ANOMALY DETECTION IN TIME SERIES DATA

PYTHON REPORT

RAGHAV RATHI (RR17D)
TEJAS HASARALI (TDH17)
SPRING 2018

Introduction:

Time series is a series of data points listed in the order of its occurrence. In our project we are using number of people visited the shop in a day collected through Wi-Fi ping. The router even counts the passer by as people visited, Hence the project was under taken to find the anomalies in the obtained data by finding the pattern in the number of people visited. Auto encoders are the artificial neural networks used in unsupervised learning techniques. Auto encoders learns the data using unsupervised learning technique by compressing the high dimensionality data in to low dimension and expanding back to the original dimension. The expanded data is compared to the original data to find the loss, model learns the data by reducing this loss in each epoch. A variant of auto encoder which makes strong assumptions on distribution of latent variables is variational auto encoder. It uses variational approach for representing latent space with an additional loss component and specific training algorithm Stochastic Gradient Variational Bayes. The difference between auto encoder and variational auto encoder is that, in auto encoder the vectors in latent space is simply passed through the decoder to generate the original data, but in case of variational auto encoder additional latent vectors are generated which follows unit Gaussian distribution and passed to the decoding layer to generate the original data. Since both the models are unsupervised learning techniques, they can be used in detecting anomalies.

The artificial neural network for auto encoders can be built using various techniques like Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). CNN is a class of deep and artificial neural network which is majorly used in analyzing images whereas LSTM as the name suggests, the neuron remembers the data for an arbitrary interval of time. It is mainly used in natural language processing and speech recognition. CNN requires minimal preprocessing as it uses multilayer perceptron, it has a shared weights architecture and has a translation invariance characteristic. CNN consists of input layer, output layer and multiple hidden layers, hidden layers mainly consists of convolutional layers, pooling layers, fully connected layers and normalized layers. LSTM are the building blocks of Recurrent Neural Network, it consists of memory cell, input gate, output gate and forget gate. Memory cell stores the value either for a long term or short term, input gate controls the extent new data inflow to the cell, forget gate control the extent to which it is remembered, and the output gate controls the extent of data used for output activation computation. In our project we have used both CNN and LSTM as the building blocks of auto and variational auto encoders. We have built two models of variational auto encoders and two models of auto encoders using CNN and LSTM. We have found that, variational auto encoder with 3 hidden LSTM artificial neural network layers perform better compared to other three models.

Literature Survey:

The paper [1] tries to find a way to detect anomalies for seasonal KPIs with various patterns and data quality and the solution they came up with is a model named 'Donut', which is an unsupervised anomaly detection algorithm based no VAE. The KPI (key performance indicators) used in this project are time series data measuring the page views number of online users and number of orders, which is kind of similar to the data we used in our project. One of the import thing they found out was that, adopting VAE for anomaly detection requires training on both normal and abnormal data. So, they trained their donut model on both the normal data and abnormal data which made it more accurate. They used three techniques in Donut, modified ELBO, missing data injection for training and MCMC (Markov chain Monte Carlo) imputation for detection, which gave their model an edge over the existing VAE based models. ELBO or evidence lower bound. In missing data injection, is basically making all the missing data as 0, as it not possible to fill the missing data accurately to make it normalized and this is a necessary process before the imputation. MCMC (Markov chain Monte Carlo) is a technique for estimating by simulation the expectation of a statistic in a complex model. They have used the above three techniques to build, train and improve their model which can find out the anomalies. This approach could not be used, since collecting the ground truth is difficult in our case.

Auto encoder is a neural network which works on the principle of unsupervised learning. Unsupervised learning is learning on uncharacterized data. In a general way, they work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. The auto encoder tries to learn a function hW, b(x) ≈ x. In other words, it tries to learn an approximation to the identity function, so as to output x' that is similar to x. Also auto encoders are data specific, which means they can only compress similar data to which they were trained on. To build an auto encoder, you need an encoding function, a decoding function, and a loss function which in simple words is a distance function between the amount of information loss between the compressed representation of your data and the decompressed representation. The encoder works by compressing or down sampling the input into a fewer number of bits and this is called encoding. The decoder works by reconstructing the original input from the encoded data. Just like regular Auto encoders, VAEs try to reconstruct output from input and consist of an encoder and a decoder, which are encoding and decoding the data. The encoder outputs a compressed representation of the output data. The decoder then learns to reconstruct the initial input data by taking this compressed representation as input data. This way, you can use the decoder as a generative model which is able to generate specific features—such as specific digits or letters**.** The main difference between variational auto encoder and auto encoder is that it generates similar data in the latent space which follows the Gaussian distribution function.

The auto encoders and variational auto encoders can be built using the Keras library. Keras is an open source neural network library written in python. We used several other modules like pandas to read and parse the data, we used skit-learner's train test split module to divide the entire data into training data and test data but then we found out that it was sampling the data randomly, so we divided the data manually, since the order is important in time series data. Two neural networks were used to build the auto encoders, Long Short-term Memory and Convolutional Neural Network. A convolutional Neural Network(CNN) is an artificial neural network which is best used for image analysis, they can also be used for other data analysis and classification uses like we tried in our project. It works by analyzing the given input and then picking out a pattern, that's why it is so efficient in image analysis. It contains hidden convolutional layers which receives and transforms the input and pass it to the other layers. Whereas LSTM are the building blocks of Recurrent Neural Network, it consists of memory cell, input gate, output gate and forget gate. Memory cell stores the value either for a long term or short term, input gate controls the extent new data inflow to the cell, forget gate control the extent to which it is remembered, and the output gate controls the extent of data used for output activation computation.

Various optimizers were used to improve the results, some of them are RMSprop, Adam, Nadam and Adamax. Adamax and RMSprop gave the best results, but RMSprop turned out to be the better optimizer. The number of layers in the neural network was defined arbitrarily since there is no thumb rule as to how many layers and number of cells has to be defined for obtaining accurate results.

Design:

The obtained data consists of date and the number of people visited the shop on that day for almost a period of two years. The data was analyzed to find the number of null values, but no null values was found. Day, Month and Day of the week was extracted from the formatted date string. Dummy variables for day, month and day of the week is extracted by creating separate columns for all the values. The date column was then dropped as the number of people do not show any increasing or decreasing trend as the time progressed. This we can see in the figure 1, which plots the number of people visited against time.

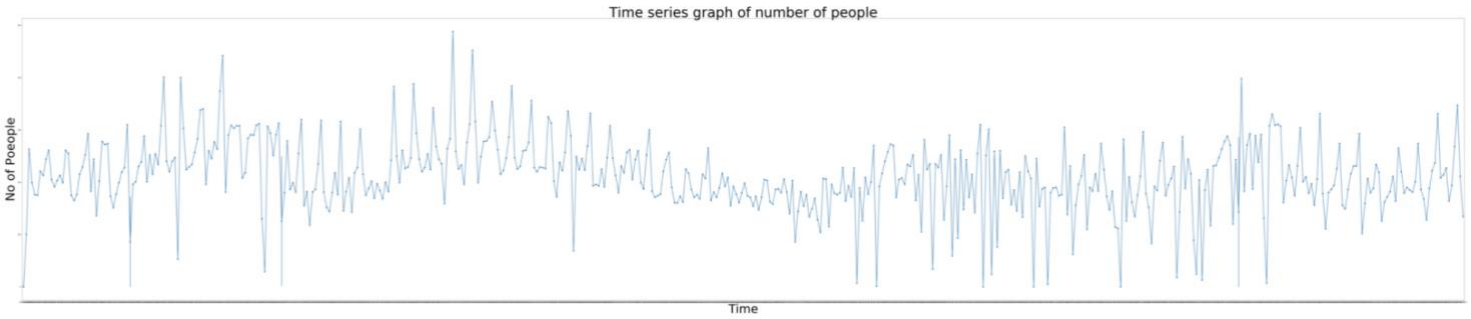

Figure 1. Time Series graph of number of people visited

The data is normalized to fit in the range of 0 to 1, this is done to give equal importance to all the features used in training the neural network. Since the number of people visited reach values greater than 400, the neural network will give more importance to this feature and the output will completely depend on this. Hence normalizing the data is crucial. The processed data is split into training and testing, since it is time series data the order is very important. Using the skit-learner test train split module will divide the data randomly this will affect the output of the model. Hence the data is divided manually by splitting the first 80% as training and the remaining 20% as testing. We trained four models variational auto encoder with LSTM, variational auto encoder with CNN, auto encoder with LSTM and auto encoder with CNN. The parameters of each model were changed using trial and error method, the model with best output was selected.

- **Auto Encoder with CNN:** Deep auto encoder with three hidden layers, one input layer, one latent layer and one output was built as shown in the figure 2. Here input_1 with input dimension 51 is the input layer, dense_1 to dense_3 are the encoding layers, dense_4 is the latent layer and dense_5 to dense_8 are the decoding the layer. Except input and output layer all the layers have 'relu' as their activation function and for the output layer sigmoid is used as activation function. The model is trained for 100 epochs by dividing the data into batch size of 100 with 'root mean square' optimizer and 'mean squared error' loss functions. The number of layers, optimizer and loss functions are selected after extensive trial and error operation. The model loss reduces from 0.2425 to 0.0456 in just 100 epochs. The prediction made based on the training data on the testing data is shown in the figure 3. As we can see the model does not follow the trend followed by the actual test data. This shows that the auto encoder with CNN fails to learn any useful information from the training data even though the loss is very low.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 51)	0
dense_1 (Dense)	(None, 38)	1976
dense_2 (Dense)	(None, 28)	1092
dense_3 (Dense)	(None, 21)	609
dense_4 (Dense)	(None, 2)	44
dense_5 (Dense)	(None, 21)	63
dense_6 (Dense)	(None, 28)	616
dense_7 (Dense)	(None, 38)	1102
dense_8 (Dense)	(None, 51)	1989

Total params: 7,491
Trainable params: 7,491
Non-trainable params: 0

Figure 2. Summary of the auto encoder with CNN neural network built

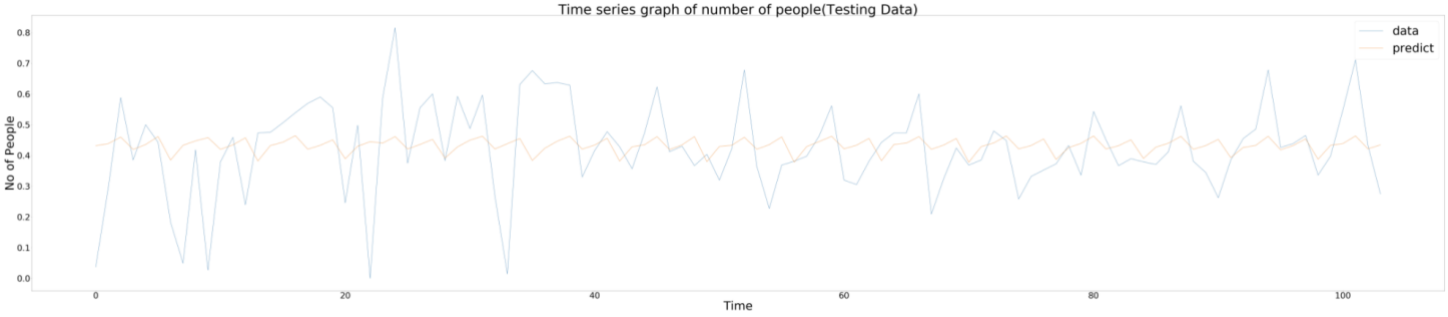

Figure 3. Time series graph of actual test data and prediction on the test data

- **Auto Encoder with LSTM:** The LSTM neural network expects the input to be three-dimensional, second dimension being the number of time steps or in other words the number of data the neuron cell has to remember. In our model we choose one as the time step as it was giving the best results. Similar to auto encoder with CNN we used three hidden layers with one input layer, one output layer and one latent layer. As shown in the figure 4 lstm_1 to lstm_4 represents the encoder layer, lstm_4 is the latent layer, lstm_5 to lstm_7 are the decoder layer and lstm_8 is the output layer. The activation function of all the LSTM layers are 'tanh'. We also have an additional repeat vector layer to convert the two-dimensional data in the latent space into three-dimensional data for LSTM input. The model is trained just for 25 epochs with loss reducing from 0.0571 to 0.399. Similar to auto encoder with CNN we use 'root mean square' optimizer, 'mean squared error' loss function and the model is trained in the batch size of 1. The prediction made based on the learning from the training data along with the actual test data is shown in the figure 5. We can see from the figure 5 that the model is following one particular pattern rather than learning the pattern based on different days, day of the weeks or the months, it looks like irrespective of any changes in other features the model is following one particular pattern.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 1, 51)	0
lstm_1 (LSTM)	(None, 1, 38)	13680
lstm_2 (LSTM)	(None, 1, 28)	7504
lstm_3 (LSTM)	(None, 1, 21)	4200
lstm_4 (LSTM)	(None, 2)	192
repeat_vector_1 (RepeatVecto	(None, 1, 2)	0
lstm_5 (LSTM)	(None, 1, 21)	2016
lstm_6 (LSTM)	(None, 1, 28)	5600
lstm_7 (LSTM)	(None, 1, 38)	10184
lstm_8 (LSTM)	(None, 1, 51)	18360

Total params: 61,736
Trainable params: 61,736
Non-trainable params: 0

Figure 4. Summary of the Auto encoder with LSTM neural network

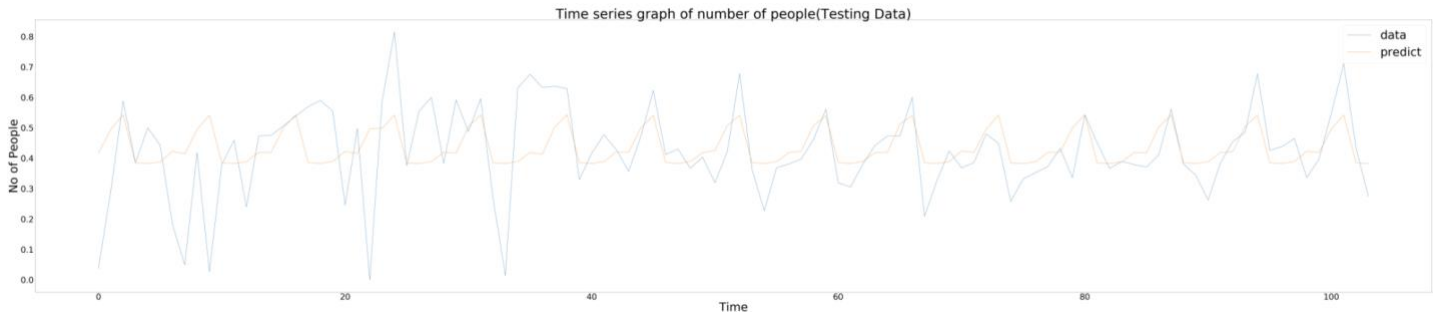

Figure 5. Time series graph of actual test data and the prediction based on the test data

- **Variational Auto Encoder with CNN:** Variational auto encoder differs slightly from the auto encoder, it generates similar data which follows the simple Gaussian gradient function in the latent layer and adds additional loss to compensate the generation of new data. We built a deep variational auto encoder with one input layer, three hidden layers and one output layer. As we can see in the figure 6 dense_1 to dense_3 represents the encoder layer, dense_6 to dense_8 represents the decoder layer, input_1 is the input layer and the dense_9 is the output layer. Dense_4 and dense_5 represent the latent layers which is responsible for sampling and regeneration of data, the output of these two layers are combined in lambda_1 layer after adding the loss. All the dense layers use 'relu' as the activation function. The model is trained in a batch of 100 for 100 epochs with 'root mean squared' as optimizer. The loss reduces from 35.0076 to 11.3966 in just 100 epochs. The figure 7 shows the prediction made using the trained model and the actual test data, as we can see the model hardly learned anything from the training data and the prediction almost follow a straight line.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 51)	0	
dense_1 (Dense)	(None, 38)	1976	input_1[0][0]
dense_2 (Dense)	(None, 28)	1092	dense_1[0][0]
dense_3 (Dense)	(None, 21)	609	dense_2[0][0]
dense_4 (Dense)	(None, 2)	44	dense_3[0][0]
dense_5 (Dense)	(None, 2)	44	dense_3[0][0]
lambda_1 (Lambda)	(None, 2)	0	dense_4[0][0] dense_5[0][0]
dense_6 (Dense)	(None, 21)	63	lambda_1[0][0]
dense_7 (Dense)	(None, 28)	616	dense_6[0][0]
dense_8 (Dense)	(None, 38)	1102	dense_7[0][0]
dense_9 (Dense)	(None, 51)	1989	dense_8[0][0]

Total params: 7,535
Trainable params: 7,535
Non-trainable params: 0

Figure 6. Summary of Variational Auto Encoder with CNN

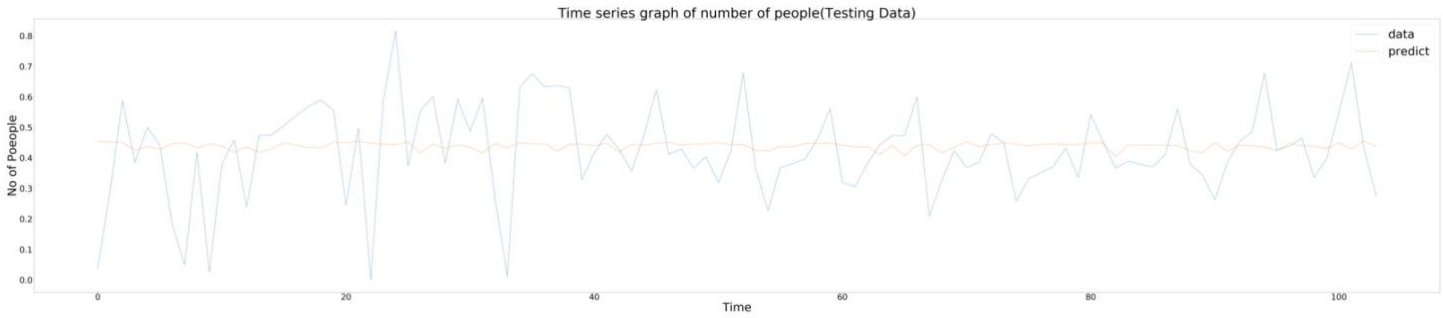

Figure 7. Time series graph of actual test data and the prediction based on the test data

- **Variational Auto Encoder with LSTM:** Deep variational auto encoder with three hidden layers, one input layer and one output layer performed the best with number cells shown in the second column of the figure 8. As shown in the figure lstm_1 to lstm_3 represents encoder layer, lstm_4 to lstm_6 represents the decoder layer, input_1 is the input layer and lstm_7 is the output layer. Similar to variational auto encoder with CNN it uses two dense layers for sampling and data generation, one lambda layer for combining and adding loss. We also use repeat vector layer to convert the two-dimensional data in the latent layer to three-dimension for LSTM input. We run the model for 100 epochs with 'root mean squared error' as the optimizer and the loss reduce from 0.0625 to 0.0530. LSTM layers use 'tanh' as the activation function and CNN layers use 'relu' as the activation function. Figure 9 shows the prediction based on the training data and the actual test data, we can see that the prediction follows almost the same path as the actual test data. Hence for this combination of parameters the variational auto encoder with LSTM performs better compared all other models.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1, 51)	0	
lstm_1 (LSTM)	(None, 1, 38)	13680	input_1[0][0]
lstm_2 (LSTM)	(None, 1, 28)	7504	lstm_1[0][0]
lstm_3 (LSTM)	(None, 21)	4200	lstm_2[0][0]
dense_1 (Dense)	(None, 2)	44	lstm_3[0][0]
dense_2 (Dense)	(None, 2)	44	lstm_3[0][0]
lambda_1 (Lambda)	(None, 2)	0	dense_1[0][0] dense_2[0][0]
repeat_vector_1 (RepeatVector)	(None, 1, 2)	0	lambda_1[0][0]
lstm_4 (LSTM)	(None, 1, 21)	2016	repeat_vector_1[0][0]
lstm_5 (LSTM)	(None, 1, 28)	5600	lstm_4[0][0]
lstm_6 (LSTM)	(None, 1, 38)	10184	lstm_5[0][0]
lstm_7 (LSTM)	(None, 1, 51)	18360	lstm_6[0][0]

Total params: 61,632
Trainable params: 61,632
Non-trainable params: 0

Figure 8. Summary of Variational Auto Encoder with LSTM

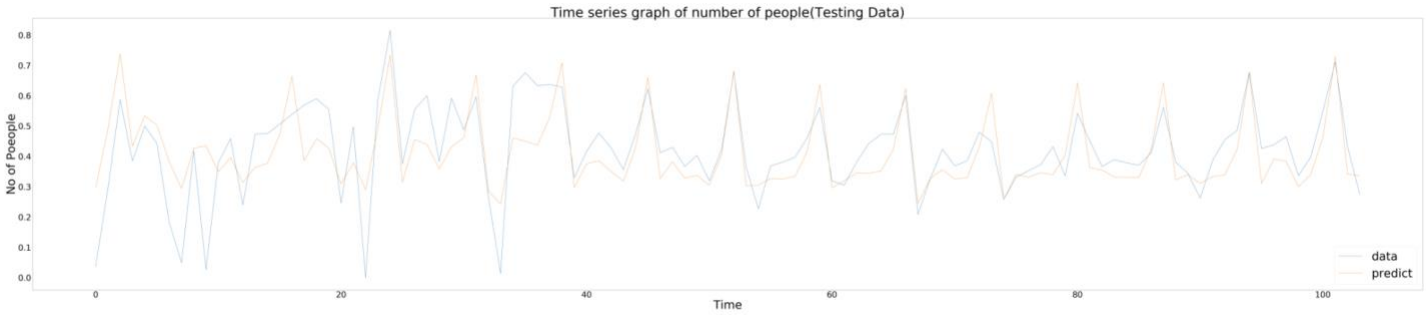

Figure 9. Time series graph of actual test data and the prediction based on the test data

Conclusion & Future Work: Variational auto encoder with LSTM performs better compared to all other models for the given dataset. The prediction follows almost the same path as the actual data. The parameters of the model were defined by performing many trails and comparison of the results. We built a model that is capable of eliminating the anomalies by detecting pattern in the given dataset and predicting the number of people visiting in the future. The model can also be used to generate similar data and representing the 51-dimension data in two-dimensional space after encoding.

The model can be further improved by performing more experiments on different parameters. If the data was available for every hour, the model could be used for predicting the busiest hours in a particular day.

References:

[1] Unsupervised Anomaly Detection Via Variational Auto-Encoder for Seasonal KPIs In Web Applications
[2] https://www.youtube.com/watch?v=umGJ30-15_A
[3] https://www.youtube.com/watch?v=aircAruvnKk
[4] https://www.youtube.com/watch?v=9zKuYvjFFS8
[5] https://blog.keras.io/building-autoencoders-in-keras.html
[6] https://cs231n.github.io/
[7] https://github.com/vdumoulin/conv_arithmetic
[8] https://en.wikipedia.org/wiki/Feedforward_neural_network
[9] https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
[10] http://philipperemy.github.io/anomaly-detection/
[11] https://medium.com/@curiousily/credit-card-fraud-detection-using-autoencoders-in-keras-tensorflow-for-hackers-part-vii-20e0c85301bd
[12] https://www.sparkcognition.com/2017/09/more-effective-approach-unsupervised-learning-time-series-data/
[13] https://arxiv.org/pdf/1802.03903.pdf
[14] https://blog.statsbot.co/time-series-anomaly-detection-algorithms-1cef5519aef2
[15] https://en.wikipedia.org/wiki/Multilayer_perceptron
[16] https://www.quora.com/Whats-the-difference-between-a-Variational-Autoencoder-VAE-and-an-Autoencoder
[17] https://github.com/cheng6076/Variational-LSTM-Autoencoder
[18] https://codeburst.io/deep-learning-types-and-autoencoders-a40ee6754663