Edge and Fog computing based Energy Prediction using Neural Networks

1 Abstract

Smart grid being the upcoming technology to respond to our quickly changing electric demand. The Internet of Things(IoT) and cloud computing technology is the major contributor to make the smart grid network more reliable, efficient and fast. Considering the aspects of cloud technology and its limitations in terms of delays and latencies incurred during the processing of data. This paper shows the performance and accuracy of energy predictions algorithms using neural networks on various hybrid edge and fog computing architecture based implementation.

Keywords: smart meters,smart grid, neural networks, fog computing, edge computing, energy predictions, internet of things, smart meter communication network

2 Introduction

This paper discusses about the techniques and solutions of an extension of smart grid where one can, not just collect and record the data but can also make predictions using low computing hardware. In this paper we have discussed about different architecture and software design implementations for collection and prediction of the energy usage for short-term,long-term and very-long term periods[1] as part of smart energy environment for smart grids.

The paper is divided into four major sections: *Edge and Fog Computing based hybrid architecture, Neural Network, Software Implementation, Results and Conclusion.* In the first section, we will discuss about two major types of architecture using edge and fog computing following with neural networks used for energy usage predictions. In the latter sections, we will present the details about the test and implementation with results.

3 Edge and Fog Computing based hybrid architecture

In the era of IoT, where huge amount of data is being generated from numerous devices and interactions. The cloud computing has played a significant role for processing, analysing and storage of data cost-effectively. But on it’s own it can’t handle the influx of information, creating issues related to delays and performances for devices that are far away from the centralised sources in cloud.

Considering the nature of data and its vicinity of importance in the smart energy environment. And processing latency and data significance as per the geography being the motivation. A hybrid architecture of edge and fog computing is being considered for the energy prediction as part of smart energy environment. There can be various ways the concept of edge and fog can be deployed in the smart energy environment. Basically, the amount of processing power and storage will be in increasing order as we go away from the source data nodes i.e the sensors.Wherein, the smart meters[2] at consumer location being the edge node with some processing power and storage, and higher capability processing and storage units on the fog node.

The smart meter at the edge with processing power in megahertz and storage capacity in megabytes can serve various purposes mainly collecting pre-processed energy data from sensor nodes being a part of Home Area Network, network connectivity and transmission of pre-processed data to fog node. In addition to this, it can also serve as an user interface to control appliances as per the predictions provided to consumer from the fog nodes which allows them to save money and eventually save energy.

As the energy predictions can be classified into three major categories being short-term, long-term and very-long term long predictions. The platform resources required for the processing of these predictions will play a significant role in deciding the processing power and storage requirements.

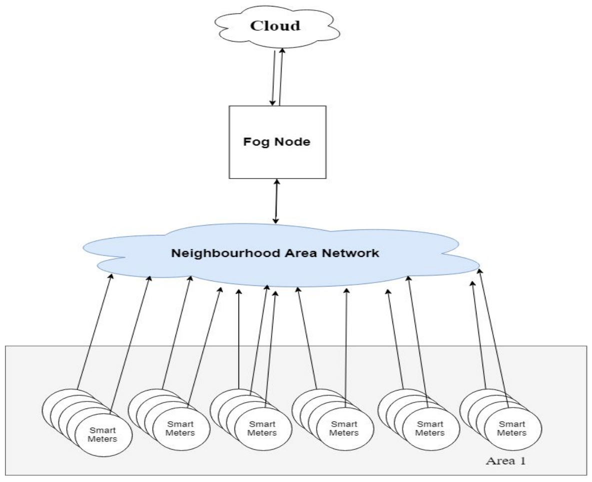
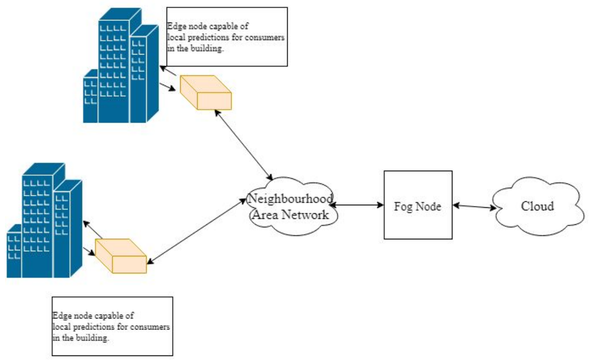


Fig 1: Edge nodes connected to fog node via neighbourhood area network

In this paper, we have considered two types of architecture for the study which are shown in Fig 1 and Fig 2. The architecture in Fig 1, basically considers all the smart meters nodes as edge which sends the pre-processed data to the fog node for further processing and energy predictions. Here fog node is responsible for all three kinds of energy predictions mentioned earlier in this paper. As per the storage and other computing requirements, fog node may use the cloud services for deferred processing task. In this case, all the houses in the county or sub-urban area can be considered as edge nodes sending data to fog node which can be deployed at the county office of the electric distribution company.



Edge node capable of local predictions for consumers in the building.
Neighbourhood Area Network
Fog Node
Cloud
Edge node capable of local predictions for consumers in the building.

Fig 2: Edge node with more processing and storing capabilities.

A more powerful edge node in terms of processing and storage capabilities is shown in Fig 2 wherein the node is capable of energy predictions for the local community or buildings. But not all kinds of predictions can be done at this node as edge has limited processing power and storage capacity compare to typical fog nodes. This architecture can be also be used in industrial sectors apart for buildings or community housing. This additional processing layer can contribute to lesser latencies and delays caused by networks. As per recent advancements in hardware technology, the cost incurring for this additional layer doesn't have significant effect when it is compared to adding fog node platform.

In the later sections, we will discuss about the implementation and results obtained considering above two architectures for energy predictions.

4 Neural Network

4.1 Input variables

As we discussed already in earlier part of paper that energy prediction is very much related with weather conditions and human behavior patterns. This makes whole prediction model very simple for boiling down to very specific variables. Each variable has their own importance in the predictions. The following list are the variable used for the input of neural network including their measurement unit in brackets.

1. Temperature (F)
2. Humidity (%)
3. Month (1-12)
4. Day of week (0-6)
5. Weekend/Weekday (0/1)
6. Hour of the day (0-23)
7. Minute of the day (0-3599)
8. Holiday (0/1)

To train neural network based on weather conditions and human behavior patterns these variables makes most impact on electricity usage. Regarding to collection and process of cleaning the data we will discuss this in following part.

4.2 Data Collection and processing

Data collection and processing was most crucial part for the algorithm because predictions are based on the reliability of the data. To collect weather data we used weather station data located nearby the housing community. For experimental purpose we used weather data from weather underground and energy usage data from Pecan Street. Major data processing problem is to deal with the discontinuity of data. While grabbing data from the source it is very important to check the continuity of data. If the the data is not continuous either you have to re generate the data for missing points or exclude the whole data. It will be very simple to regenerate or obtain weather data according to historical data points, but if some houses are missing electrical usage data one can not regenerate the missing data points. Here, for this case one will have to drop the houses whose data is not continuous,

For missing data point using moving average is best option as hourly weather is continuous change and there are very less chance of unexpected radical changes in an hour for temperature and humidity data.

Example: If one there is one data point missing (T2) between T1 and T3. As we know all are hourly data points, it is highly probable that T2 will be in range of T1 to T3. As per the thermodynamics and analogous world this shall hold true for temperature.

So we can easily determine missing data point with,

$$T2 = (T1 + T3) / 2$$

For data types like Holiday, Weekday, Month and Hour it will be easy to write small scripts to generate the data. They will be followed by the simple formulas.

One can use famous module of python which is pandas which can separate day, weekday, hour, minute from timestamp. example,

*```Python*

df['Hour'] = df['DateTime'].dt.hour

df['Minute'] = df['DateTime'].dt.minute

df['Month'] = df['DateTime'].dt.month

df['Day'] = df['DateTime'].dt.day

```

Here 'DateTime' is actual timestamp from the original data and this snippets creates data for Hour, Minute, Month, Day.

### 4.3 Neural Network Algorithm

For the prediction purpose we have used the Neural network to generate energy predictions. Neural networks like Long Short-Term Memory (LSTM) recurrent neural networks are able to almost seamlessly model problems with multiple input variables. This is a great benefit in time series forecasting, where classical linear methods can be difficult to adapt to multivariate or multiple input forecasting problems.

Prediction purpose we took 2 years worth of continuous electricity usage data of 12 houses from pecan street. Using one house of electrical usage was very low to predict so we choose bunch of house's usage and used summation of it as the electricity usage. we divided collected data into two major parts. 1) Training data set, 2) Testing dataset. Dividing them om 8:2 ratio we made predictions based on input variables we discussed above.

In this process, First we divided the dataset into 2 parts. We trained neural network using Keras library of python which uses tensorflow at backend. The whole network consists single layer of neurons and trained for 20 epochs. We experimented designing network using multiple values of epochs but from results we found that there after 20 epochs it isn't making significant difference on errors.

### 4.4 Testing and Validation

To test and validate the modal we used another 20% of the dataset which we separated earlier. We stored the trained model and separated actual electricity usage from the data set. By putting input variables in stored trained model we tried predicting the electricity usage.

Calculating accuracy and error we used actual electricity usage and compared it with the predicted value. The error was measured on bases of Root Mean Squared.

### 4.5 Actual Predictions

Best thing about using Keras library is that you can store the trained model and load it when you want for next predictions. We can store the whole trained model into '.h5' file and load it when we want to make real time predictions.

## 5 Software Implementation

Prediction is one part of the whole project we also have implemented the design of collecting data from edge nodes. This software implementation is very diverse and scalable that can be used in more than one edge/fog node architectures discussed in earlier part of the paper.

Collecting data from different hardware is kind of difficult task in terms of compatibility and diversity. Here we solved both the problems using the REST based APIs listener integrated with the SQL database.

RESTful APIs are the normal HTTP/s requests which can be send or received over the internet. Any hardware compatible with facility of TCP connection can do that. The whole implementation of APIs are so simple and lightweight that it can be scalable without many platform dependencies.

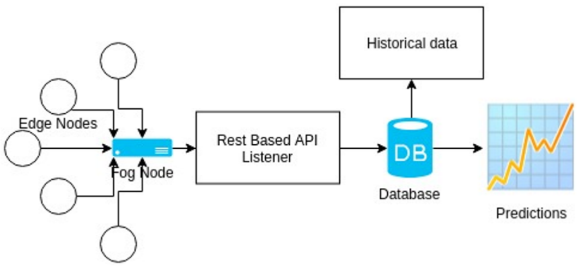


*Fig 3: Implementation Architecture*

Here on diagram above different edge nodes are communicating with the fog node where API listener is listening to the requests. Each edge node will send request and include electricity usage data inside the request. On application layer the data will be sent using JSON format that listener can understand. For particularly energy usage data the JSON schema will be like following.

*{*

*'HouseID': 10046566393,*

*'unit': u'kW',*

*'timestamp': u'4-29-2017T16:30L22',*

*'TotalUsage': u'12.00'*

*}*

Here the schema shown above is example of the data exchanged in experimental set-up. *HouseID* represents the unique number of the house in the
```

community, *Unit* is interpretable as the unit of measurement which is *KiloWatt* here, timestamp is the actual timestamp format datetime and last, *TotalUsage* is electrical usage for the last hour. The frequency of sending data can be less or higher depending upon the data collection and period of predictions.

The whole data will be parsed by the fog node and be stored database. Here we used SQL type of database. The database is used to store the data for the training of the predictions and the historical data viewing.

6 Results

6.1 Hardware performance

Edge/Fog node performance

The proposed implementation of design is very light in terms of performance throughout the tests. There are 2 major hardware performance results we considered to measure. One is performance at Edge node while sending electricity usage data to fog node. In that process, we used raspberry pi to replicate the real-world scenario where we emulated the community of small houses sending data to fog node using port forwarding service on the internet. The edge node and fog node communication uses the client-server model using REST APIs being application services on the network.

Raspberry pi running Raspbian OS being the edge node sending data used 2.3% of CPU which was platform dependent wherein the frequency of data transmission was tested per minute and per second. As fog node, we used Intel i7 laptop running Ubuntu OS to test the performance. In ideal listening situation, the CPU usage observed 1.7%. We emulated 12 edge nodes(consumer houses) sending data to the fog node which doesn't results into significant CPU cycles consumption. All 12 houses sending data at each second meaning 12 x 60 = 720 requests per minute. For that kind of load server's CPU usage was only about at 2.7% of Intel i7 CPU.

Neural Network

The neural network we used, LSTM type is 1 layered neural network. for 20 epoch performance it takes 70 seconds on 55% of the Raspberry Pi CPU. Which is quite less comparing with other type of ML/NN algorithms. Talking of running the same in i& processor it takes only 10 seconds with 62% of total CPU usage. Here 100% means all running all cores bottleneck at same time.

If we make small changes and run it for 50 epochs it takes nearly 120 seconds on Raspberry Pi. The whole performance was measured while training and testing the neural network model. In case we have already stored model and we are making predictions from loaded model it will be very fast more like real-time. Total data we used was 2 years worth hourly data of 8 columns and 2 x 365 x 24 = 17,520 data points.

6.2 Accuracy

Neural network algorithm used here have quite impressive accuracy of 97%. Another thing we experimented is what with each epoch the error decreases.

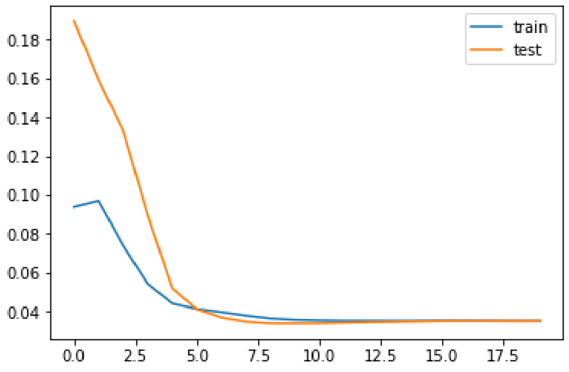


Fig 4: Epoch vs RMS Error

Here its graph of 20 epoch neural network where we can see accuracy increases with each iteration and gets constant after some time.

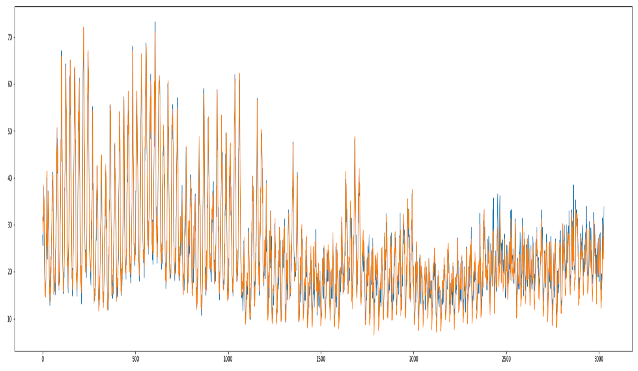


Fig 5: Predictions vs actual Usage

The interesting fact we get to know from digging deep into algorithm analysis that each input variable has their own impact on the output.
Like Temperature, Weekday and weekend are most important for the accuracy. Holiday, Hour of the day have lesser impact than others. And last, Humidity, Holiday has less impact on the error rate than every other input.

7 Conclusion

Predicting energy usage can be very useful for energy distributors as wells as consumers. Here we can put this whole thing using 2 different scenarios solving 2 different problems. From side of electricity generators what they can do from collected data and predictions is that, they can judge the requirement of specific grid and produce accordingly. This can avoid the energy wastage and can avoid blackouts as well.

Not limited to just distributor. This whole concept has more to offer to specific users/community as well. One can look on data predictions and find out the electricity pricing from it. This can help users to save the money and get most from their grid.

One more thing here to notice is that, Both of the things mentioned above can not be implemented at same level. It will create contradictory effect. One can implement the whole design in for one side only. It will be really interesting to see how user and distributor both can get advantage from this at the same time.

Discussing the technical aspect of the whole implementation, It looks that the whole design is so compatible and scalable that it can be used in many current smart grid designs. Example, The REST APIs and database thing can scale to any level accepting thousands of requests at the time. Also, neural network algorithm which is so light that even raspberry pi can handle it. This can lead to very cheap or cost-effective implementation in industry.

8 References

[1] K. Muralitharan, R. Sakthivel, R. Vishnuvarthan *"Neural Network based optimization approach for energy demand prediction in smart grid"*

[2] Yasin Kabalci *"A survey on smart metering and smart grid communication"*

[3] Otavio Carvalho, Manuel Garcia, Eduardo Roloff *"IoT Workload Distribution Impact between Edge and Cloud Computing in a Smart Grid Application"*