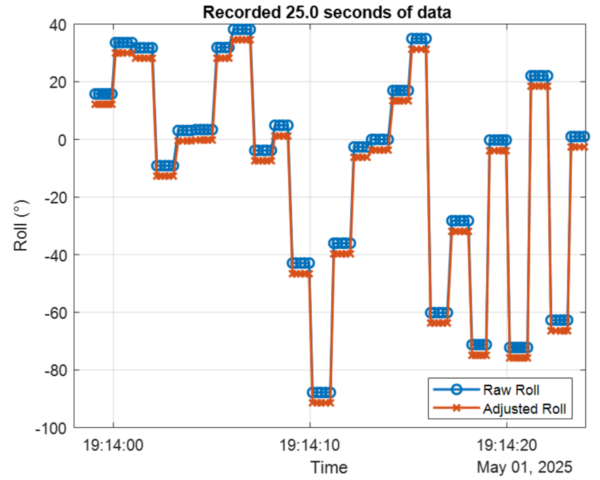
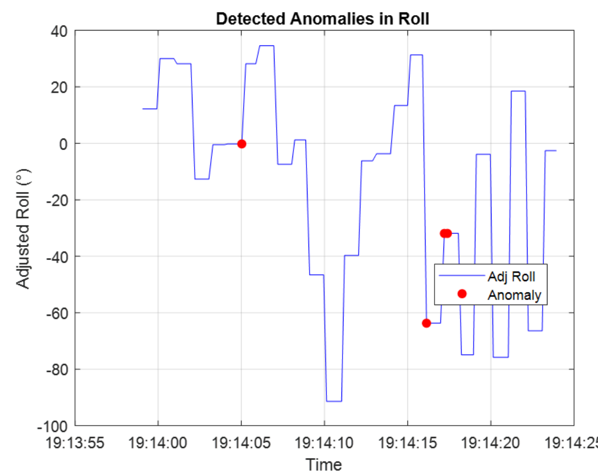
1) “Adjusted Roll with Anomalies”

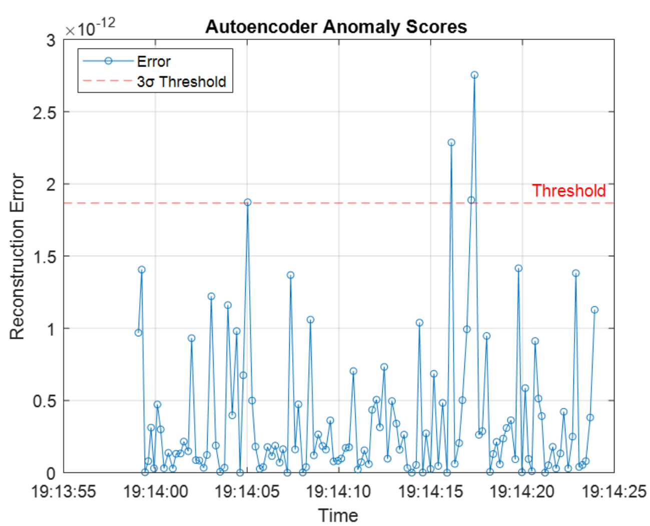
- **Blue line** plots your **adjusted roll** (adjVals) over time.
- **Red circles** mark the exact timestamps where the reconstruction error from the autoencoder exceeded the 3-σ threshold.
- In plain terms: whenever your roll-angle dipped or spiked in a way the model hadn’t seen during training, the reconstruction error jumped high enough to cross the red threshold line, and we plot a red dot there.



2) “Autoencoder Anomaly Scores”

- **Blue circles** show the **reconstruction error** (mean squared error between the normalized input and its autoencoder reconstruction) at each sample.
- The **horizontal dashed red line** is your **3-σ threshold** (mean error + three standard deviations).
- Every time a blue marker sits above that line, it indicates the autoencoder couldn’t faithfully reconstruct that roll value—i.e. it was sufficiently “unusual” compared to what it was trained on.





How they were detected

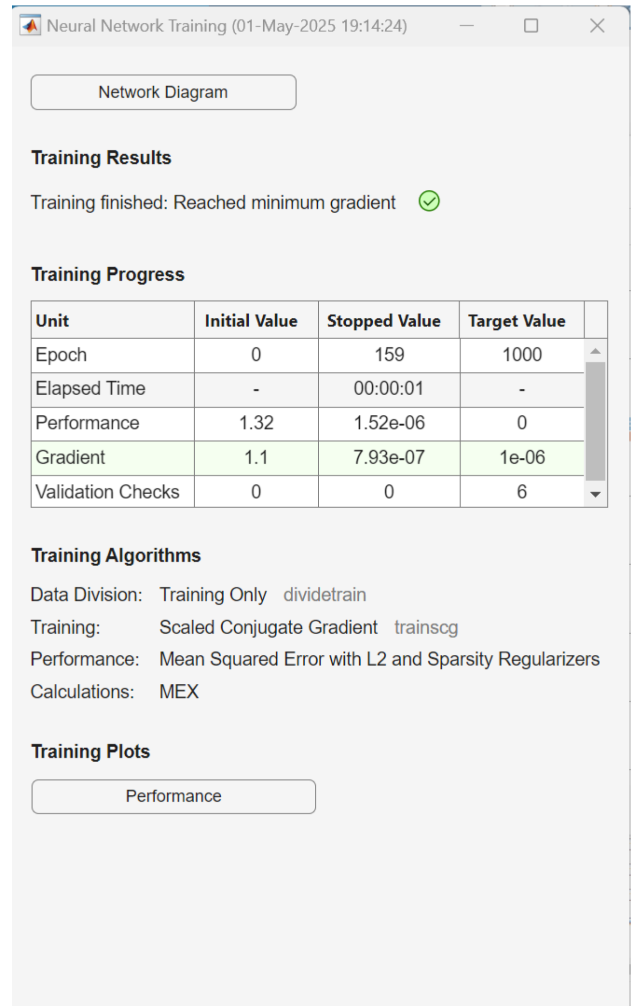
1. **Train on "normal" data only**
   We used your entire recorded sequence of adjusted rolls (assumed to be normal posture) to train a small autoencoder that learns to compress and then reconstruct those patterns with minimal error.
2. **Compute per-sample error**
   After training, we ran each adjusted-roll sample back through the network to get a reconstruction. The squared difference (input – output)^2 is plotted as the blue "Error" trace.
3. **Set an anomaly threshold**
   We chose threshold = mean(error) + 3*std(error) so that any sample whose error sits more than three standard deviations above the typical reconstruction error is considered an outlier.
4. **Flag and overlay anomalies**
   Wherever the error crosses that threshold, we mark the point red on your roll-vs-time plot. Those timestamps are the moments where your back-angle pattern deviated enough from "normal" that the network deemed them anomalous.

So in summary: the autoencoder translates typical roll-angle patterns into near-zero errors, and anything it can't reconstruct cleanly (error > 3σ) gets flagged as an anomaly, which you see both in the error-score plot and as red dots on the roll trace.

Think of the training phase as "teaching" our little autoencoder what "normal" rowing posture looks like. These three windows show **what happened** during that teaching process, and **what the network itself** looks like once it's finished.

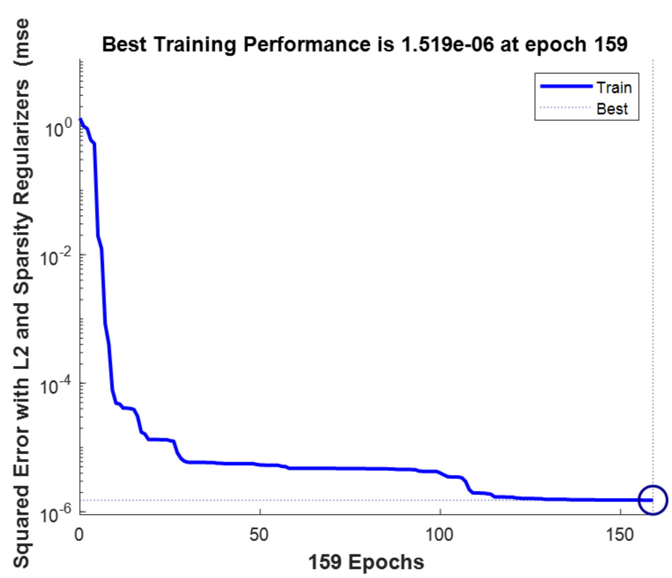
1) Training Results Summary

- **“Training finished: Reached minimum gradient”**
 Means the learning algorithm slowed down weight-updates to almost zero—i.e. it has “settled” on a set of parameters that don’t change much anymore.
- **Performance = 1.52×10^{-6}**
 That tiny number is the final average squared difference between each input and its reconstruction (plus our little regularization penalties). Near-zero error means it’s mastered reproducing the examples you gave it.
- **Data Division: Training Only**
 We showed it **all** of your “normal” roll data—there was no separate test set—because our goal is purely to model “normal” so we can spot deviations later.
- **Training Algorithm: Scaled Conjugate Gradient**
 A fast optimizer that zipped through the small network to find those weight & bias values.



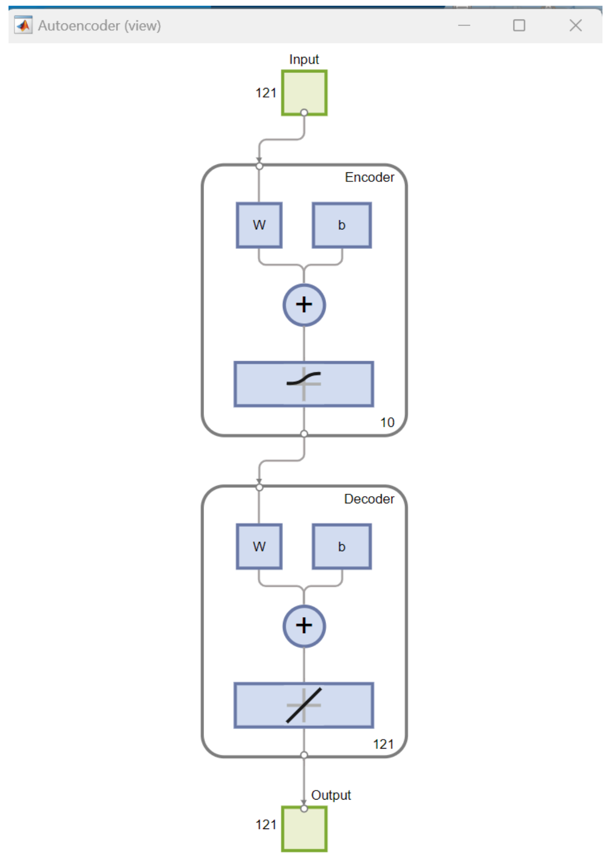
2) Training Progress Plot

- **Horizontal axis = Epochs (passes through the data)**
 You can see it take roughly 160 sweeps before the error bottoms out.
- **Vertical axis (log scale) = Reconstruction Error**
 Starts near 10^0 (pretty large mistakes), then rapidly falls into the 10^{-4}–10^{-6} range as the autoencoder “remembers” the patterns.



- **Dotted line at 1.5×10^{-6}**
  That's the **best** error it achieved (circled at epoch 159), exactly matching the summary above.

3) Final Autoencoder Architecture



- **Input = 121 values**
  That's your window of roll-angle samples (e.g. one full second of data at ~120 Hz).
- **Encoder (top box)**
  Compresses those 121 numbers down to a **10-element code** (via a weight matrix and bias, then a nonlinear activation).
- **Decoder (bottom box)**
  Expands that 10-element code back out to 121 numbers—our reconstruction of the original posture sequence.
- By learning just how to pack & unpack those roll-values with minimal error, the network internalizes your "normal" motion pattern.

In English:

1. We showed the network a bunch of roll sequences labeled "normal."
2. Over about 160 training passes, it learned weights and biases so it could compress and then perfectly reconstruct those sequences (error ≈ 0).
3. The final 10-unit "bottleneck" is its memory of the most important features, and the near-zero error means it has a very tight model of what "normal" looks like.

Later, when you feed it a brand-new roll sequence, any big reconstruction error (above the red line) means "this doesn't fit my normal-motion memory" and that's exactly how we detect anomalies in real time.