Anomaly Detection for Time Series Data using VAE-LSTM Model

Randall Fowler
Department of ECE
UC Davis
rlfowler@ucdavis.edu

Conor King
Department of ECE
UC Davis
cfking@ucdavis.edu

Ajay Suresh
Department of ECE
UC Davis
ajsuresh@ucdavis.edu

***Abstract*—Here is an abstract...**

***Index Terms*—Anomaly Detection, VAE, LSTM, Time Series Data, IMU, PPG, EV Drive Cycle, Deep Learning**

I. Introduction

This is an introduction...

II. Methodology

Anomaly detection is represented by measuring the reconstruction error from a given deep learning model. For this system, a VAE and LSTM hybrid model is used for finding a latent space with a known distribution and capturing temporal information over time series data. This data is presented in a windowed fashion to locate anomalies within a given window. Evaluation of the anomaly detection will be measured using the harmonic mean of precision and recall (F1 score).

A. Preprocessing

The nature of the data used for this detection system does not need to be specific, but it does require being time series data. With a given window size, a sliding window will sample the original signal, and the VAE input and output will be equivalent to the size of the window.

As this data does not need to be labeled, it does need to be clean of noise or anomalies. This is due to the model learning the clean distribution of the data, and after training, the poor reconstruction of a signal will indicate the possibility of an anomaly.

B. Variational Autoencoder

An autoencoder structure establishes a method for compressing information into a dense latent space. This space is not unique as the original signal can be encoded into a large number of different latent spaces. The VAE addresses this non-regularization issue of the latent space by forcing the space to follow a specific distribution [**?**]. For most VAEs, this distribution is a normal distribution, and the output of the encoder will be a prediction of mean and standard deviation. Once the distribution of the latent space is known, a sampling will be performed to create a latent vector for the input of the decoder. When the latent space distribution is well formed, the output of the decoder aims to reconstruct the original signal as best as possible.

As the autoencoder represents encoder and decoder models, these structures can be created in a variety of ways. For this project, both will be presented by 1-dimensional convolutional layers with a leaky ReLU activation function and batch normalization. The number of samples per window will decrease while depth increases for every layer in the encoder structure. At the end of the encoder, the samples and depth are flattened and passed to linear layers to condense to the final latent space mean and standard deviation values. Samples from the normal distribution will be given to a decoder that is identical to the encoder, but with transpose convolutional layers to increase the size and decrease the depth. Padding for each layer is forced to maintain a consistent linear decrease in the number of samples. For this architecture, four convolutional layers are used with a split set of linear layers for the mean and standard deviation prediction.

C. Long Short-Term Memory

With a sampled latent space created from the encoder, the LSTM takes the current latent vector and attempts to predict the next sample. If the VAE is trained well, it should hold the distribution for the dataset, and the LSTM should learn the temporal differences between each latent window. The LSTM tracks previous outcomes by utilizing memory variables or acting as a state machine. Cell memory and hidden state become the long-term and short-term aspects of the model, and those components carry into the model as inputs [**?**]. After each iteration, the memory variables will output to be used by another LSTM layer or dropped before other layers.

A standard LSTM layer contains a tangent activation function, and the PyTorch library does not provide an option to adjust the activation function [**?**]. For this reason and the lack of activation function post latent space sampling, a linear layer is placed after the final LSTM layer to adjust for scaling.

D. VAE-LSTM Model Training

Since the two models hold separate functions for the detection system, the training for the VAE and LSTM are handled separately. Both will use the same training set as the models will be combined after training. Batch size, loss function, and optimizer may vary between the two models.

Loss function can differ greatly as the VAE can be trained with Kullback-Leibler (KL) divergence. This loss relates to the similarity between the latent space distribution and the unit normal distribution [?]. Essentially, it enables the latent space to hold this normal distribution shape. Reconstruction loss is another term that focuses on training the weights regardless of the latent space. The sum of the similarity and reconstruction loss promotes a smooth and continuous latent space while improving signal reconstruction.

The LSTM only needs to focus on the error for prediction on known future samples. This error may be the same as reconstruction loss for the VAE. With the memory components, these need to be initialized properly when training. For each given batch, the batch needs to remain in sequential order, and the memory should be reset if batches are not in order. For predictions on an entire dataset, the memory should only be reset at the very beginning. When resetting, the cell memory and hidden state are initialized with a Normal Xavier Initialization [?]. This sets the weights using a normal distribution with a mean of 0 and a variance dependent on the number of inputs and outputs of a layer.

E. Anomaly Detection

After training, anomalies are detected using the reconstruction error of the predicted sample and thresholding this value. When an anomaly occurs, the model will predict the known distribution and notice the disruption after decoding. The VAE can detect anomalies without the LSTM, but it will be lacking the temporal aspects and recognize the anomaly on reconstruction error alone.

Since the original signal will be windowed prior to input, each window will have an error associated with predicting an anomaly. To represent the anomaly detection in the original signal rather than the windowed version, each data point in the window will be stored in its respective position. Each data point will be the average of all window errors that data point is utilized in. For the first data point in the original signal and first window, this error will be the error of the first window. The second data point will be the average error between the first window and second window. Once the last data point of the first window is reached, this point will be averaged over the same number of windows as the window size, and all following data points will be averaged the same until the final window is reached.

The purpose of averaging error over windows provides a smoother transition between several points where an anomaly would be. It is unlikely that a signal data point will hold an anomaly as typically, an anomaly will occur over a range of time. This issue is also addressed in the next section to increase the score if a section of an anomaly is detected.

F. Evaluation Metrics

Evaluation of the detection system is handled using an F1 score as this is a measure of precision over sensitivity [?]. Precision is the accuracy of positive predictions, and it is divided by the total positive predictions. Sensitivity, or recall, is the rate of true positives where the number of true positives is divided by the sum of true positives and false negatives. While accuracy of predictions may be a good metric, F1 score considers the distribution of predictions as it is the harmonic mean between the precision and recall.

Since anomalies typically occur in sections of a signal rather than pointwise, assessment per anomaly region could be a better method for measuring true positives. One strategy is to assume any anomaly segment in the ground truth that is detected would be considered a correct detection for that segment, and any point outside of that segment would be treated normally [?]. This method will be referred to as the augmented F1 scoring, and both the regular and augmented F1 score will be evaluated.

III. Experiments

Due to the restrained time, the exploration of hyperparameters, architecture, and optimization techniques were limited to first implementations.

The window size was set to be 100 samples of the original signal with a stride of 1. Each prediction by the LSTM is dedicated to a single window, but since the stride is one, this is a sample-by-sample prediction. Padding for each layer is forced to maintain a consistent linear decrease in the number of samples. Different window sizes should be explored to relate to the different range of time noticed by the model in each moment, but with the difference between signal size and convolutional layers, the exploration was neglected. Four convolutional layers are used in the encoder with a split set of linear layers for the mean and standard deviation prediction, and the decoder layers are identical to the encoder. In the latent space, the LSTM is also given 4 LSTM layers to capture a temporal distribution in the latent space. Number of parameters for the entire detection model was nearly 1.25 million parameters.

For the loss function of both models, mean square error (MSE) is selected for ease. Binary cross entropy (BCE) was considered for loss, but the input and output would be difficult to constrain between 0 and 1. One challenge related to training with similarity loss, KL divergence, as the calculated values did not seem appropriate. This is expected to be a small bug in the implementation, but for the sake of time, it was not used. Without the KL divergence, the VAE would resemble a standard autoencoder.

Training was conducted with varying batch sizes and number of epochs between different datasets, and an Adam optimizer with a learning rate of 4e-4 and betas as 0.9 and 0.95 was selected.

To explore the robust design of the detection model, three datasets are used for training and testing. IMU dataset consists of motion recordings from multiple sensors to detect movement or changes to the monitoring system. Synthesized PPG data is created from simulating tissue models and aims to detect motion, electromagnetic interference, and other variations of noise. *SOMETHING ABOUT AJAY'S DATA*

A. Inertial Measurement Unit Dataset

Here is the dataset...

B. Synthesized Photoplethysmography Dataset

Here is the dataset...

C. Electric Vehicle Drive Cycle Dataset

Here is the dataset...

IV. Results

Here are the results...

A. Inertial Measurement Unit

Here is the dataset...

B. Synthesized Photoplethysmography

Here is the dataset...

C. Electric Vehicle Drive Cycle

Here is the dataset...

V. Conclusion

Here is a conclusion...

A. Future Work

Here is some future work...

Acknowledgment

The authors would like to thank Dr. Yubei Chen for his guidance and support on this course project.

Contributions

Randall Fowler developed the models, evaluation methods, and experimented with the IMU dataset. Conor King experimented with the synthesized PPG dataset. Ajay Suresh experimented with the EV drive cycle dataset. All authors contributed to the writing of the paper.

References

[1] Citation 1
[2] Citation 2
[3] Citation 3
[4] Citation 4
[5] Citation 5