Weather Forecasting and Monitoring using Machine Learning and Deep Neural Network Models

Dr. Basudeba Behera[1], Nitish Kumar[2], Mukesh Ranjan Mahato[3], Banoth Krishna Prasad[4] and Dr. Vijay Bhaskar Semwal[5]

Department of Electronics and Communication Engineering

National Institute of Technology Jamshedpur, Jharkhand, India[1234], MNIT Bhopal, M.P., India[5]

***E-mail:-** basudeb.ece@nitjsr.ac.in[1]*

DOI:- https://doi.org/10.47531/MANTECH/ECC.2021.17

Abstract

The exponentially increasing traffic on the road highways is creating safety issues for the vehicles. It is a challenge to make the highways safe and secure against traffic congestion and road accidents, which may happen due to a continuous rise in both traffic density and speed of the vehicles. It is imperative to devise vehicular communication technologies to ensure a reliable and powerful driving support system for the safety and efficiency of road transportation. Vehicular communication technologies are being designed to make early detection of the perilous situation and exchange the information among the vehicles, which may be used for issuing the warnings to the drivers and /or navigational aids. In addition, non-safety applications of vehicular networks are being embedded in the vehicles for the purpose of infotainment and the comfort of the passenger. In this paper, we present a comparative performance evaluation of three technologies- WAVE, WiMAX, and LTE-V2V with the aim to examine their relative strength in a dynamic vehicular environment. The impact of node density, node speed and beacon transmission frequency on the throughput, delay and packet delivery ratio is analyzed. Through numerical results, LTE-V2V is shown to outperform the other two alternatives in terms of throughput and delay.

***Keywords: -** Deep Neural Network, Regression, Monsoon Break, Satellite System.*

INTRODUCTION

This world is a very diverse place with respect to habitation, weather and atmospheric conditions. So, we need advanced weather monitoring and prediction system not only to avoid calamities but for a better understanding of the earth and future developments in Science and Technology. Mathematical simulations developed by scientists have been able to predict the weather in advance, where data is fed as input for computation [1]. Though there has been a great advantage by this method, but now is the time to advance our research as an era of automation is on the dawn. So, we come up with automated forecasting using Neural Network, which provides better-optimized results [2]. Giri et al. [3] talked about drip irrigation using microcontroller and solenoids. The achievement is the decrease in soil erosion. However, the drawback was its incompatibility with real-life boundaries. Bansal et al. [4] has presented work on WSN using the microcontroller and GSM module. Their work is used to inform farmers about the environs and soil conditions. However, this technique is not suitable for us, as deploying and using GPS in rural areas is difficult due to poor connectivity. Devika et al. [5] has introduced automation in watering plants by using simple Arduino.

As the previously mentioned systems were not based on cloud tech so the drawback was the restriction in resource sharing as the information could not be accessed on global basis. An intelligent irrigation system using microcontroller 8051 and GSM module was worked by Ramu et al. [6] and Zareen et al. [7]. Today, we have much better options other than GSM, such as cloud computing, machine learning, neural networks and much more modern technology which can be more efficient in managing different weather conditions, natural calamities that will build a better lifestyle for the people of our country as well as for the rest of the world [8], which in turn will be much beneficial in maintaining a database of the different types of soils, different weather present

in different regions of India. Mishra et al. [9] and Stesel et al. [10] used Arduino Uno and Arduino Nano, respectively, along with the Wi-Fi module to achieve the same work. Krishna et al. [11] presented a paper for weather forecasting and analysis of the surroundings so that better measures could be taken ahead of any natural calamity as well as farmers could manage their wellbeing of crops in extreme weather situations. They presented time series calculations of weather data which is again mathematical and empirical calculation. Holmstormet. al. [12] presented that when ML applied to weather forecasting can create an impact. The work presents the Linear Regression model and explains about how accurate predictions can warn the farmers ahead of disasters, how soils can be prepared ahead of climate change, people can be warned and evacuated ahead of the flood, excessive rainfall, tsunami and other disastrous calamities.

In this work, practical problems solved with the help of the technologies of Apache Hadoop to store data and apply neural network-based calculations to modern technology. The main objective of the proposed research work is to use the Neural Networks to achieve the following goals:

- To monitor the ongoing processes of the atmosphere such as weather pollution, global warming and flood-like situations.
- To replace the old time-consuming method of the empirical system with an efficient automatic forecasting system based on Perceptrons.

SYSTEM ARCHITECTURE

1. Core of the Matter

This paper introduces a new realm on the weather data so far collected by introducing Machine Learning (ML) Models. Machine learning actually finds the relationships between the training features and the target variables.

Like for example we have series

5	10	30	45	87	x
12	22	62	92	?	Y

We need to find out the relation such that

$Y = mx + C.$

This whole of the calculation is done by Machine learning by taking the arbitrary value of m (weights) and C (biases).

Figure 1: Prototype Traditional programming

Figure 1 shows that traditionally works on rules and the data provided to find the answers to the problems. In the case of the activity detection, the rules (the code we wrote to define types of activities) acts upon the data (the person's behavior, speed, habit) in order to determine the activity status of the user (whether they were walking, running, moving, etc.).

Figure 2: Prototype of ML Model

In ML, instead of trying to define the rules, Figure 2 clearly expresses the prototype of ML models as opposite to the programming language, we provide the answers (labels) along with the data, and the machine will find the rules showing relations between the answers and the data and Figure 3 shows the model when we provide the test data. The Model (rules) works on the test data and provides predictions for the future.

Figure 3: Model in Test Phase

STEPS OF MODEL BUILDING

The following steps were considered during the building of the model.

- Process the dataset/preparation
- Make the deep neural network
- Train the DNN
- Test the DNN
- Compare the result from the DNN to another ML algorithm

Preparation/ Preprocessing of Dataset:

In this article, the data from Jaipur city, India, for the year 2016 to 2018 taken. We prepare the data by specifying the feature and target variable using the code. We drop the mintempm, maxtempm, date from the various variables available. X-axis label shows all variables except meantempm, and Y-axis label shows as target variable consisting of meantempm.

```
data = data.drop(['mintempm','maxtempm','date'],axis=1)

# X will be a pandas dataframe of all columns except meantempm
X = data[[col for col in data.columns if col != 'meantempm']]

# y will be a pandas series of the meantempm
Y= data['meantempm']
```

We will go on to processing the dataset and getting the dataset ready to be fed into our models. We will get rid of any features with missing values; then, we will encode the categorical features and normalize the data to make it between 0 and 1. Also, with the help of the following code, we divide our dataset into training and test set.

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)

X_train = preprocessing.scale(X_train)

X_test = preprocessing.scale(X_test)
```

Model the Deep Neural Network:

- Define a sequential model
- Add some dense layers
- Add function for the hidden layers.

Here is the code for the Neural Network layers.

```
model = Sequential()
model.add(Dense(13, input_shape=(36,), activation='relu'))
model.add(Dense(13, activation='relu'))
model.add(Dense(13, activation='relu'))
model.add(Dense(13, activation='relu'))
model.add(Dense(13, activation='relu'))
model.add(Dense(1,))
model.compile(Adam(lr=0.003), 'mean_squared_error')

# Runs model for 2000 iterations and assigns this to 'history'
model.summary()
```

This is a basic simple example of what is Model building in a neural net.

Sequential: sequence of the neural net is defined in the network.

Dense: Adds a layer of neurons. Each layer is supplied with an activation function for their firing up.

Relu: effectively means, If X>0 then it gives output as true else false.

Softmax: It takes a set of values and effectively picks the biggest one.

For example, if the output of the last layer looks like

[0.1, 0.1, 0.05, 0.1, 9.5, 0.1, 0.05, 0.05, 0.05]

It removes hardcore coding by automating things like [0,0,0,0,1,0,0,0,0].

As a result of the implementation of the above code, Figure 1shows the model summary consisting of 13 layers of Neurons with 36 flattened layer of input of shape, which means 36 variables are the input variables with activation functions as Relu.

Model Summary:

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_32 (Dense)             (None, 13)                481
_________________________________________________________________
dense_33 (Dense)             (None, 13)                182
_________________________________________________________________
dense_34 (Dense)             (None, 13)                182
_________________________________________________________________
dense_35 (Dense)             (None, 13)                182
_________________________________________________________________
dense_36 (Dense)             (None, 13)                182
_________________________________________________________________
dense_37 (Dense)             (None, 1)                 14
=================================================================
Total params: 1,223
Trainable params: 1,223
Non-trainable params: 0
```

Figure 4: Model summary

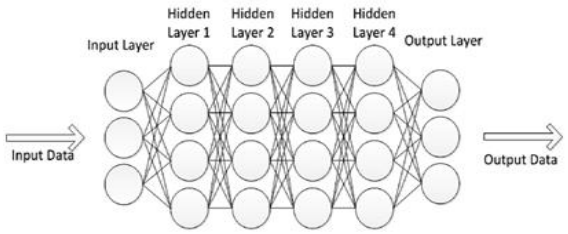

Figure 5: Representation of Model (Neural Network Architecture) [13]

Figure 4 is a representation of the Neural Network formed as a result of the implemented code. The first layer is the input layer, which is always equal to the number of input feature which is fed to the system. And middle layers are hidden layers, and the last layer is the output layers which is equal to the number of output required.

Train the model:

```
history = model.fit(X_train, y_train, epochs = 6000,
                    validation_split = 0.2, verbose = 1)
history
```

```
Epoch 2916/6000
425/425 [==============================] - 0s 71us/step - loss: 0.0412 - val_loss: 3.6510
Epoch 2917/6000
425/425 [==============================] - 0s 78us/step - loss: 0.0333 - val_loss: 3.5313
Epoch 2918/6000
425/425 [==============================] - 0s 71us/step - loss: 0.0337 - val_loss: 3.7637
Epoch 2919/6000
425/425 [==============================] - 0s 104us/step - loss: 0.0350 - val_loss: 3.7888
Epoch 2920/6000
425/425 [==============================] - 0s 76us/step - loss: 0.0325 - val_loss: 3.6795
Epoch 2921/6000
425/425 [==============================] - 0s 76us/step - loss: 0.0363 - val_loss: 3.6103
Epoch 2922/6000
425/425 [==============================] - 0s 86us/step - loss: 0.0203 - val_loss: 3.6877
Epoch 2923/6000
425/425 [==============================] - 0s 85us/step - loss: 0.0149 - val_loss: 3.6756
Epoch 2924/6000
425/425 [==============================] - 0s 87us/step - loss: 0.0135 - val_loss: 3.6836
Epoch 2925/6000
425/425 [==============================] - 0s 77us/step - loss: 0.0113 - val_loss: 3.7098
```

This is a simple example of how our training model for regression works. It works on training and testing data along with 20% cross-validation data at each epoch. Here verbose = 1 does mean that it will show the calculation at each epoch.

RESULTS

Once we execute the above code for the training dataset. Figure 5 graph shows that the testing and

training dataset plot where blue represents the training data had and red testing data. In the beginning 100 iterations, we Overfitting as the model performed very well on the training data but failed to work well for the testing data. Later on, the 6000 iterations Model worked well for the training as well as testing data as we can see that Loss is decreasing with increasing iterations, and It works well for the training as well as testing dataset.

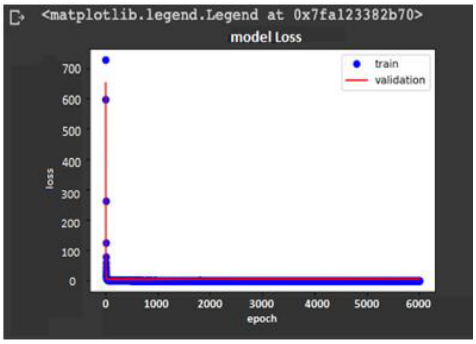

Figure 6: plot for Training and Test set

When we have the predicted values against the training as well as test set, we observe the relations as Linear Figure 6 is the plot for the actual train/test values and the predicted value from the graph; we can infer that there very less variance between the actual and predicted value. Figure 6 shows the relationship between our features and the labels, which is of the relation y = mx + c. Hence there is the continuous variable plot. Hence, there is y = mx + c plot.

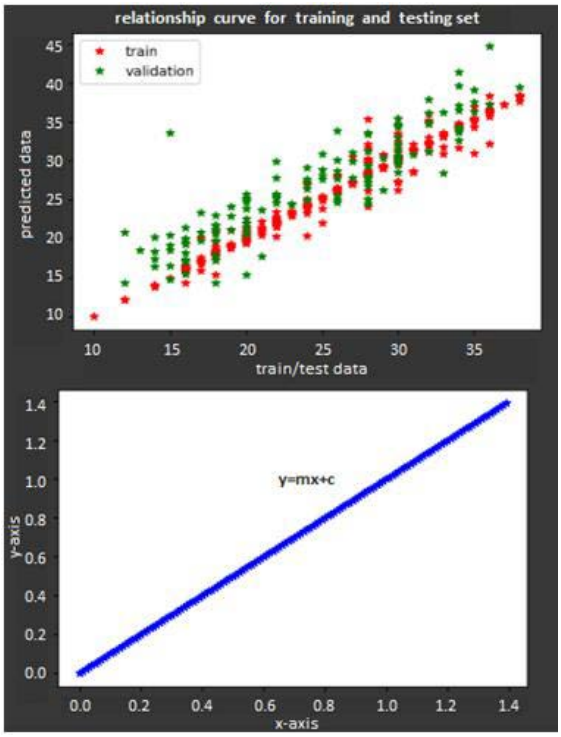

Figure 7: Relationship between testing and training data

SOFTWARE AND DATASET USED

To perform the computation, we worked Google Collaboratory with data from Jaipur city. In the Google Collab, we prepared the data into the training and testing set then preprocessed it using the sklearn library. We went on to collect all the features except meantempm, which was our target variable. After having the feature and target variable, we went on to build the Neural Model, which consisted of 1 input layer having 36 flat variables and four hidden layers with 32 neurons and 1 output layer having 1 output variable. This is the fully connected layer. After the model is built, we compile the Model with optimizer such as Mean squared error and Adam optimizer having a learning rate of 0.03. Then we further go on to train the model using a training dataset, with 20% being a validation dataset at each epoch of 6000 total epochs. After training, we get the relationship between the input and output variable, which is a rule to predict the future weather like y = mx+ c, finally, we plot the straight-line graph out of our data.

CHALLENGES

At times our network fits well with the training dataset and gives good accuracy, but at many times it does not perform well with the testing dataset, this is the condition of overfitting, and we need to avoid it. To know that our model is overfitting we divide our dataset into three parts - training set, dev set (also known as cross-validation or hold-out) and test set. Learning takes place with only one part. Dev set is used for our progress that what is the current direction. The test set is used in the end to evaluate our performance of the model. Figure 7 shows the perfect plot for the different sections of data.

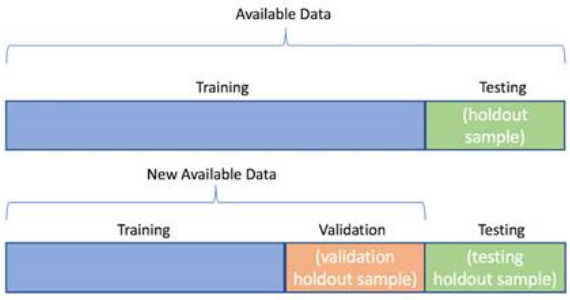

Figure 8: Train, Test and validation dataset

1. Ways to prevent overfitting

a) Cross Validation

The idea behind the Cross Validation is generating many mini train-test splits. Use these splits to tune your model. In k-fold cross-validation, we partition the data into k subsets, so-called folds. Then, we train the algorithm on k-1 folds while

using the remaining fold as the test set (called the“holdout fold”). Cross-validation allows you to tune hyper with our original training set. This allows us to keep our test set as a truly unseen dataset for selecting your final model.

b) Train with more data

Having more and more data is the key to overcome errors. A child can learn better by numerous examples; similarly, machine trains better on more and more datasets.

c) Remove features

We can manually improve our model by removing irrelevant input features.

d) Dropout

In this method, we reduce the number of neurons to analyse if there is better accuracy.

e) Early Stopping

It works on the principle that the moment we achieve certain accuracy by certain iterations and for further iterations, if the accuracy decreases, then stop the iterations and keep the required accuracy.

CONCLUSION

This work demonstrates how Deep Neural Networks (DNN) can be directly applied to the output of numerical weather models by using observed data to annotate the samples. The design of the (DNN) used in our experiments is very simple. Despite their simplicity, results show that Neural Layers can be used to interpret the output of weather models. We worked on the data of Jaipur city for two years. By building the Neural Model on training and testing data for 6000 epochs, we achieved only a 2% loss on training data and 5% loss on the testing and validation data. Although the model was a bit overfitting, but it performed better than the traditional models, and it was much more convenient. We also used the concept of early stopper via the Callback method to stop at the time the accuracy is achieved, and it goes on to decrease the accuracy.

This paper demonstrated the Numerical Method of monitoring, but today with the advancement of technology, it’s edging towards the use of better deep learning techniques which works on images collected from different weather scenarios. Images processing and predicting weather is going to be more precise in which convolutional neural networks (CNN) will be used. CNN models will be used to interpret numerical weather model data, which, by capturing the spatial and temporal relationships between the input variables, can produce local forecasts. Further, for correlation between different images to develop a model, we will use RNN (recurrent Neural network) to relate the weather between the consecutive days.

REFERENCES

1. V. B. Semwal, J. Singha, P. K. Sharma, A. Chauhan, and B. Behera, “An optimized feature selection technique based on incremental feature analysis for bio-metric gait data classification,” Multimed. Tools Appl., no. December, p. 18, Dec. 2016.
2. M. Giri and D. Wavhal, "Automated Intelligent Wireless Drip Irrigation Using Linear Programming," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 2, no. 1, pp. 1-5, 2013.
3. J. Goswami and A. Choudhury, "Dynamic Modeling Technique for Weather Prediction," International Journal of Computer Science & Engineering Technology (IJCSET), vol. v, no. 1, pp. 1-2, 2014.
4. D. Bansal and S. Reddy, "WSN Based Closed Loop Automatic Irrigation System," International Journal of Engineering Science and Innovative Technology (IJESIT), vol. 2, no. 3, p. 229–237, 2013.
5. C. M.Devika, K. Bose and S. Vijayalekshmy, "Automatic plant irrigation system using Arduino," IEEE International Conference on Circuits and Systems (ICCS), Thiruvananthapuram, India 20-21 Dec, 2017.
6. M. R. and C. H. Ramu, "Cost effective atomization of Indian agricultural system using 8051 microcontrollers," International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 7, pp. 2563-2566, 2013.
7. S. Zareen, K. Zarrin, A. Ali and S. D. Pingle, "Intelligent Automatic Plant Irrigation System," International Journal of Scientific Research and Education, vol. 4, no. 11, p. 6071–6077, 2016.
8. R. Vagulabranan, M. Karthikeyan and V. Sasikala, "Automatic Irrigation System on Sensing Soil Moisture Content," Int. Res. J. Eng. Technol., vol. 3, no. 1, p. 2012–2019, 2015.
9. D. Mishra, A. Khan, R. Tiwary and S. Upadhay, "Automated Irrigation System-IoT Based Approach," International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU). IEEE, Bhimtal, India 23-24 Feb., 2018.
10. A. Stesel and A. Osanlou, "A Sustainable Indoor Plant Production Management System with Wireless Internet Access," Young Researchers in Electrical and Electronic

Engineering (EIConRus), 2018 IEEE Conference of Russian IEEE Moscow, Russia 29 Jan.-1 Feb., 2018.

11. G. V. Krishna, "An Integrated Approach for Weather Forecasting based on Data Mining and Forecasting Analysis," International Journal of Computer Applications (0975-887), vol. 120, pp. 1-2, June 2015.
12. M. Holmstrom, D. Liu and C. Vo, Machine Learning Applied to Weather Forecasting, 2016.
13. Available Online: https://www.quora.com/What-is-a-fully-connected-layer-in-DEEP-learning-I-didnt-understand-clearly-because-no-ones-contents-are-explained-in-a-simple-detail-Everyone-twisted-this-concept.