Google Cloud

ASL Tech Talk:
Time Series Anomaly Detection

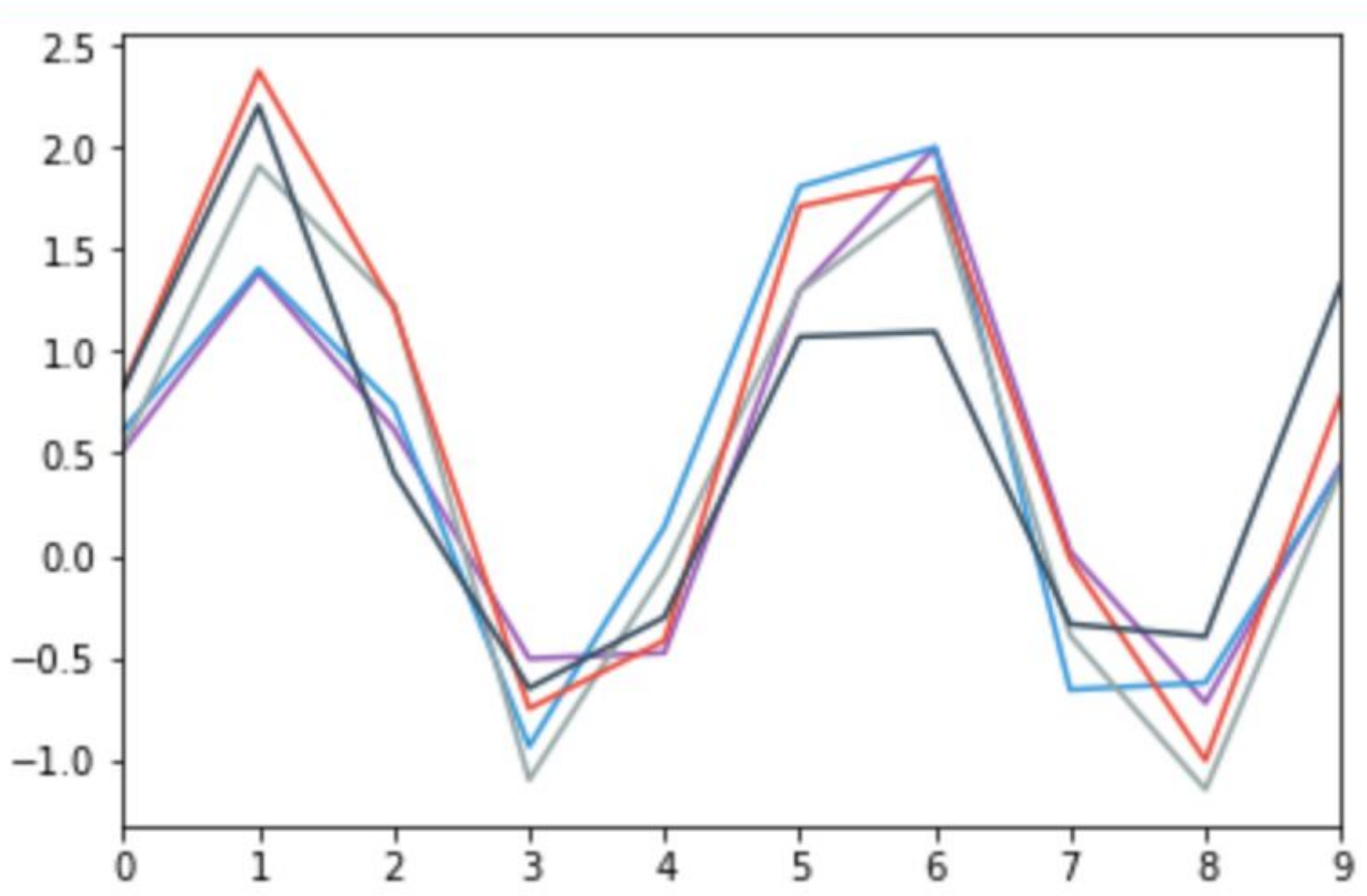
What is anomaly detection?

- Often times in data mining/analysis we want to be able to find outliers; rare events, occurrences, etc. that don't belong to our distribution of interest.
- Important for many industries such as banking fraud detection, IoT predictive maintenance, cybersecurity threat detection.

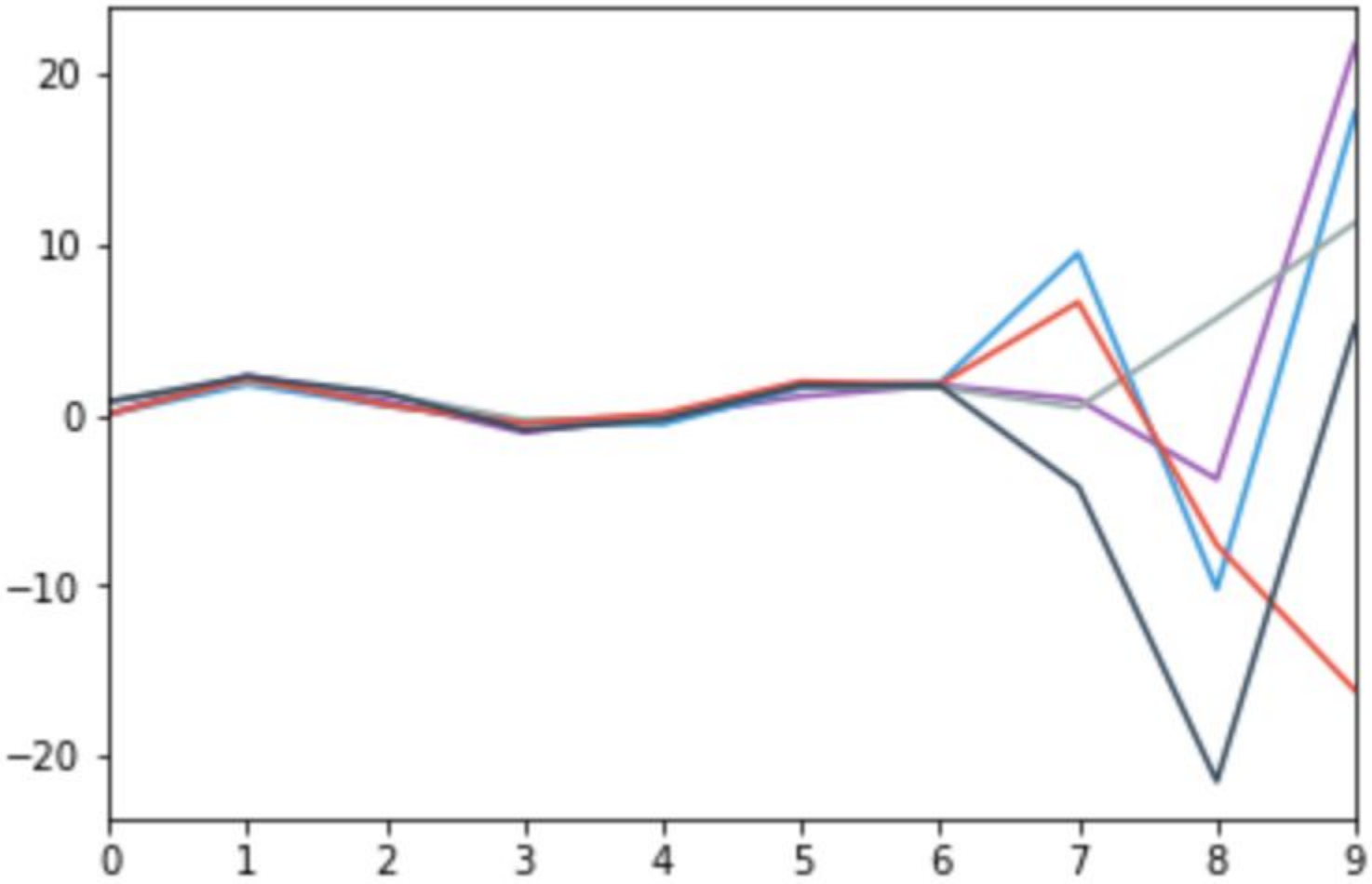
With time series?

- When most people think of anomaly detection, they first go to a point prediction.
- However, a point by itself may be very hard to flag.
- Often, it is a sequence of activity that together indicates an anomaly!
- Can look at this from both a time or feature major view.

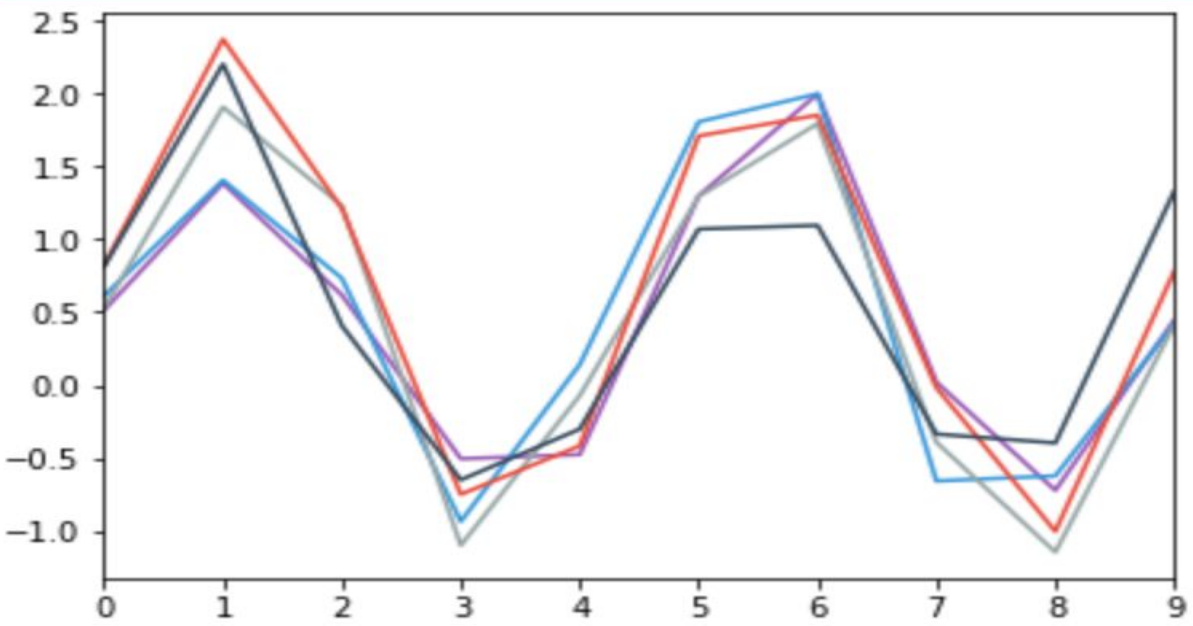
Is this anomalous?

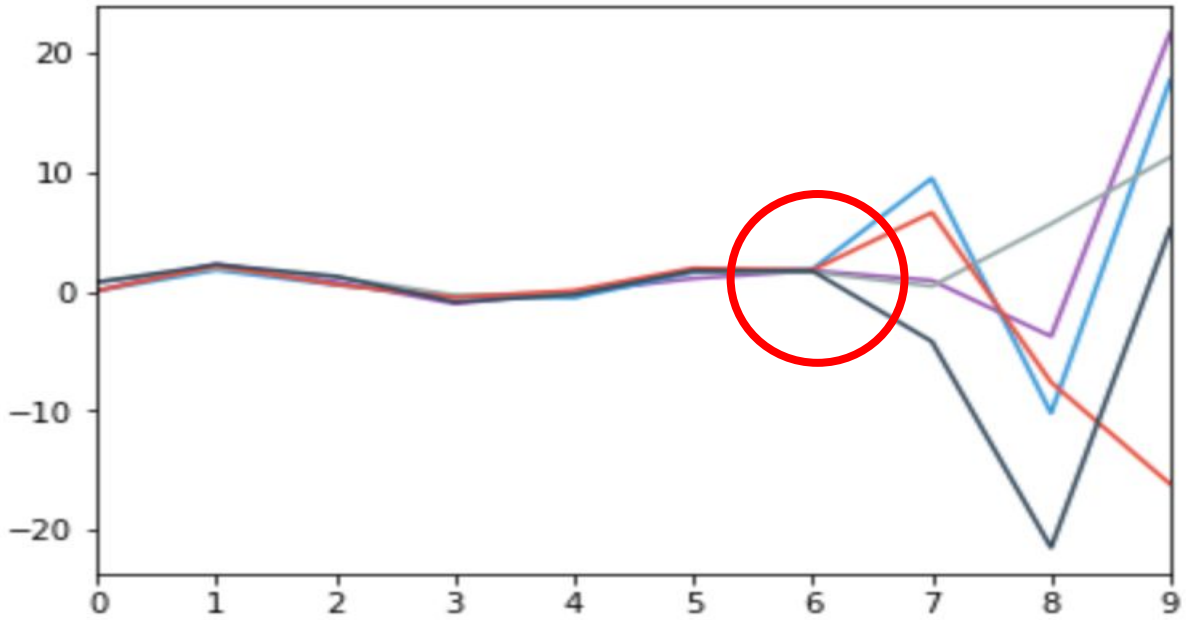


Is this anomalous?



Is this anomalous?

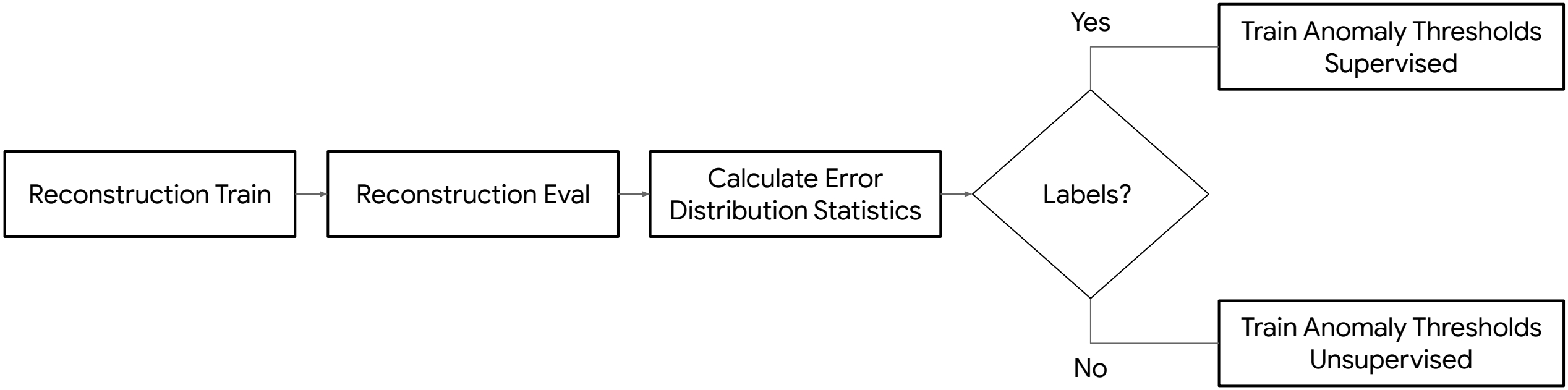




How to measure?

- There are many methods for time series anomalies detection.
- A common method is reconstruction where the predictions are the original inputs.
- We then calculate the distribution of errors so we run new data through it.
- Lastly, we tune anomaly thresholds so we can draw a line in the sand on what we're saying is what.

Training High-level System



LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection
https://arxiv.org/abs/1607.00148

Datasets

- Since there are multiple stages there are multiple datasets.

- sn1: Large num of norm seqs to be used in reconstruction training.
- vn1: Smaller num of norm seqs to be used for reconstruction evaluation and calculating error distributions.
- vn2/va: Smaller num of norm and anom seqs mixed together.
- tn/ta: Smaller num of norm and anom seqs mixed together.

Reconstruction Training

- The main goal for reconstruction is to bottleneck the inputs through the system so that a compressed representation will be learned (and not the identity function).
- Can do this with autoencoders, dimensionality reduction, density estimation, etc.
- For this solution example, used a meta-model with a dense network, LSTM encoder-decoder, and PCA.

Reconstruction Evaluation

- Obviously, we want to be able to predict "normal" sequences very well since it is the reconstruction error that we will use to flag for anomalies or not.
- This is a great point to use Hyperparameter Tuning to try and get the best reconstruction possible.

Calculate Error Distribution Statistics

- We now have a trained reconstruction model, but what to do with it?
- Our hypothesis is that since it was trained on "normal" sequences then it should have low error when a sequence should be "normal".
- However, if there is high error this might be an indication of an anomaly.

Calculate Error Distribution Statistics

- Therefore we need to learn a distribution so that we can have some semblance of distance from it for each example.
- A common assumption is that our error is normally distributed therefore the usual approach is to perform Maximum Likelihood Estimation (MLE).
- Once we have found the mean and covariance matrix, we can calculate the Mahalanobis Distance for each example.

Mahalanobis Distance,
generalized
n-dimensional z-score

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} X_{ij}$$

$$\Sigma = (X - \mu)^T (X - \mu)$$

$$MD = diag((X - \mu)\Sigma^{-1}(X - \mu)^T)$$

Calculate Error Distribution Statistics

- For this solution, since our examples are being read in batch, there was a need to create variables to store the running counts, column means, and covariance matrices.
- This needs to be run with the Estimator in TRAIN mode so that they will be written to a checkpoint.

Tuning Anomaly Thresholds

- We've learned the reconstruction error distribution, now we need to set a good threshold so that we minimize the numbers of false positives and negatives, each of which will have different costs.
- The solution branches here depending on if you have labeled data where sequences are annotated as anomalous or not.

Tuning Anomaly Thresholds: Supervised

- If we're lucky enough to have labels that human knowledge will help in our classification task.
- The dataset is a mix of normal and anomalous sequences.
- Remember this isn't an annotation per timestep or per feature, but overall for each matrix in the batch.
- Here we are maximizing an F-β score (with β<1 due to the rarity and wanting to catch all of them).

Tuning Anomaly Thresholds: Supervised

- Sometimes you can cheat a little with supervision...
- For instance, for IoT predictive maintenance, there is a high probability that right after a device begins collecting data, it is probably normal.
- Likewise, right before a device crashes, there was probably anomalous activity just before.
- With some logic based on domain knowledge you can fuzzily assign labels.

Tuning Anomaly Thresholds: Unsupervised

- If we don't have labels, then we have to resort to unsupervised methods.
- In this solution, we calculate the MLE this time for the Mahalanobis Distance and allow a certain number of standard deviations from the mean to flag a sequence as normal.
- This won't have as good of tuning as in the supervised case, but at least it uses the data rather than just be arbitrarily set by a human.

Inference

- For inference, just like in training the examples go through the reconstruction model.
- Then their reconstruction errors are calculated.
- Calculate next their Mahalanobis Distances.
- Then compare with trained anomaly thresholds.
- If any timestep is flagged as anomalous then the whole sequence is flagged in the time major view.
- Likewise for features in the feature major view.

cloud.google.com