Time Series Analysis, Forecasting and Clustering on Household Electricity Data

Midhush Manohar
Computer Science
PES University
Bangalore, India
mimosk25@gmail.com

Naveen Suresh
Computer Science
PES University
Bangalore, India
naveen.213@gmail.com

Srikumar S
Computer Science
PES University
Bangalore, India
srik8552@gmail.com

***Abstract*—For our project, we attempt to analyze, forecast and cluster time series data obtained from electricity sensors of a household. We aim to provide basic analysis with changes in seasonality and trends in data. We also attempt to implement and compare various time series forecasting methods. We finally attempt to cluster slices of time series data effectively using a density initialized clustering algorithm.**

***Index Terms*—Time series, forecasting, ARIMA, LSTM RNNs, Clustering**

I. Introduction

Across the globe, heavy dependence on electricity results in high power demand and hence it requires planning of resources of electricity well in advance to ensure a continuous supply of electricity both now and in the future. Therefore, modelling, forecasting and clustering of electricity consumption is of great importance in deregulated electricity markets for all of the stakeholders: energy wholesalers, traders, retailers, and consumers. The ability to accurately forecast the future electricity consumption will allow them to perform effective planning and efficient operations, leading to ultimate financial profits for them. Moreover, energy-related time series forecasting and clustering plays an important role in the planning and working of the power grid system; for instance, accurate and stable wind speed forecast has primary importance in the wind power industry and make an influence on power-system management and the stability of market economics [1].

A time series is a sequence of real-valued signals that are measured at successive time intervals. Time series data occur naturally in many application areas such as economics, finance, environment, and medicine and often arrives in the form of streaming in many real-world systems. Time series prediction and clustering has been successfully used in a wide range of domains including speech analysis, noise cancelation, and stock market analysis [2].

A. Forecasting

Previous work on similar datasets have followed different approaches. AR(1) with a high pass filter was used to forecast data on monthly electric energy consumption in Lebanon. [3] ARIMA was found to have a lower error rate when predicting monthly and quarterly consumption of electricity in a household [4].

Differences are taken instead of absolute values of Y to make the data stationary. Predictions of weekly and monthly usage are performed using ARIMA. However for making daily predictions using ARIMA on IoT data such as this the seasonal component must be considered. Thus SARIMA is used to model the data instead.

These methods all need to deal with the whole dataset to identify the parameters of the model when facing new coming data, which is not suitable for large datasets and online time series prediction.The dataset used contains measurements taken every minute. Data taken so frequently is susceptible to a lot of white noise, which results in vanishing gradients. To address this problem, online learning methods are explored to extract the underlying pattern representations from time series data in a sequential manner. Compared to traditional batch learning methods, online learning methods avoid expensive retraining cost when handling new coming data. Due to the efficiency and scalability, online learning methods including methods based on linear models, ensemble learning, and kernels have been applied to time series prediction successfully [2].

Recent studies have shown the classification and prediction power of the Artificial Neural Networks. This has been supported by the advancements of technology, which is an essential factor for successfully deploying a neural network as a lot of computational power is essential [5]. It has been demonstrated that a neural network can approximate any continuous function [6]. Neural networks have been successfully used for forecasting of financial data series. The classical methods used for time series prediction like Box-Jenkins, ARMA or ARIMA assumes that there is a linear relationship between inputs and outputs. Neural Networks (LSTMs) have the advantage that can approximate any nonlinear functions without any apriori information about the properties of the data series.

Amongst Neural Networks themselves, and in particular, LSTMs, there are several approaches (architectures) which can be applied for Time Series data. These include:

- **Sequence to Sequence Architecture**
- **Ensemble LSTM Network via Adaptive Weighting**

B. Clustering

To cluster time series data, there is the necessity to define the measure of similarity (or dissimilarity) between two waves. Many such measures exist, including Euclidean Distance, Dynamic Time Warping Distance, Pearsons correlation coefficient, Mahalanobis Distance and Spearmans correlation coefficient. Dynamic Time Warping distance measure has proved to be an accurate measure of dissimilarity for time series data [7] [8]. This is due to the fact that offsets in the shape of the curve do not affect DTW distance, unlike most other distance measures (such as Euclidean distance or Pearsons correlation). Further, by imposing a window size restriction during the calculation of the DTW, it can be made to select only patterns which are within a certain threshold of magnitude of phase shift. This also makes the DTW calculation more efficient [9]. Existing clustering algorithms suffer from various drawbacks. The standard k-means algorithm for clustering requires that the measure of similarity be a distance metric. DTW, however, is not a metric, and we can only obtain dissimilarities between two certain time series waves, not an absolute value for a wave (cannot calculate medoids). Several pure-density based clustering algorithms such as DBSCAN and OPTICS also prove to display suboptimal results with IoT data, due to the nature of the time series data. Time series data from IoT sensors almost always have subtle variations, even in repeating processes. Hence, these waves with these variations can act as a bridge between two classes. Hence, using a pure density-based algorithm for clustering yields large clusters which are generally a combination of two or more processes a user would like to identify individually. The clustering algorithm we have decided to employ uses density as a starting criterion for identification of clusters, followed by cluster refinement by set subtraction, and boundary point reassignment on the basis of distance from clusters. This method of clustering yields groups which have been clustered based on relative density of data points, which works especially well in capturing the essence of sensor data.

II. Related Work

A. Forecasting

The current various researches have used the method of forecasting with time series data such as the electric power consumption.

- **Conditional Restricted Boltzmann Machines and Factored Conditional Restricted Boltzmann Machines:** The Conditional Restricted Boltzmann Machine (CRBM) is a recently proposed model for time series that has a rich, distributed hidden state and permits simple, exact inference. It has an efficient, approximate learning algorithm called contrastive divergence [10]. However it performs poorly with null values, as the content must be clearly specified by the user.
- **Least Square Regression** Least Square Regression: Although being highly efficient, is unsuitable for datasets with non linear terms. This is because these terms curve relatively slowly, and for inherently non linear processes it becomes increasingly difficult to find a linear model that fits the data well as the range of the data increases.

B. Clustering

There are several algorithms that try to slice and cluster a time series. Purely density based approaches, such as DBSCAN and OPTICS exist [11]. However, these methods tend to cluster IoT data very loosely, and clusters obtained are not very pure. But most of these algorithms either rely on hard parameters or are not robust enough for noisy IoT data.

III. Our Contributions

Midhush Design and analysis of LSTM-RNN and extending it to a hybrid model of LSTM and SARIMA.

Naveen Preprocessing for time series clustering, slicing time series data, implementation of DTW distance measure needed for clustering and density based clustering of time series data.

Srikumar Cleaning and preprocessing of data, Analysis of time series data, forecasting of time series data using SARIMA and extending it to a hybrid model of LSTM and SARIMA. Analysis of the time series clusters.

IV. Our Approach

A. Forecasting

We have approached the problem of forecasting in three different ways:

1) SARIMA
2) LSTM-RNN
3) A Hybrid model that uses SARIMA and LSTM-RNN to model residuals

B. SARIMA

Chujai et al compared the performance of ARIMA and ARMA models in the forecasting of electricity [4]. SARIMA models data more accurately because it takes account of the seasonal pattern of data. While ARIMA considers all the points up-to a certain lag SARIMA extends this concept to include the lag terms which correspond to the time period of the season. The equations using backshift notation are:

SARIMA

$$(1-\phi_1 B)(1-\Phi_1 B^4)(1-B)(1-B^4)y_t = (1+\theta_1 B)(1+\theta_1 B^4)\epsilon_t$$

ARIMA

$$(1-\phi_1 B-...-\phi_p B^p)(1-B)^d y_t = c+(1+\theta_1 B+...+\theta_q B^q)\epsilon_t$$

The global active power readings sampled in one minute intervals were downsampled to intervals of day, week and month. The ACF for PACF were used to find the starting point of order and the seasonal order of the SARIMA model. Finer adjustments were made by applying a grid search on nearby values. The SARIMA model was used to forecast values in each of these intervals.

C. LSTM

The BDS test is a portmanteau test for time based dependence in a series [12]. It can be used for testing against a variety of possible deviations from independence including linear dependence, non-linear dependence, or chaos. The Null hypothesis checks for the linearity of data, whether it is independent and identically distributed. As the p-value for this test comes out to be lesser than 0.05 for daily, weekly and monthly data, it indicates that the chosen data is non-linear in nature, and it can be modelled with Neural Networks. Long short-term memory (LSTM), a class of recurrent neural networks (RNNs), is particularly designed for sequential data. It is an algorithm that is trained using Backpropagation through time, and overcomes the vanishing gradients problem [13] [14]. Instead of neurons, which are predominant in traditional neural networks, LSTM networks have memory blocks that are connected through layers. A block has components that make it smarter than a classical neuron and a memory for recent sequences. It contains gates that manage the blocks state and output, and operates upon an input sequence. Each gate within a block uses the sigmoid activation units to control whether they are triggered or not, making the change of state and addition of information flowing through the block conditional. There are three types of gates within a unit:

- **Forget Gate:** conditionally decides what information to throw away from the block.
- **Input Gate:** conditionally decides which values from the input to update the memory state.
- **Output Gate:** conditionally decides what to output based on input and the memory of the block.

Each unit is like a mini-state machine where the gates of the units have weights that are learned during the training procedure. This structure of LSTMs also makes it relatively insensitive to gaps in data, which gives it an upper hand over traditional neural networks.

In applied machine learning, we split our data into a train and a test set: the training set used to prepare the model and the test set used to evaluate it. We generally use k-fold cross validation that repeats this process by systematically splitting the data into k groups, each given a chance to be a held out model. These methods cannot be directly used with Time Series data. This is because they assume that there is no relationship between the observations, that each observation is independent. This is not true of time series data, where the time dimension of observations means that we cannot randomly split them into groups. Instead, we must split data up and respect the temporal order in which values were observed. Thus Walk Forward Validation is used. It consists of the following steps:

1) Starting at the beginning of the time series, the minimum number of samples in the window is used to train a model.
2) The model makes a prediction for the next time step.
3) The prediction is stored or evaluated against the known value.
4) The window is expanded to include the known value and the process is repeated (go to step 1.)

Because this methodology involves moving along the time series one-time step at a time, it is often called Walk Forward Testing or Walk Forward Validation. Additionally, because a sliding or expanding window is used to train a model, this method is also referred to as Rolling Window Analysis or a Rolling Forecast.

For the chosen dataset two different variants of the LSTM model has been used:

- **A simple Vanilla LSTM model**

```
model = Sequential()
model.add(LSTM(200, activation='relu',
input_shape=(n_timesteps, n_features),
bias_regularizer=L1L2(l1=0.01, l2=0.01)))
model.add(Dense(100, activation='relu'))
model.add(Dense(n_outputs))
model.compile(loss='mse', optimizer='adam')
```

- **A CNN Encoder-Decoder LSTM**

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3,
activation='relu',
input_shape=(n_timesteps,n_features)))
model.add(Conv1D(filters=64, kernel_size=3,
activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(RepeatVector(n_outputs))
model.add(LSTM(200, activation='relu',
return_sequences=True,
bias_regularizer=L1L2(l1=0.01, l2=0.01)))
model.add(TimeDistributed(Dense(100,
activation='relu')))
model.add(TimeDistributed(Dense(1)))
model.compile(loss='mse', optimizer='adam')
```

The encoder-decoder architecture uses two separate LSTMs, the encoder to read and encode the input sequence, and the decoder that will read the encoded input sequence and make a one-step prediction for each element in the output sequence. The important difference is that an LSTM model is used in the decoder, allowing it to both know what was predicted for the prior day in the sequence and accumulate internal state while outputting the sequence. A convolutional neural network, or CNN, is used as the encoder in the CNN-LSTM architecture. The CNN does not directly support sequence input; instead, a 1D CNN is capable of reading across sequence input and automatically learning the salient features. These can then be interpreted by an LSTM decoder as per normal.

D. Hybrid

Regression models are used to model data that is linear. They cannot model non-linear data as effectively. On the other hand LSTM-RNNs are very good at modeling non-linear data.

However LSTM-RNNs require large amounts of data in order to forecast accurately. Hence LSTM-RNNs are used here to complement the SARIMA model.

Zhang et al used feed forward ANNs to complement ARIMA by modeling the residuals [15].

The LSTMs in this case are used to model the residuals to enhance the prediction accuracy of the existing SARIMA model.

E. Clustering

The clustering algorithm by itself requires the following parameters:

1) Distance Matrix, containing the distances between different wavelets in the dataset (Δ)
2) Cluster Cut-off Size (λ)
3) Eps Value (ϵ)

The cluster cut-off size is the minimum number of motifs that a cluster should contain. It is dependent on the volume of data being clustered and can hence be heuristically determined for most purposes. The Eps value, , is used in the algorithm to determine if a point belongs to anothers vicinity or not, i.e. for data points A and B,

$$A \in \mu_B, \Delta_{AB} \geq \epsilon \tag{1}$$

where ϵ_X is the set of points in the vicinity of X. This is used to calculate initial densities of points, as specified by the algorithm. A similar parameter is expected as an input to DBSCAN, which purely clusters points based on the density of points on the hyperplane. The algorithm can be divided into three parts:

1) Pure cluster formation
2) Reassignment of points from clusters that do not satisfy the cluster cut-off size

Pure density clusters are points which are objectively clustered based on a vicinity based upon the EPS value. This is done by considering every point in the space as a centroid. These clusters will have very similar points, but the points may belong to multiple clusters.

Initially, fuzzy, pure clusters are formed. Here, each point in the dataset acts as a cluster centre and all points within an EPS value radius of a point is considered a part of the corresponding cluster. Hence, a point may belong to multiple clusters, but only N total clusters exist. If a point has less than the cluster cut off size number of points in its EPS value vicinity, it is also marked as an anomaly. The clusters are then be sorted in descending order by length. Now, an iterative set-subtraction is applied on the clusters, wherein the smaller clusters are replaced with the set difference between the smaller cluster and the larger one. This removes points which exist in multiple clusters from the smaller of the clusters, leaving the larger clusters large while making the smaller clusters further smaller. This is demonstrated in Algorithm 1. This creates the pure clusters. The clusters formed by the anomalous points as the cluster centres are now removed from the list of pure clusters. These pure clusters are further refined by removing border points from each cluster and re-assigning them to the nearest cluster. A border point j belonging to cluster i is one which meets the following criterion:

Algorithm 1 Pure Cluster Formation and Marking of Anomalies

Input: Distance Matrix Δ[N], EPS value ϵ, Cluster cut-off size λ
Output: Pure Clusters C

Initialisation :
C, densityBuffer
Initial Fuzzy Cluster Formation
for $i = 1$ to N **do**
  for $j = 1$ to N **do**
    if ($\Delta[i][j] \leq \epsilon$) **then**
      Append Point j to densityBuffer
    end if
  end for
  if (Length of densityBuffer $< \lambda$) **then**
    Mark Point i as Anomaly
  end if
  Append densityBuffer to C
  Clear densityBuffer
end for
Sort C in Descending Order of Length
for $i = 1$ to N **do**
  for $j = i$ to N **do**
    C[j] = set(C[j]) - set(c[i]) //set subtraction of larger cluster from smaller one
  end for
end for
Remove Empty Clusters from C
return C

$$D_{mean} = \text{Mean inter-cluster distance of Ci}$$

$$C_{mean} = \text{Mean distance of point j to all other points in Ci}$$

$$D_{mean} \leq C_{mean}$$

All the pure clusters obtained no longer contain any points that exist in more than one cluster. However, there could be clusters present which do not meet the cutoff-size constraint. For this, cluster reassignment is done. All points in clusters which do not meet the cluster cut-off size are reassigned to the closest cluster to the point. Here, the average distance between the point and the members of the cluster are used to determine the closest cluster to the point. This process is outlined in Algorithm 2.

V. Results

A. Analysis

1) BDS test: The BDS test performed on these intervals gave the following results.

p(daily)	3.67 * 10-29
p(weekly)	7.01 * 10^{-5}
p(monthly)	9.86 * 10^{-7}

Algorithm 2 Point Reassignment

Input: Pure Clusters C, Cluster cut-off size λ, Distance Matrix Δ[N]

Output: Final Clusters $\hat{C}$ *Removal of Anomalies*

$\hat{C}$= C - $C.anomalies$
for $i = 1$ to len($\hat{(C)}$) **do**
  pairwiseDist = array of all distance between all pairs of points in c_i
  for $j = 1$ to $len(c_i)$ **do**
    if $(\mu(\Delta[i, j]) > \mu(pairwiseDist))$ **then**
      Append Point j to $\hat{C}$ and remove point from c_i
    end if
  end for
end for
sort $\hat{C}$ in descending order
remove empty clusters from $\hat{C}$
for $i = 0$ to len($\hat{C}$) **do**
  if len(c_i) ¡ λ **then**
    for $j = 0$ to len(c_i) **do**
      Point j is appended to closest cluster that is above cluster cut-off size, removed from $\hat{C}$
    end for
    sort $\hat{C}$ in descending order
  end if
end for
return $\hat{C}$

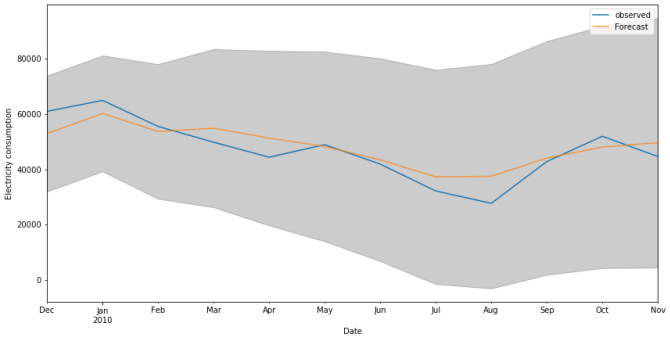

Fig. 1. Monthly Data

B. Forecasting

1) SARIMA: Results.

The model forecasted values from 06-12-2009 to 02-01-2010

	AIC	Train RMSE	Test RMSE
Daily	16357.181	459.899	424.600
Weekly	1888.728	2377.935	2112.522
Monthly	472.195	16634.531	5233.018

C. LSTM

	RMSE
LSTM	446.994
LSTM-CNN	432.906

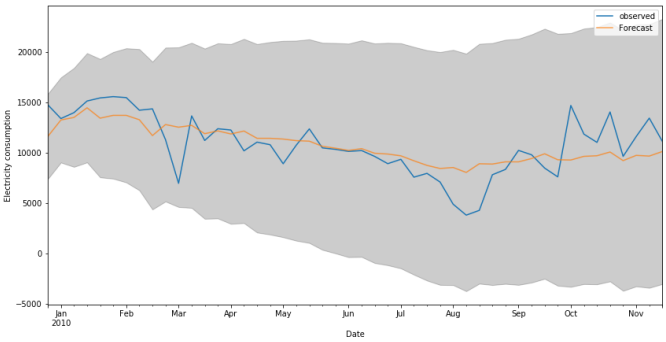

Fig. 2. Weekly Data

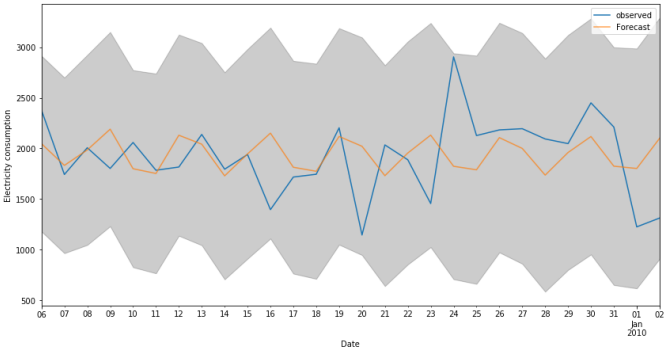

Fig. 3. Daily Data

1) Hybrid Model: The hybrid model acheived a testing RMSE of 421.23 which is marginally better than that of the SARIMA model's testing RMSE of 424.60 signifying an improvement in the model's performance due to residual modeling by LSTM.

D. Clustering

To test clustering, z-normalized slices of window size two hours was taken for an entire month, with a one hour overlap between slices. The distance matrix which contains the DTW

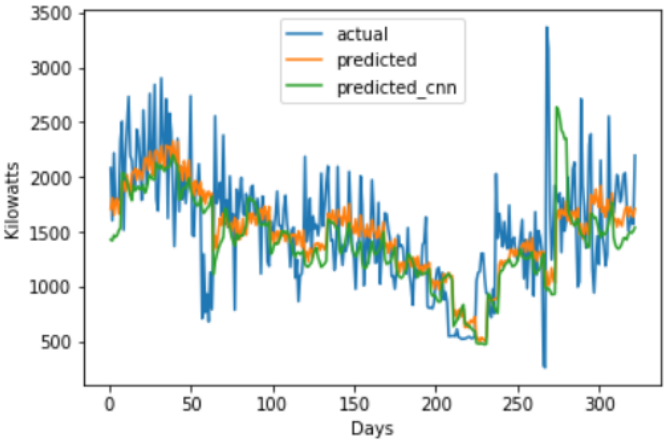

Fig. 4. Pure LSTM Test Results

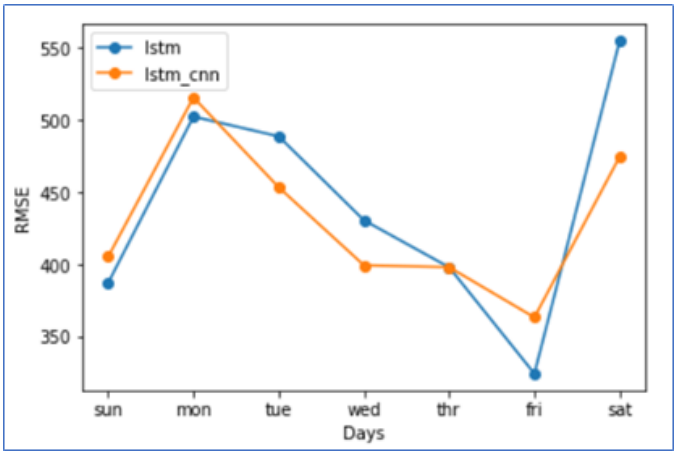

Fig. 5. Predictions for a week

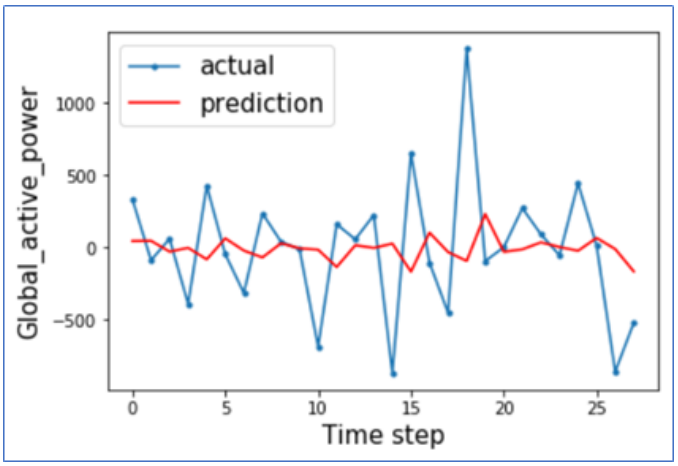

Fig. 6. Residual Predictions from Hybrid Model

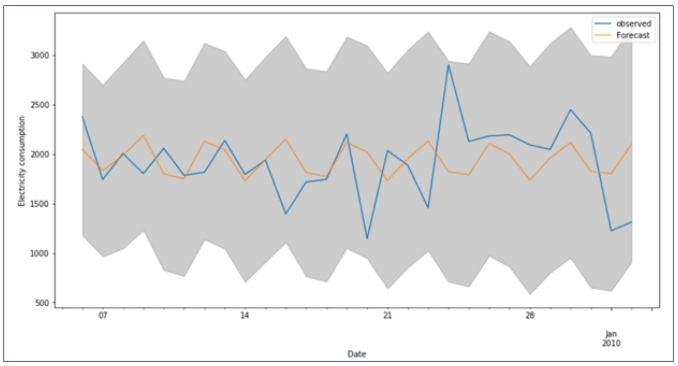

Fig. 7. Hybrid Model Prediction

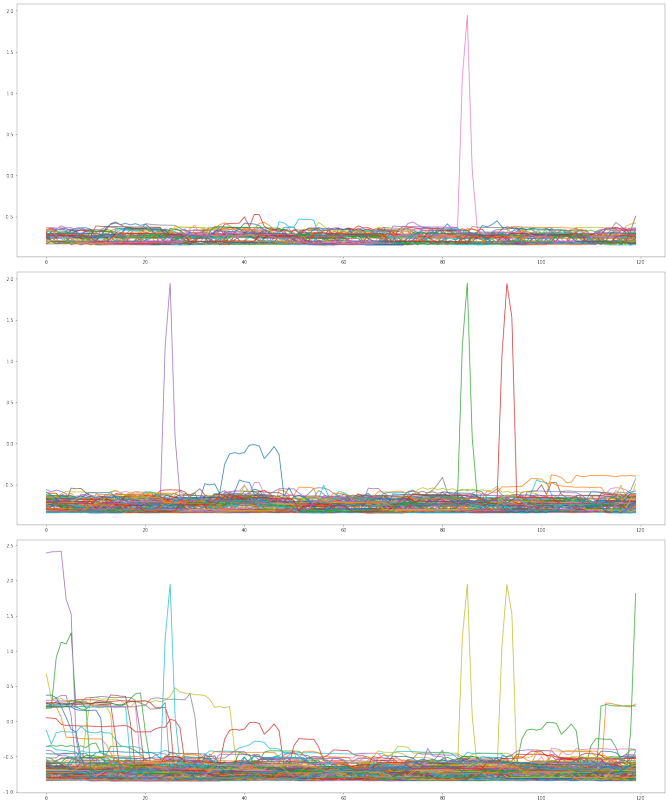

Fig. 8. Cluster obtained with $\epsilon = 0.025$, $\epsilon = 0.05$, and $\epsilon = 0.1$ respectively

distance between each pair of slices was computed. The distance matrix was fed into the clustering algorithm with various Eps values. fig. 1 represents the largest cluster obtained with Eps values 0.025, 0.05 and 0.1 respectively.

The same cluster becomes less and less refined with increasing Eps value. This happens as more slices fail to meet the cut-off distance criteria, and are removed from the clustering process. Hence, a Eps = 0.025 is used for clustering.

The cluster shown in the fig 2. further displays the effectiveness of the clustering algorithm with phase-differenced patterns.

Further, the cluster represented in fig 3. is one where the recurring patterns occurred at nearly the same time of day for the month, at 4 am. This information could be used by the electricity company to allocate extra blocks of power in this particular grid at 4am.

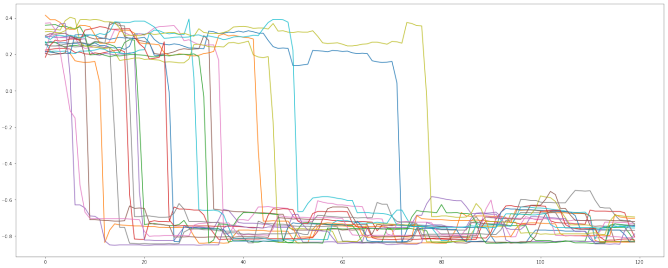

Fig. 9. Phase Differenced Slices in One Cluster

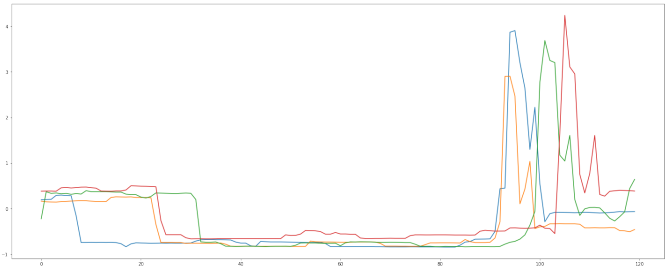

Fig. 10. Cluster occuring consistently at 4 in the morning

REFERENCES

[1] B. Young Choi, Jae Lee, "Combining lstm network ensemble via adaptive weighting for improved time series forecasting," 2018.

[2] Z. . Y. X. . C. H. . A. K. . F. K. Guo, Tian Xu, "Robust online time series prediction with recurrent neural networks," 2016.

[3] S. Saab, E. Badr, and G. Nasr, "Univariate modeling and forecasting of energy consumption: the case of electricity in lebanon," 2001.

[4] P. Chujai, N. Kerdprasop, and K. Kerdprasop, "Time series analysis of household electric consumption with arima and arma models."

[5] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," 1992.

[6] D. Yarotsky, "Optimal approximation of continuous functions by very deep relu networks," 2018.

[7] A. Mueen and E. Keogh, "Extracting optimal performance from dynamic time warping," 2016.

[8] K. W. Iglesias F, "Analysis of similarity measures in times series clustering for the discovery of building energy patterns," 2016.

[9] C. Ratanamahatana and E. J. Keogh, "Making time-series classification more accurate using learned constraints," 2004.

[10] G. W. Taylor and G. E. Hinton, "Factored conditional restricted boltzmann machines for modeling motion style," 2009.

[11] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," June 1999.

[12] O. J. O. O. G. A. kintunde, M. O., "Detection of non-linearity in the time series using bds test," 2015.

[13] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," 2015.

[14] F. B. Hasim Sak, Andrew Senior, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," 2014.

[15] P. Zhang, "Time series forecasting using a hybrid arima and neural network model," 2003.