keras.layers.Dense(1),
#output layer
tf.keras.layers.Lambda(lambda x: x * 100.0)
```

The model is compiled and then for 1000 epochs predictions has been drawn.

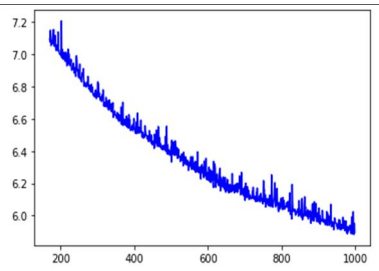
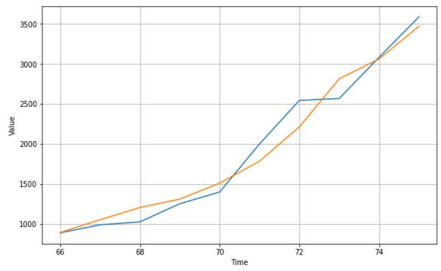
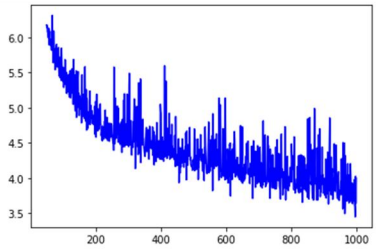
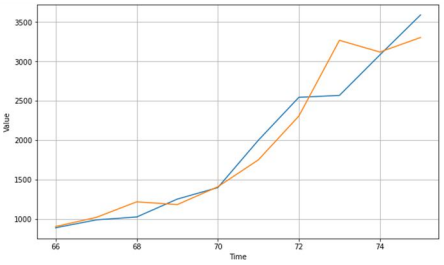
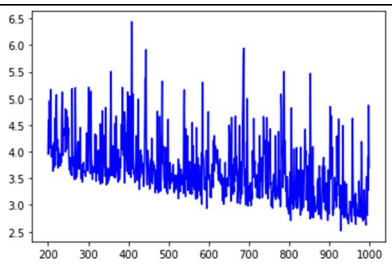
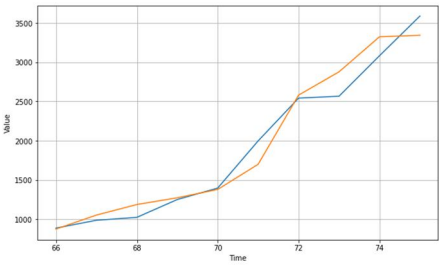
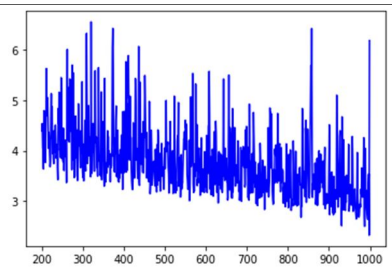
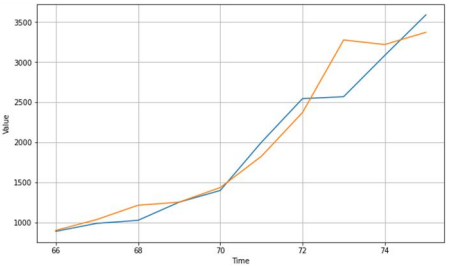
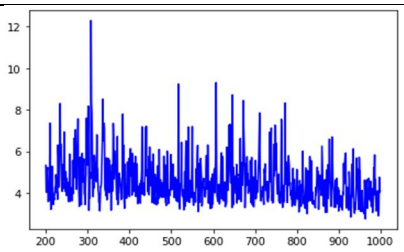
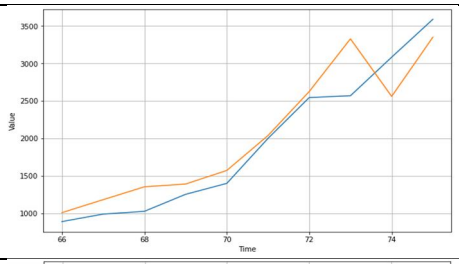
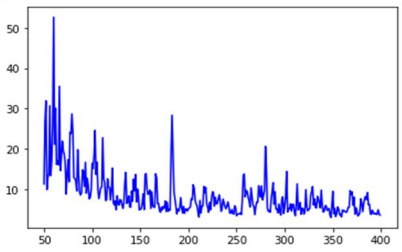
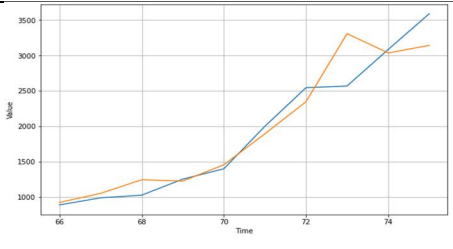
b) Code Snippet

```
model.compile(loss = 'mae', optimizer = 'Adam',metrics=['accuracy'])
history = model.fit(dataset, epochs=1000)
model.predict(series[time:time + window_size][np.newaxis])
```

IV. EXPERIMENTAL RESULTS AND COMPARISION

After Execution of the models, predictions has been drawn. The results will be in the form of graphs. First graph depicts epocs vs. loss and the second graph depicts relation between day and no. of confirmed case. The Blue line in second graphs depicts Actual numbers of confirmed cases while the orange line in the same graph depicts predicted number of confirmed cases. The comparison of these results has been depicted in the following table:

TABLE I
Results And Comparison Table Of All Models

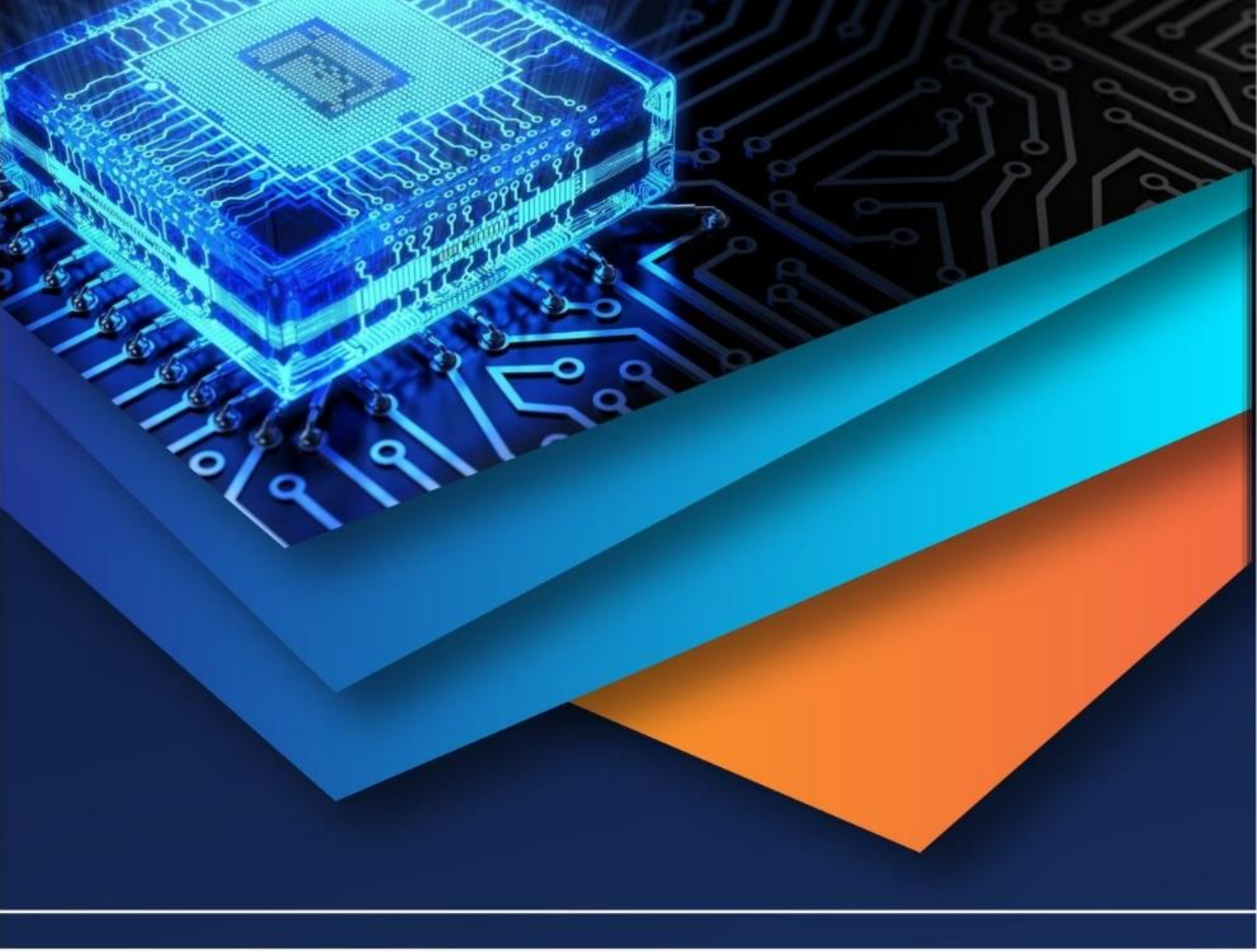
Model Name	Hyper parameters	Loss (after 200 epochs)	Predicts vs actual data
SNN	loss = 'MeanAbsoluteError optimizer = 'Adam' eopchs=1000		
DNN Model 1	loss = 'MeanAbsoluteError optimizer = 'Adam' eopchs=1000		
DNN Model 2	loss = 'MeanAbsoluteError optimizer = 'Adam' eopchs=1000		
DNN Model 3	loss = 'MeanAbsoluteError optimizer = 'Adam' eopchs=1000		
RNN Model 1	loss = 'MeanAbsoluteError optimizer = 'Adam' eopchs=1000		
RNN Model 2	loss = 'MeanAbsoluteError optimizer = 'Adam' eopchs=1000		

V. CONCLUSION

After conducting these experiments, we get minimum loss in the Deep Neural Network Model 2. It was expected that the Recurrent Neural Network Model 2 to have maximum overlapping hence the best prediction. But the Single Neural Network had the most overlapping in the validation and most close to real data. This might be a result because of less data in the dataset because of the latest development in the number of confirmed cased in India hence It is concluded that Single Neural Network happened to work best in this setup.

REFERENCES

[1] Alimadadi, A., Aryal, S., Manandhar, I., Munroe, P. B., Joe, B., & Cheng, X. (2020). Artificial Intelligence and Machine Learning to Fight COVID-19. Physiological Genomics. doi:10.1152/physiolgenomics.00029.2020.
[2] Lek, S., & Park, Y. S. (2008). Artificial Neural Networks. Encyclopedia of Ecology, 237–245. doi:10.1016/b978-008045405-4.00173-7.
[3] Längkvist, M., Karlsson, L., & Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, 42, 11–24. doi:10.1016/j.patrec.2014.01.008.
[4] Connor, J. T., Martin, R. D., & Atlas, L. E. (1994). Recurrent neural networks and robust time series prediction. IEEE Transactions on Neural Networks, 5(2), 240–254. doi:10.1109/72.279188.
[5] Wibisono, T., Aleman, D. M., & Schwartz, B. (2008). A non-homogeneous approach to simulating the spread of disease in a pandemic outbreak. 2008 Winter Simulation Conference. doi:10.1109/wsc.2008.4736431.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY