WAIV’s deep learning pipeline

Deep Learning for Time Series
• 2017: ideation, research
• 2018: prototype, GTC talk
• 2019: going into production

Anomaly Detection
• 2018: ideation, research
• 2019: prototype, GTC talk

KPI Generation
• 2019: ideation, research
The use case
Network anomalies come in different flavors

**Poor performance**
- Sudden performance shifts
- Exceeding thresholds
- Generating alarms

**Exhausting capacities**
- Reaching operational limits
- Exceeding thresholds
- Performing normally

**Relative performance**
- Clusters performing unusually well
- Neighboring clusters performing worse
- No obvious causes

“Broken Sites”
“Sites Needing Attention”
“Areas Requiring Analysis”
Network anomaly detection today

10’s of thousands of sites. 100’s of thousands of carriers. Millions of metrics. Multiple tools to navigate.
Areas where today’s approach can improve

- Reduce time to detect anomalies
- Reduce the number of tools needed to detect anomalies
- Enable detection based on more than just hard thresholds
- Take advantage of all possible data correlations
- Automate steps, especially when detecting complex anomalies
The generative modeling approach
Hierarchy of unsupervised learning

Unsupervised Learning

Non-Probabilistic Models
- Tractable Models
  - Fully-observed belief nets
  - NADE
  - PixelRNN/CNN
- Non-tractable Models
  - Boltzmann Machines
  - Variational Autoencoders
  - ...
Using an autoencoder for anomaly detection

The embedded/latent space will <hopefully> contain information that is useful for anomaly detection.

https://i-systems.github.io/HSE545/machine%20learning%20all/Workshop/Hanwha/Lecture/image_files/AE_arch2.png
Generative modeling architecture for anomaly detection

Initial goal is to generate clusters of sites that could be anomalies

Then develop a supervised learning model to automatically identify clusters that contain verified anomalies

Training the model
Hyperparameter tuning – very important

Dimension 1 Distribution

Dimension 2 Distribution

BAD
Model outputs when embedded space is 2-dimensional
Correlation heatmap for 2-dimensional embedded space

Original Space

Reconstructed Space
Heatmap of delta between original and reconstructed correlations

Black entries mean the original and reconstructed spaces have similar correlations

We use correlations as a proxy for model quality

There is lots of red and blue in the diagram, so the model is NOT GOOD
Learning Curve for 10-dimensional embedded space

Training and validation loss are almost identical

This means the model is learning the dataset very well
Model outputs when embedded space is 10-dimensional

<table>
<thead>
<tr>
<th>ENODEB</th>
<th>Dim1</th>
<th>Dim10</th>
<th>Dim2</th>
<th>Dim3</th>
<th>Dim4</th>
<th>Dim5</th>
<th>Dim6</th>
<th>Dim7</th>
<th>Dim8</th>
<th>Dim9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.26</td>
<td>-0.08</td>
<td>0.00</td>
<td>0.23</td>
<td>0.21</td>
<td>-0.11</td>
<td>-0.03</td>
<td>-0.22</td>
<td>0.14</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>-0.37</td>
<td>-0.10</td>
<td>0.36</td>
<td>-0.07</td>
<td>-0.10</td>
<td>-0.34</td>
<td>0.35</td>
<td>-0.34</td>
<td>0.34</td>
<td>-0.11</td>
</tr>
<tr>
<td>3</td>
<td>-0.34</td>
<td>-0.12</td>
<td>-0.07</td>
<td>0.20</td>
<td>0.21</td>
<td>0.07</td>
<td>0.12</td>
<td>-0.20</td>
<td>0.02</td>
<td>0.07</td>
</tr>
<tr>
<td>4</td>
<td>-0.28</td>
<td>-0.06</td>
<td>0.28</td>
<td>-0.13</td>
<td>-0.11</td>
<td>-0.35</td>
<td>0.28</td>
<td>-0.15</td>
<td>0.27</td>
<td>-0.05</td>
</tr>
<tr>
<td>5</td>
<td>-0.09</td>
<td>-0.32</td>
<td>-0.08</td>
<td>0.17</td>
<td>0.28</td>
<td>-0.43</td>
<td>0.17</td>
<td>-0.31</td>
<td>0.09</td>
<td>0.11</td>
</tr>
<tr>
<td>6</td>
<td>-0.33</td>
<td>-0.09</td>
<td>0.23</td>
<td>-0.03</td>
<td>-0.10</td>
<td>-0.31</td>
<td>0.28</td>
<td>-0.25</td>
<td>0.37</td>
<td>-0.12</td>
</tr>
<tr>
<td>7</td>
<td>-0.31</td>
<td>-0.08</td>
<td>0.12</td>
<td>-0.06</td>
<td>-0.13</td>
<td>-0.25</td>
<td>0.21</td>
<td>-0.18</td>
<td>0.29</td>
<td>-0.18</td>
</tr>
<tr>
<td>8</td>
<td>-0.21</td>
<td>-0.13</td>
<td>0.08</td>
<td>0.25</td>
<td>0.33</td>
<td>-0.15</td>
<td>0.10</td>
<td>-0.24</td>
<td>0.18</td>
<td>0.24</td>
</tr>
<tr>
<td>9</td>
<td>-0.03</td>
<td>-0.37</td>
<td>-0.09</td>
<td>0.32</td>
<td>0.29</td>
<td>-0.30</td>
<td>0.23</td>
<td>-0.34</td>
<td>0.23</td>
<td>0.06</td>
</tr>
<tr>
<td>10</td>
<td>-0.08</td>
<td>-0.35</td>
<td>-0.13</td>
<td>0.19</td>
<td>0.29</td>
<td>-0.37</td>
<td>0.14</td>
<td>-0.29</td>
<td>0.23</td>
<td>0.08</td>
</tr>
<tr>
<td>11</td>
<td>-0.14</td>
<td>-0.20</td>
<td>-0.35</td>
<td>-0.05</td>
<td>0.16</td>
<td>0.04</td>
<td>0.10</td>
<td>-0.28</td>
<td>-0.01</td>
<td>-0.02</td>
</tr>
<tr>
<td>12</td>
<td>-0.29</td>
<td>-0.17</td>
<td>-0.08</td>
<td>0.26</td>
<td>0.26</td>
<td>0.02</td>
<td>0.03</td>
<td>-0.27</td>
<td>0.12</td>
<td>0.07</td>
</tr>
<tr>
<td>13</td>
<td>-0.33</td>
<td>-0.04</td>
<td>0.09</td>
<td>-0.06</td>
<td>-0.16</td>
<td>-0.29</td>
<td>0.25</td>
<td>-0.14</td>
<td>0.21</td>
<td>-0.13</td>
</tr>
<tr>
<td>14</td>
<td>-0.32</td>
<td>-0.06</td>
<td>0.23</td>
<td>-0.04</td>
<td>-0.12</td>
<td>-0.32</td>
<td>0.34</td>
<td>-0.19</td>
<td>0.31</td>
<td>-0.05</td>
</tr>
<tr>
<td>15</td>
<td>-0.31</td>
<td>-0.10</td>
<td>-0.08</td>
<td>0.28</td>
<td>0.23</td>
<td>-0.07</td>
<td>0.07</td>
<td>-0.22</td>
<td>0.07</td>
<td>0.16</td>
</tr>
<tr>
<td>16</td>
<td>-0.25</td>
<td>-0.16</td>
<td>-0.08</td>
<td>0.24</td>
<td>0.28</td>
<td>0.08</td>
<td>0.16</td>
<td>-0.13</td>
<td>0.16</td>
<td>0.14</td>
</tr>
</tbody>
</table>
Correlation heatmap for 10-dimensional embedded space

Original Space

Reconstructed Space

Structural Similarity Index 0.92
Heatmap of delta between original and reconstructed correlations

Black entries mean the original and reconstructed spaces have similar correlations.

We use correlations as a proxy for model quality.

There’s lots of black in the diagram, so the model is good.
Final autoencoder model architecture

Key Features

- 163 input features
- 9 hidden layers
- 10-dimensional embedded space
- Mean Squared Error (MSE) loss
- Adam (RMSprop + Momentum) optimizer

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_1 (InputLayer)</td>
<td>(None, 163)</td>
<td>0</td>
</tr>
<tr>
<td>dense (Dense)</td>
<td>(None, 200)</td>
<td>32800</td>
</tr>
<tr>
<td>dense_1 (Dense)</td>
<td>(None, 100)</td>
<td>20100</td>
</tr>
<tr>
<td>dense_2 (Dense)</td>
<td>(None, 40)</td>
<td>4040</td>
</tr>
<tr>
<td>dense_3 (Dense)</td>
<td>(None, 20)</td>
<td>820</td>
</tr>
<tr>
<td>dense_4 (Dense)</td>
<td>(None, 10)</td>
<td>210</td>
</tr>
<tr>
<td>dense_5 (Dense)</td>
<td>(None, 20)</td>
<td>220</td>
</tr>
<tr>
<td>dense_6 (Dense)</td>
<td>(None, 40)</td>
<td>840</td>
</tr>
<tr>
<td>dense_7 (Dense)</td>
<td>(None, 100)</td>
<td>4100</td>
</tr>
<tr>
<td>dense_8 (Dense)</td>
<td>(None, 200)</td>
<td>20200</td>
</tr>
<tr>
<td>dense_9 (Dense)</td>
<td>(None, 163)</td>
<td>32763</td>
</tr>
</tbody>
</table>
Autoencoder implementation

NGC TensorFlow container optimized for Nvidia Volta GPU

- RAPIDS
- Keras
- TensorFlow

AWS P3 Instance

- Verizon RHEL AMI
Using the model outputs
Looking for anomalies in the embedded space
Looking for anomalies based on reconstructed outputs

Can we perform anomaly detection by identifying outliers based on reconstruction error?

Kernel Density Estimation

Profile Plot

Outliers

ReconsError
Clustering the reconstruction error outliers

Too many outliers to analyze individually
Cluster the outliers into groups to ease analysis
Use DBSCAN to find clusters and estimate the number of clusters

Analyzing the clustering results

Network engineers manually inspected performance data from cell sites in each cluster.

They found that a cluster that included all high-usage sites from the input data.

This means the model automatically learned to detect an anomaly that we train our new engineers to find.

What’s next
Mature the anomaly detection capability

Simple Anomalies

- Simple outliers (e.g., High-usage sites)

Known Complex Anomalies

- Complex multivariate anomalies (e.g., Passive Intermodulation)

Unknown Complex Anomalies

- New issues not seen before (e.g., SW Bug on new counter)
Try more generative models

https://i-systems.github.io/HSE545/machine%20learning%20all/Workshop/Hanwha/Lecture/image_files/AE_arch2.png

https://i-systems.github.io/HSE545/machine%20learning%20all/12%20Deep%20Learning/image_files/my_GAN.png
Variational autoencoder model architecture

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>encoder_input (InputLayer)</td>
<td>(None, 163)</td>
<td>0</td>
</tr>
<tr>
<td>vae_latent_sample (Model)</td>
<td>(None, 10)</td>
<td>58190</td>
</tr>
<tr>
<td>vae_output_sample (Model)</td>
<td>(None, 163)</td>
<td>90996</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_latent_values (InputLayer)</td>
<td>(None, 10)</td>
<td>0</td>
</tr>
<tr>
<td>decoder_1 (Dense)</td>
<td>(None, 10)</td>
<td>110</td>
</tr>
<tr>
<td>decoder_2 (Dense)</td>
<td>(None, 20)</td>
<td>220</td>
</tr>
<tr>
<td>decoder_3 (Dense)</td>
<td>(None, 40)</td>
<td>840</td>
</tr>
<tr>
<td>decoder_4 (Dense)</td>
<td>(None, 100)</td>
<td>4100</td>
</tr>
<tr>
<td>decoder_5 (Dense)</td>
<td>(None, 200)</td>
<td>20200</td>
</tr>
<tr>
<td>output_mean (Dense)</td>
<td>(None, 163)</td>
<td>32763</td>
</tr>
<tr>
<td>output_variance (Dense)</td>
<td>(None, 163)</td>
<td>32763</td>
</tr>
</tbody>
</table>
Generated embedded space from variational autoencoder

How can this be used?

Come back next year to learn more!

Kernel Density Estimation of the latent space generated by the variational autoencoder
Future work

Model improvements to increase anomaly detection rate

Optimize code for the Rapids platform

Modular anomaly detection framework

Combined KPI forecast and anomaly detection frameworks

Use variational autoencoders and GANs to improve network understanding
Thank you.
Model evaluation guideline