A comparison between noise reduction & analysis techniques for Random Telegraph Signal pixels

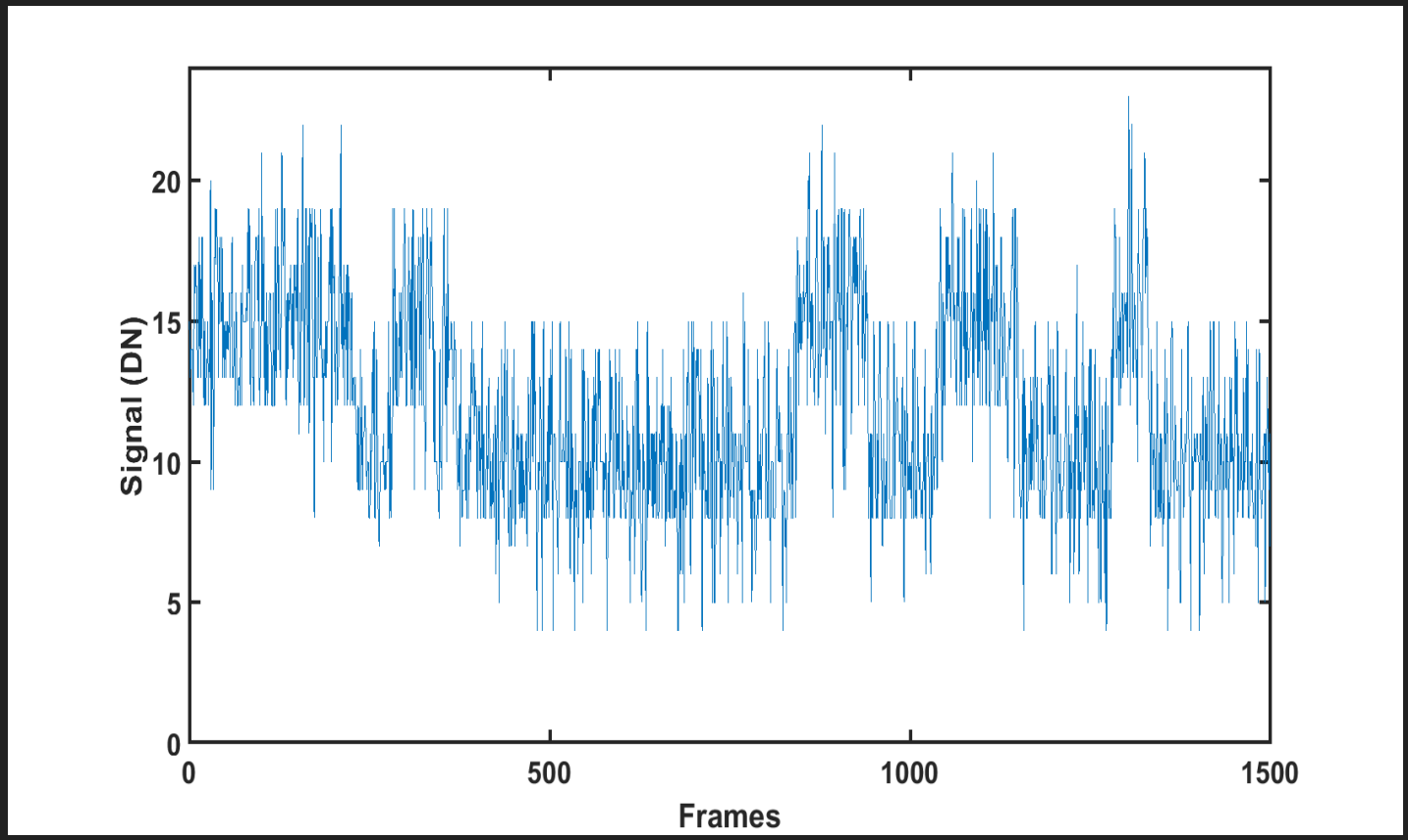
Benjamin Hendrickson, Ralf Widenhorn, Morley Blouke, and Erik Bodegom

Portland State University

Portland, OR

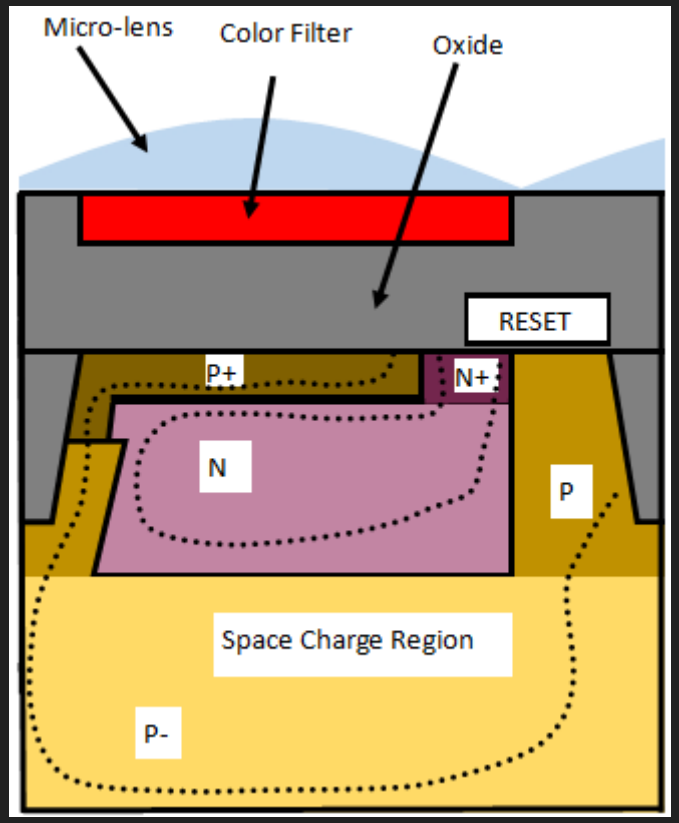
RTS Noise - Overview

- Observed in CCD and CMOS architectures
- Defined by discrete changes in signal level (blinking pixels)
- Recognized by similar lifetimes
- Two flavors of RTS
  - Source Follower RTS (not discussed)
  - Dark Current RTS (DC-RTS)



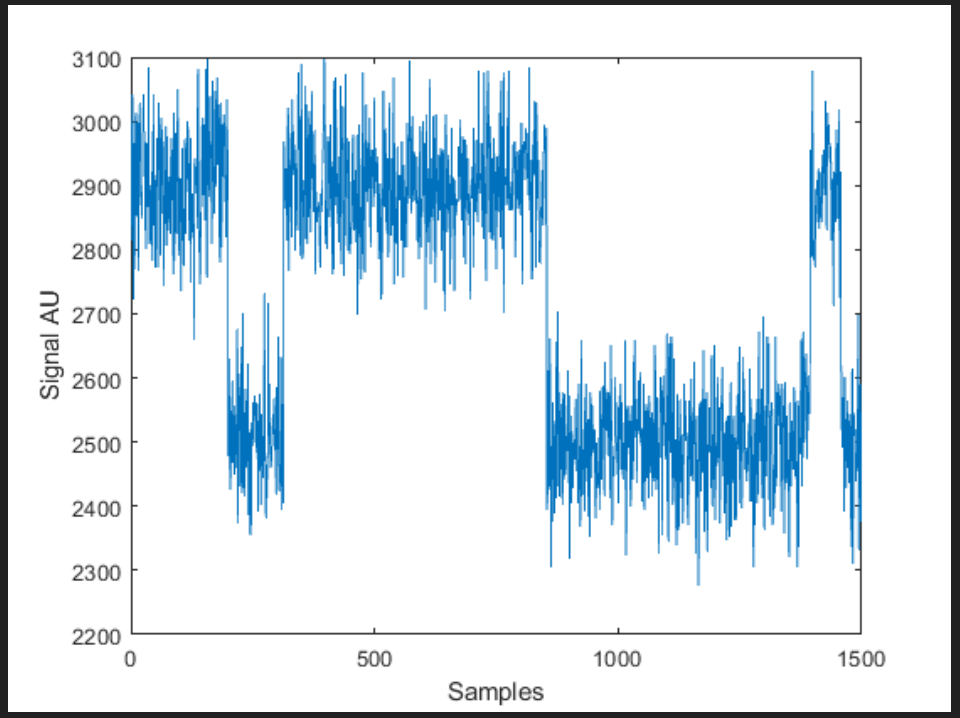
Dark Current RTS (DC-RTS)

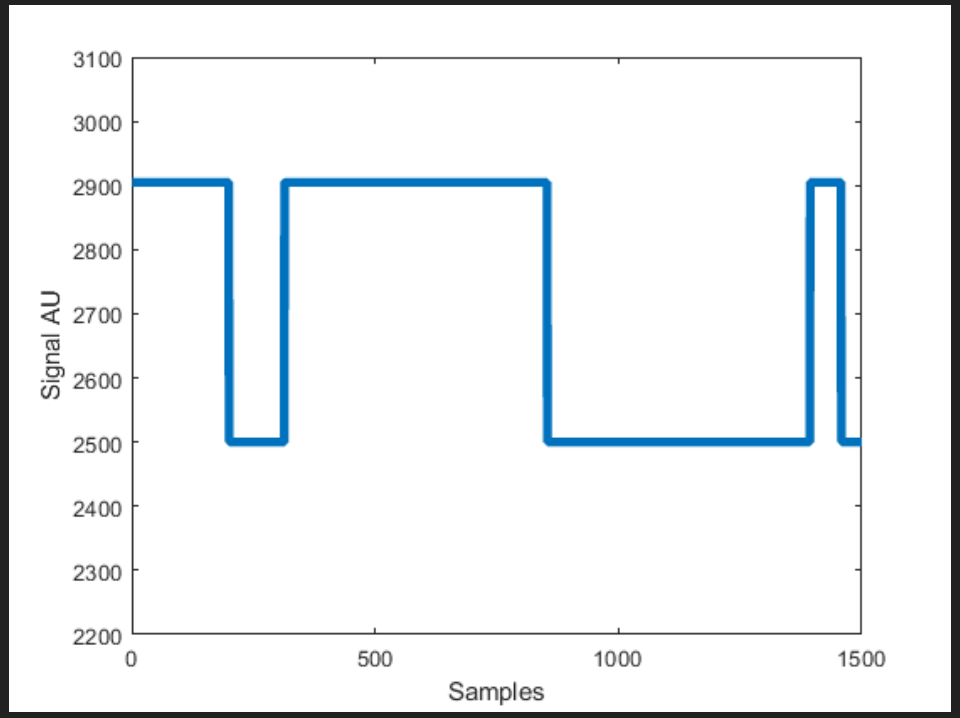
- Origin – metastable Shockley-Read-Hall generation/recombination centers
- Verified by integration time dependence on RTS amplitude
- Radiation damage effect
  - Protons typically create DC-RTS centers in bulk
  - X-rays/γ-rays typically create DC-RTS centers on Si-SiO2 interface



Signal Reconstruction

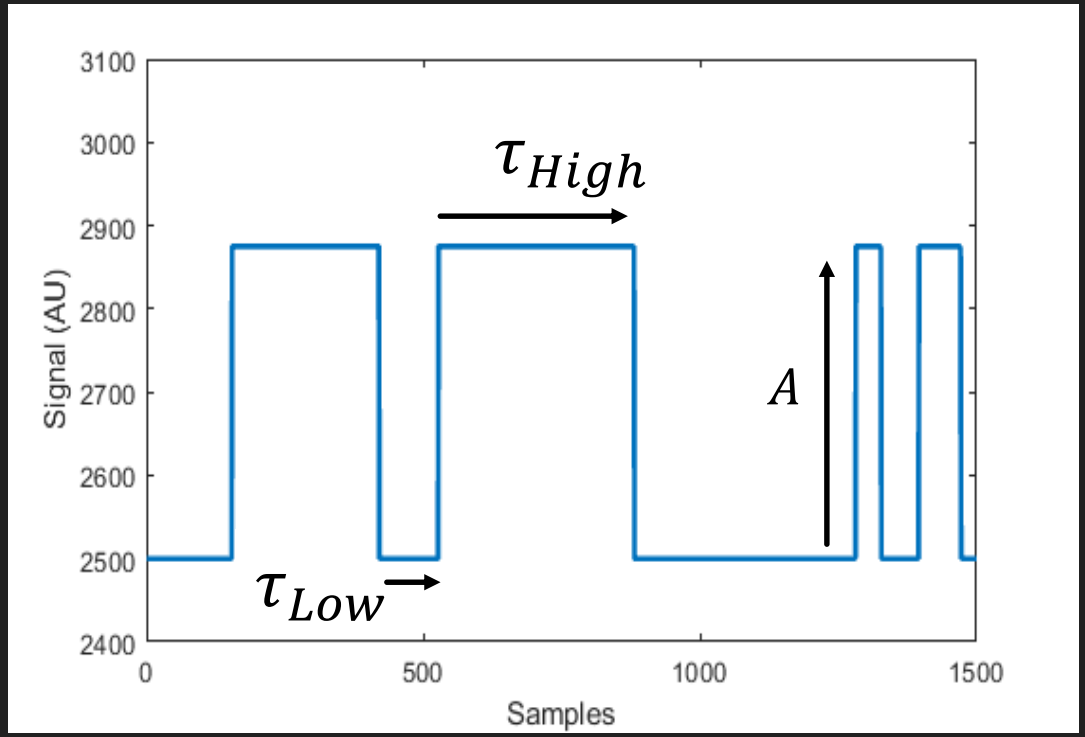
- What does perfect approximation look like?
- Zero white noise contribution
- Perfect RTS representation in shape and scale





Signal Reconstruction

- Primary RTS characteristics
  - State lifetime (time constant)
  - RTS amplitude
- No well defined limits in τ and A for RTS signals
- Small τ's and A's make RTS transitions difficult to distinguish from normal Gaussian noise



Convolution Method

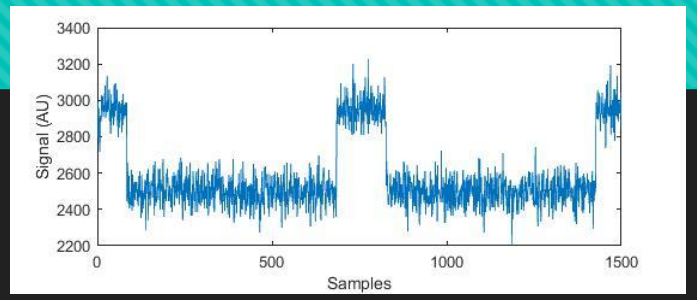
- Code provided by V. Goiffon et. al. (2009)

Multilevel RTS in Proton Irradiated CMOS Image Sensors Manufactured in a Deep Submicron Technology

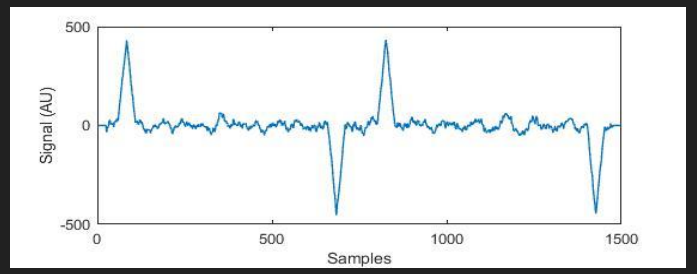
V. Goiffon, *Member, IEEE*, G. R. Hopkinson, *Member, IEEE*, P. Magnan, *Member, IEEE*, F. Bernard, G. Rolland, and O. Saint-Pé

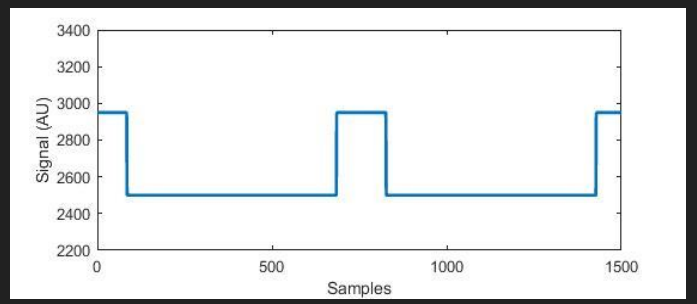
Convolution Method – cont.

- Applies a step shaped filter to a signal
- Detects RTS if $A_{max} > \sigma_{sig}$
- Measures the mean value between spikes to estimate RTS signal levels
- Sorts RTS levels and approximates Gaussian noise-free RTS signal

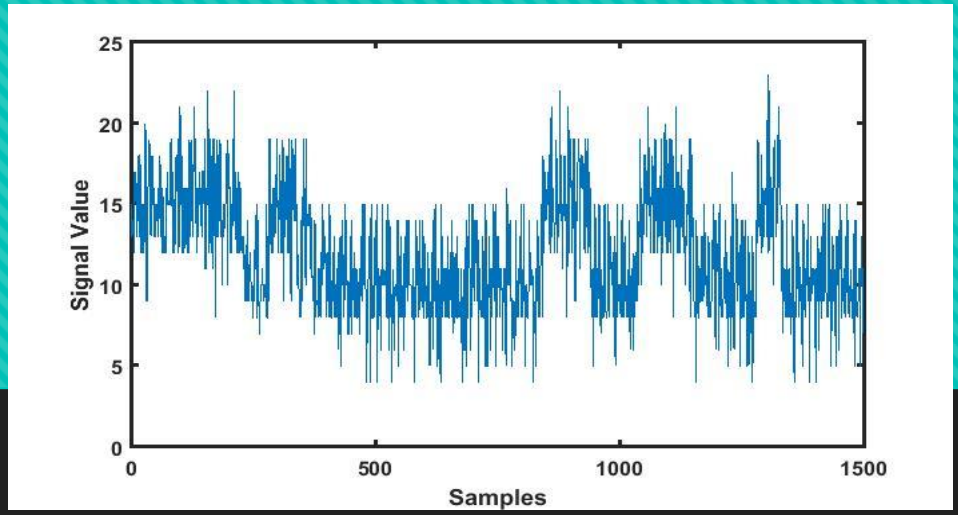


$$H(z) = \frac{2}{L}\left(-\sum_{i=0}^{\frac{L}{2}-1} z^{-i} + \sum_{i=\frac{L}{2}}^{L-1} z^{-i} \right)$$

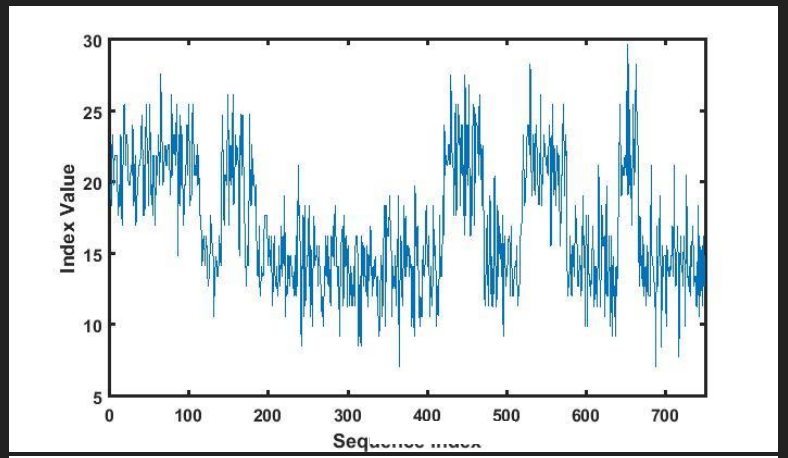


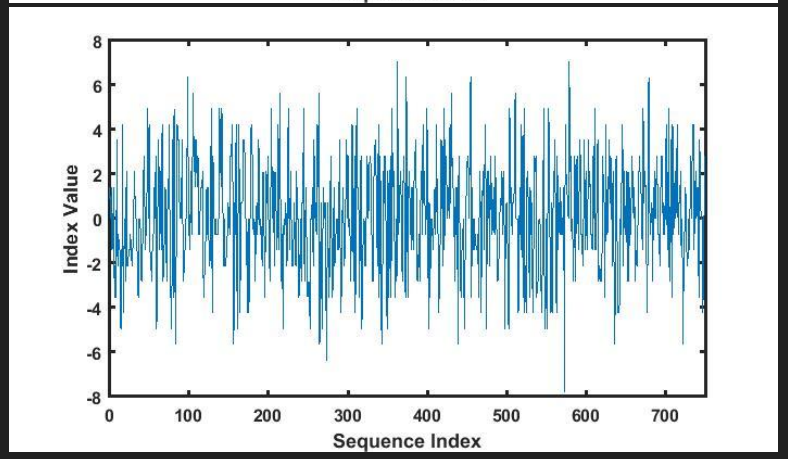


Wavelet Analysis – DWT



- The discrete wavelet transform (DWT) breaks $\mathbf{f}$ of length N into two 'daughter' sequences of length $\mathrm{N}/2$
  - Trend Sequence Members
    - $a_m = \frac{f_{2m-1}+f_{2m}}{\sqrt{2}} \qquad 1 < m \leq N/2$
  - Details Sequence Members
    - $d_m = \frac{f_{2m-1}-f_{2m}}{\sqrt{2}} \qquad 1 < m \leq N/2$





J. S. Walker, A Primer on Wavelets and their Scientific Applications. Boca Raton [Fla.]: Chapman & Hall/CRC, 2nd ed., 2008.

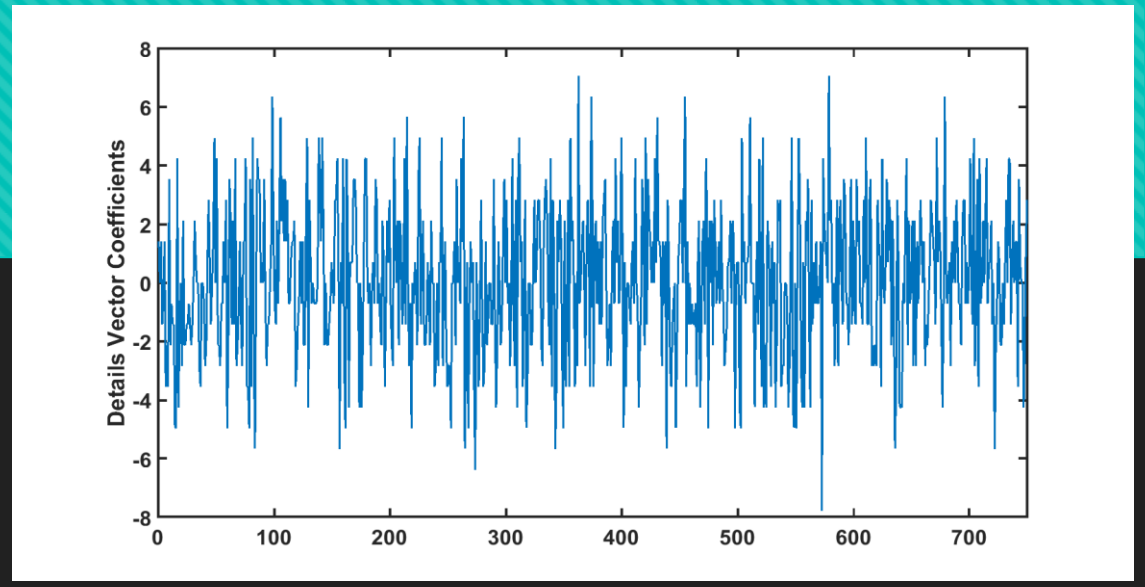
DWT Denoising Method

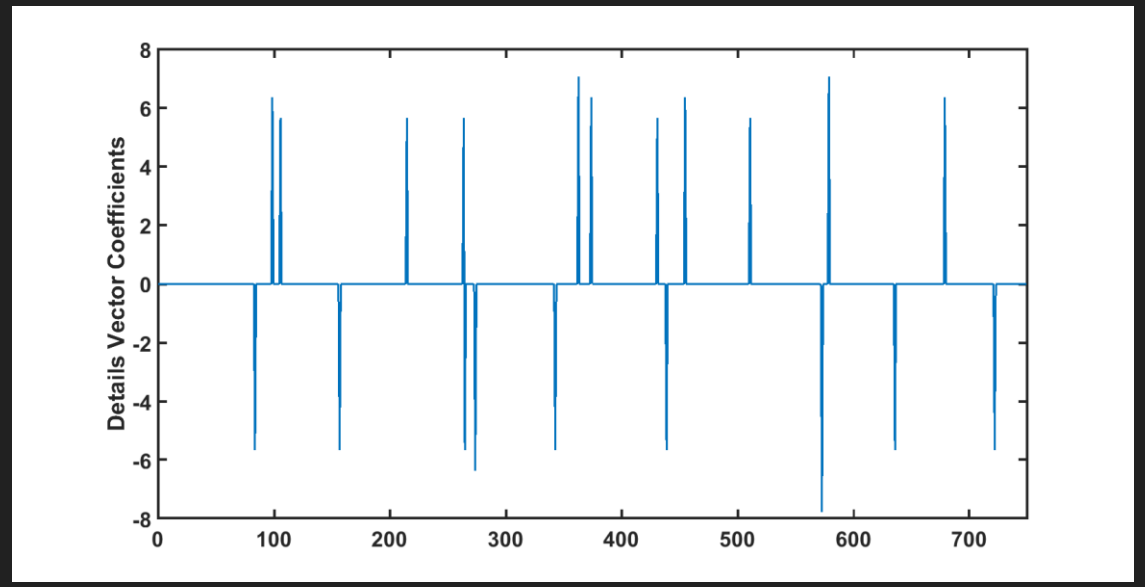
- White noise is suppressed by thresholding the details sequence
  - Similar to a high-pass/low-pass filter
  - Based on magnitude rather than frequency
- The threshold is statistically derived

$$\mathrm{T} = \hat{\sigma}\sqrt{2\log(n)}$$

- T is the universal threshold derived by Donoho and Johnstone†
- Values below the threshold are set to zero

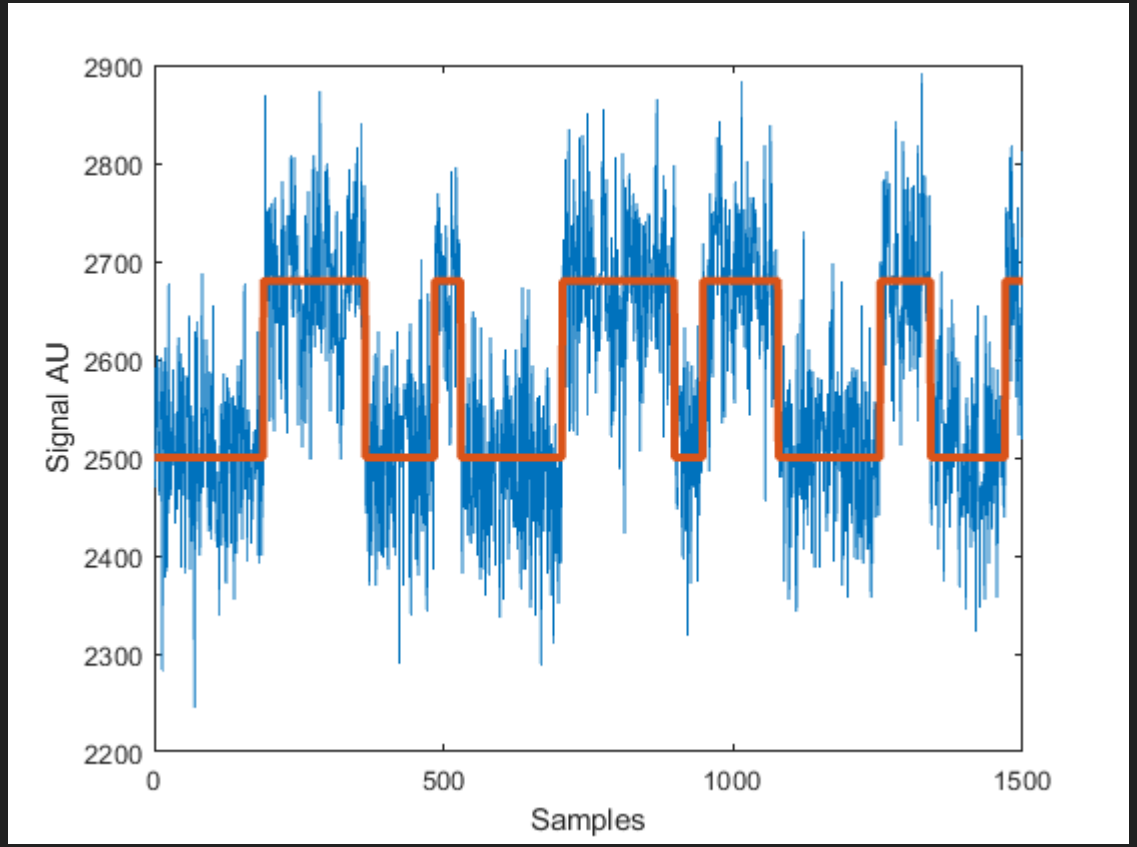
† G. P. Nason, "Choice of the threshold parameter in wavelet function estimation," Wavelets and statistics, vol. 103, pp. 261–280, 1995.





DWT Denoising Method

- Inverse DWT is performed on heavily thresholded signal
- Mean levels are sorted and a Gaussian noise free signal is constructed

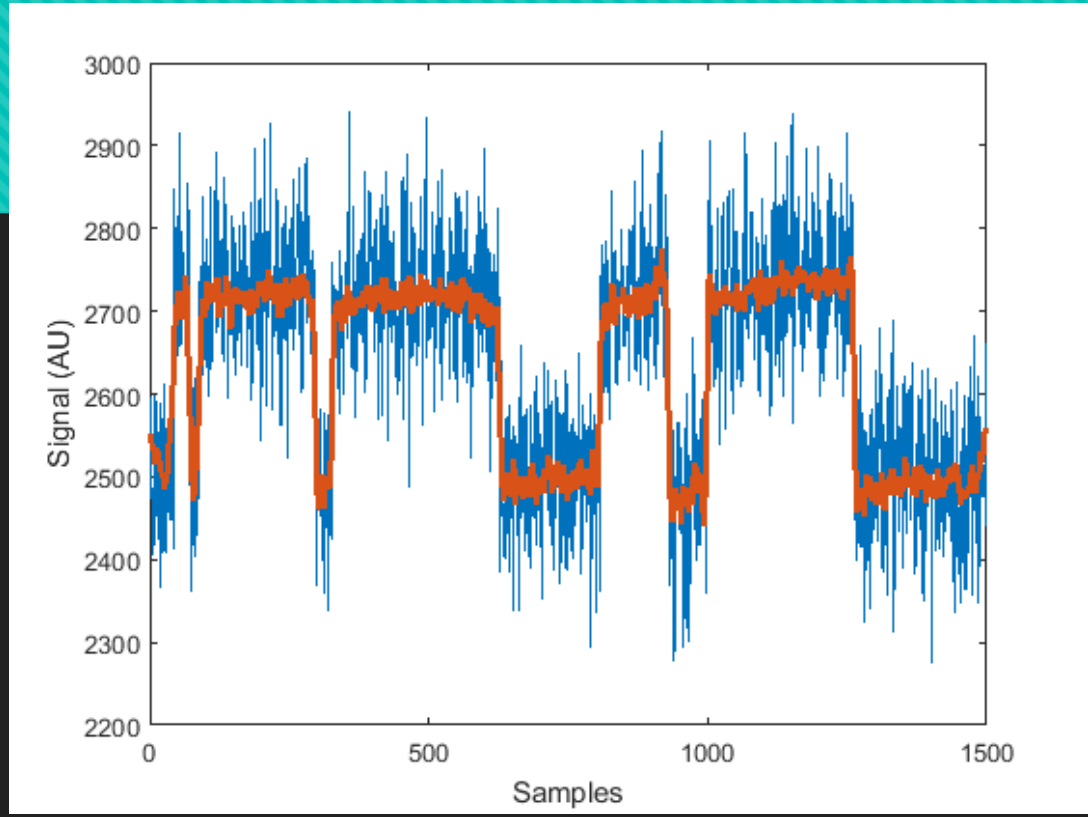


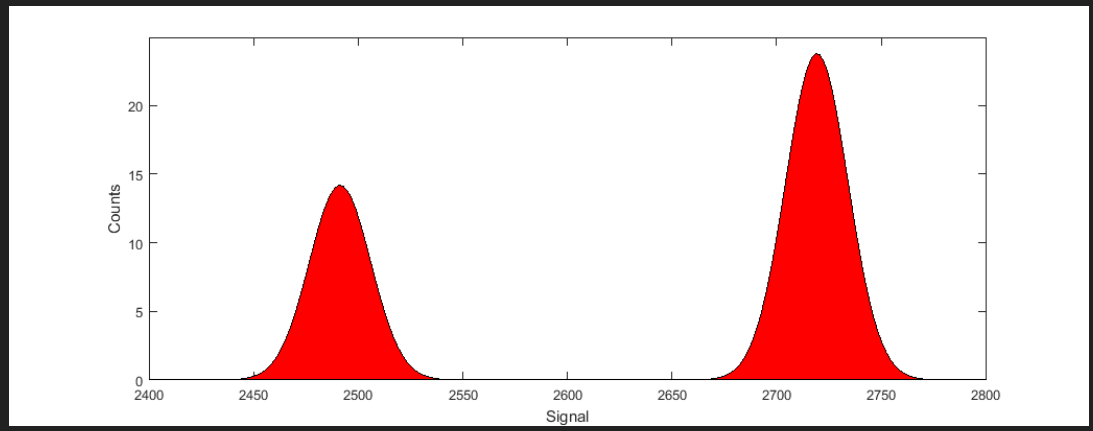
Machine Learning Method

- RTS signals are detected and approximated using convolutional neural networks
- RTS detection is performed by a classification model
  - Similar to image classification
  - Takes a signal and returns a zero for RTS or one for non-RTS
- WN reduction is performed by an autoencoder
  - Trained by creating gaps in signals, and 'learning' the best way to fill in those gaps
  - Takes a noisy signal and returns a clean signal

Machine Learning Method

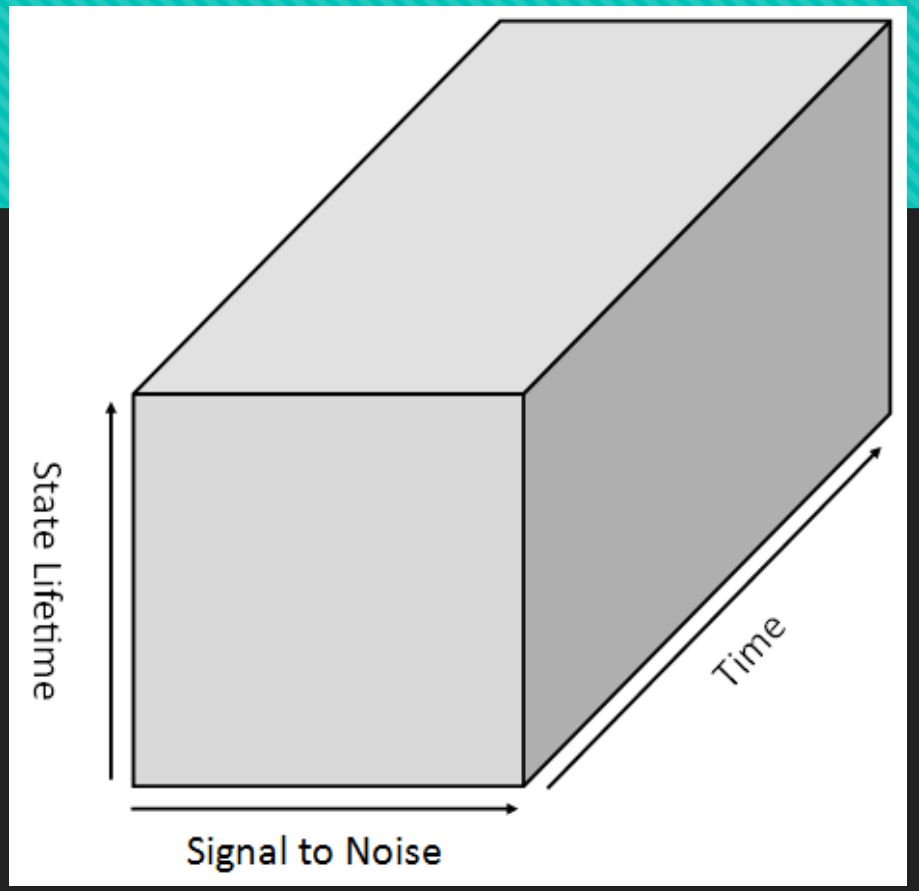
- To finish approximation with the ML method, a histogram is created from the autoencoder output
- The result is fitted as the sum of two Gaussian distributions
- The peaks are taken as the RTS signal levels, and the signal is reconstructed where each sample from the autoencoder snaps to its closest value from the histogram fit





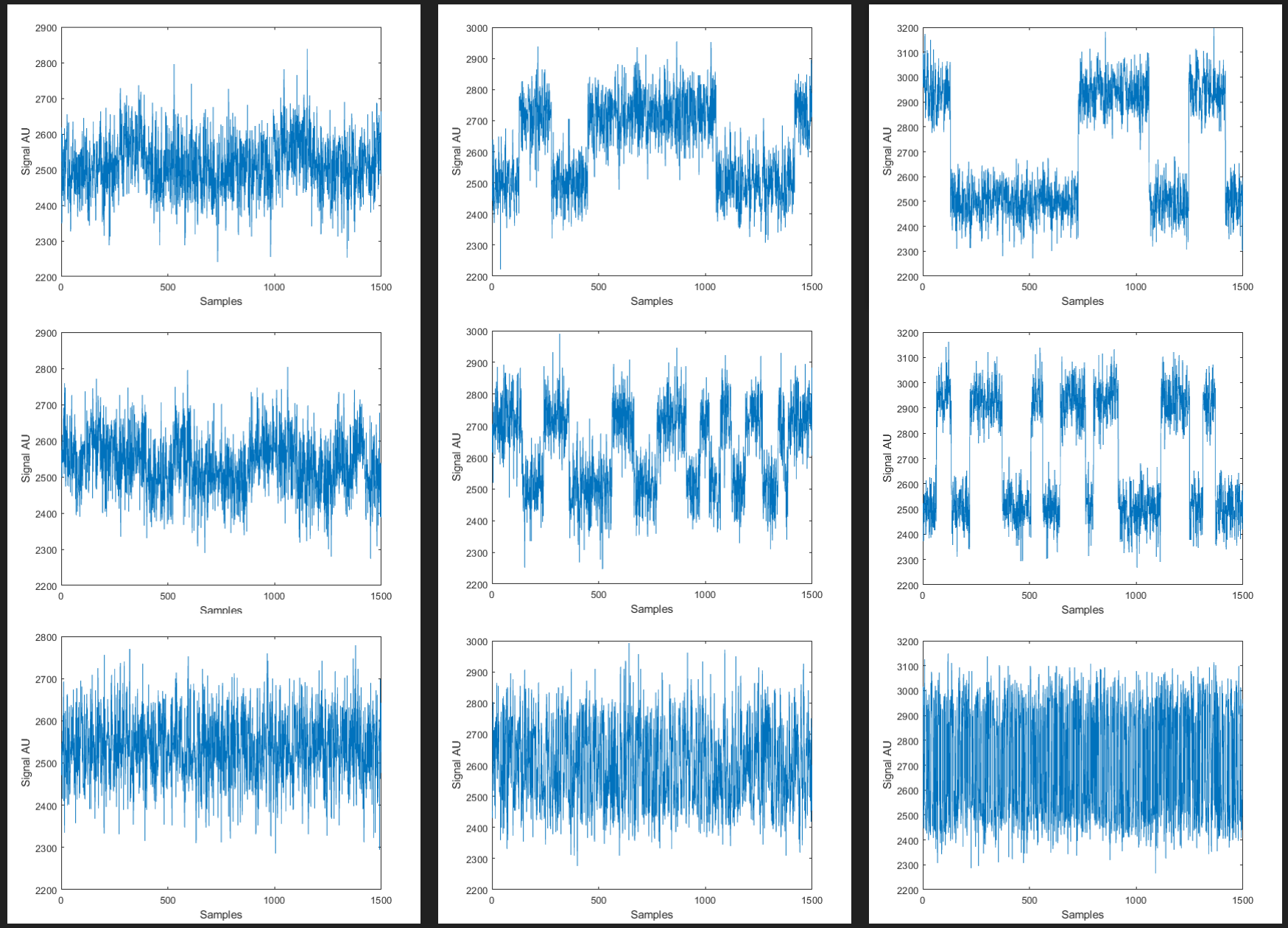
Testing Procedure

- Each method is tested for detection and approximation on a data block of 90,000 simulated RTS signals
- The signals begin clean, then have Gaussian noise added over the top
- One dimension of the block spans the signal to noise, defined as RTS amplitude/white noise floor, from ~zero to 6
- The other dimension spans the state lifetime of a signal from 1 to 300 samples
- The approximation of each signal is scored by its correlation coefficient against the noiseless version of input signal
- Each method is tested for false positive detection on a block of 90,000 non-RTS signals



$$C_{xy} = \frac{\Sigma(x-\overline{x})(y-\overline{y})}{\sqrt{\Sigma(x-\overline{x})^2}\sqrt{\Sigma(y-\overline{y})^2}}$$

State Lifetime

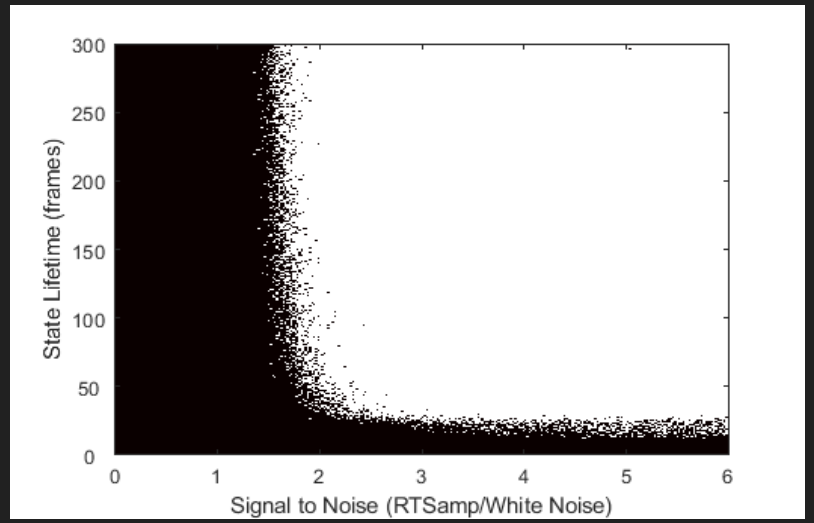


RTS Amplitude

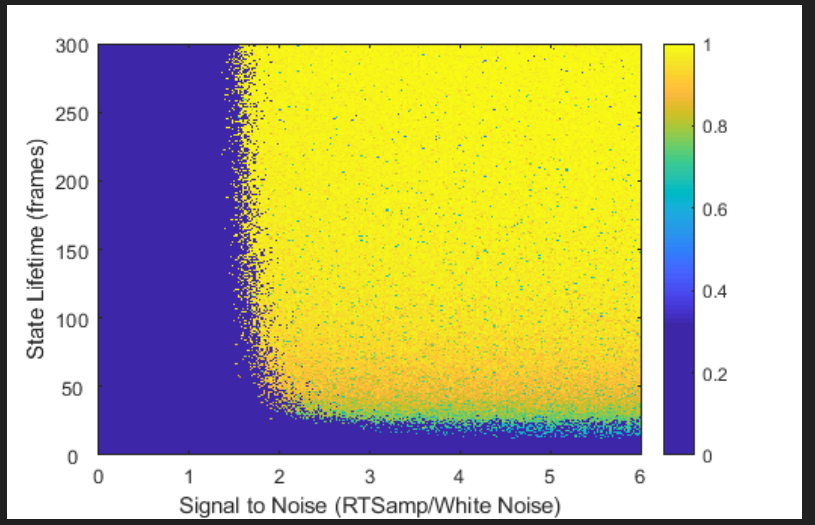
Results – Convolution Method

- Reliably works with $SNR > \sim 2$ and $\tau > \sim 50$ frames
- 66% RTS detection rate
- Mean C_{xy} of for detected signals: 0.9474
- Zero(!) non-RTS false positives

Detection



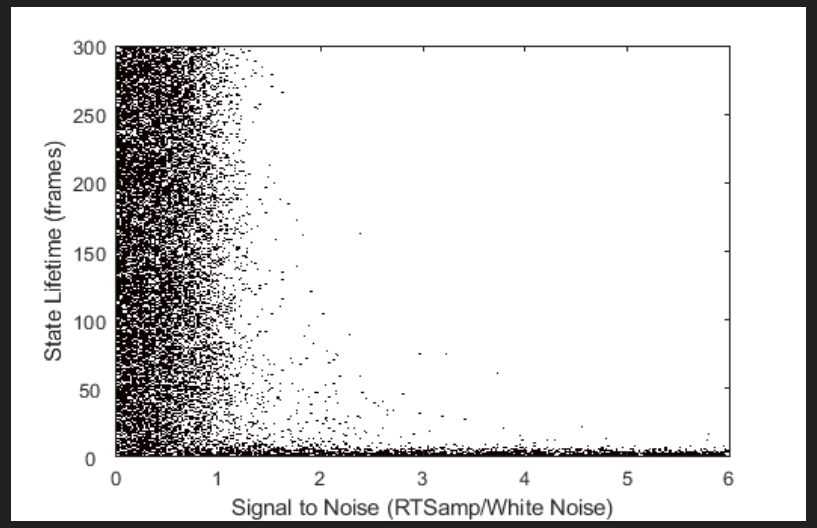
Correlation



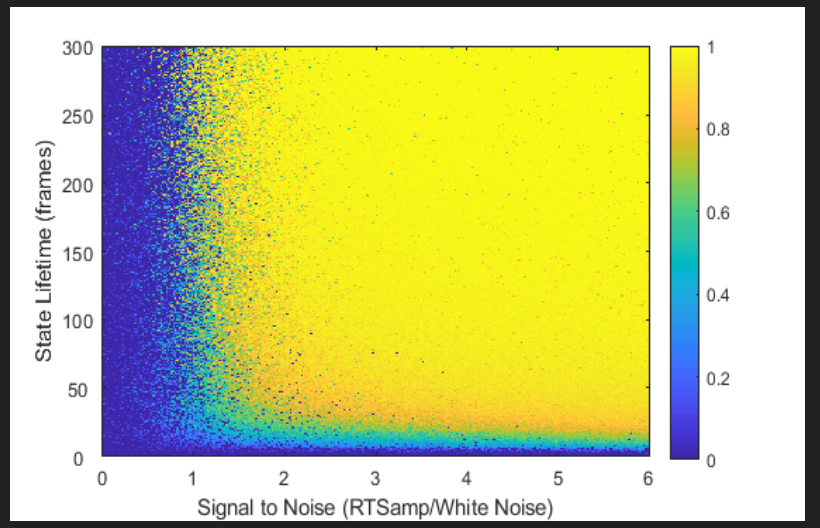
Results – Wavelet Method

- Reliably works with $SNR > \sim 2$ and $\tau > \sim 50$ frames
- 86.6% RTS detection rate
- Mean C_{xy} of for detected signals: 0.8644
- 21.7% non-RTS false positive detection

Detection



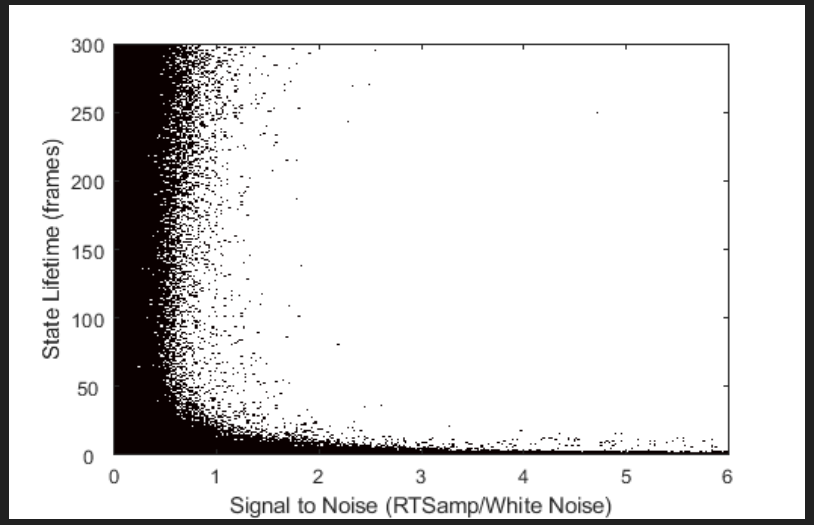
Correlation



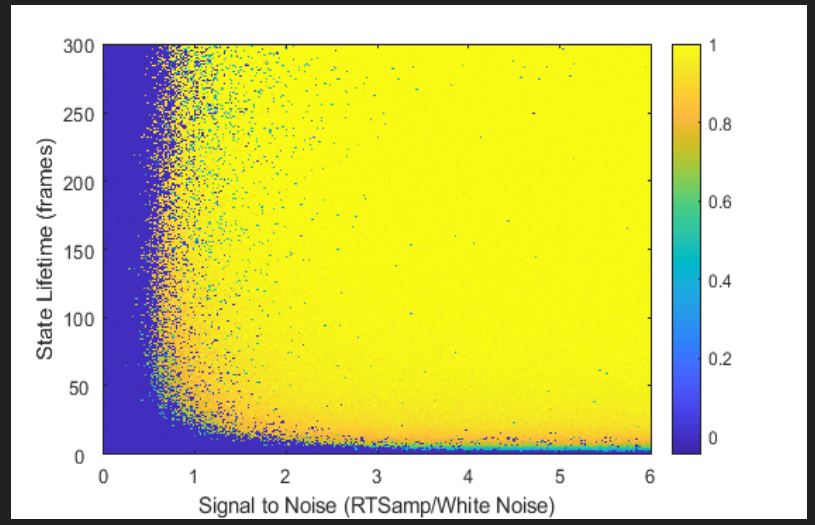
Results – Machine Learning Method

- Reliably works with $SNR > \sim 1$ and $\tau > \sim 25$ frames
- 85.4% RTS detection rate
- Mean C_{xy} of for detected signals: 0.9564
- Zero(!) non-RTS false positives

Detection

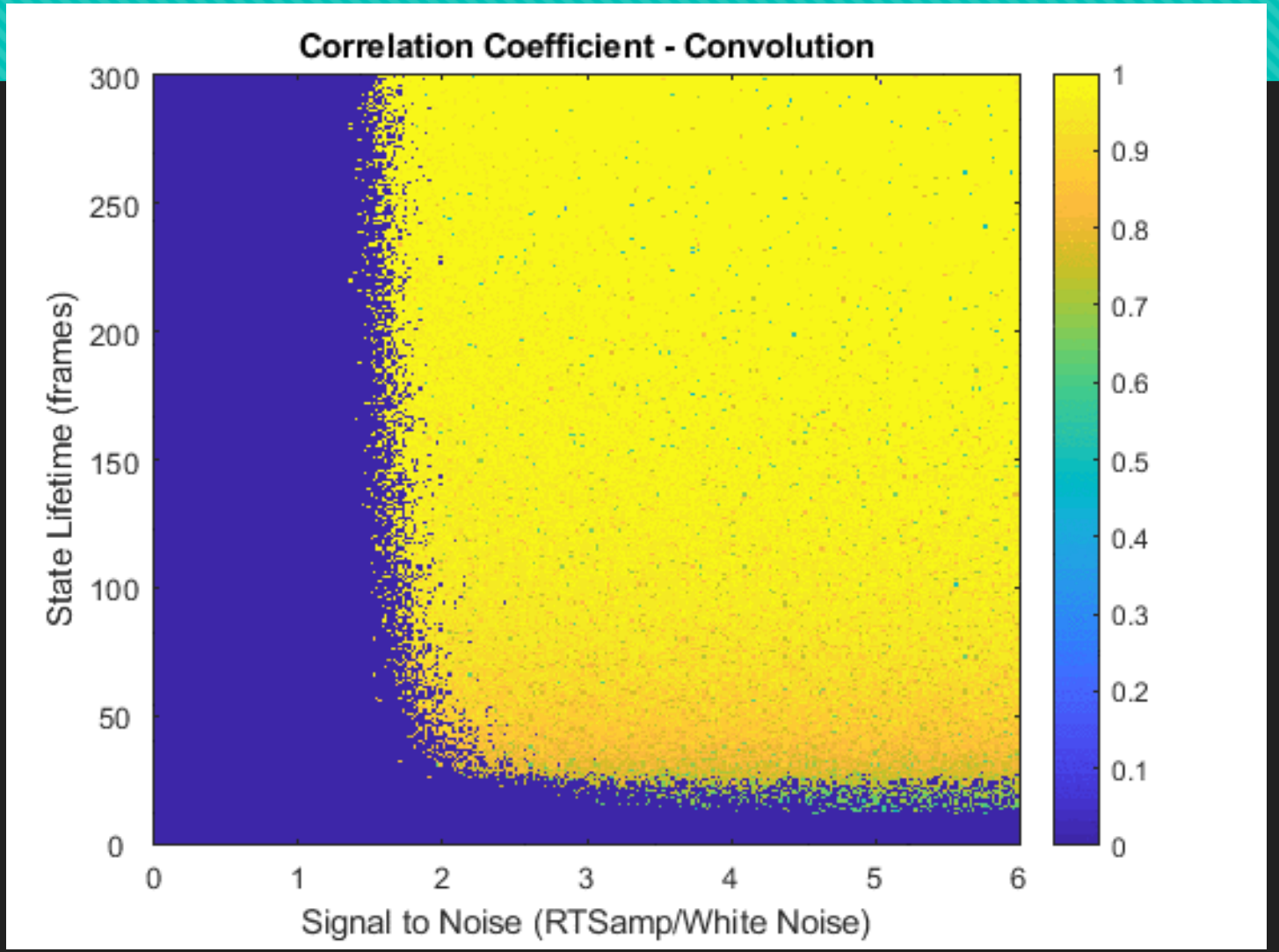


Correlation



Results – Correlation Comparison

- Direct comparison of the three methods
- All three perform well for signals $SNR > 2$ and $\tau > 50$ frames
- ML method performs reliably at half of those limits



Results - Discussion

- Convolution
  - Threshold: High
  - Correlation: High
- Wavelets
  - Threshold: Low
  - Correlation: Low
- Machine Learning
  - Threshold: Low
  - Correlation: High

Ideal

Correlation Range	Convolution Counts	Wavelet Counts	M.L. Counts
$0 < C_{xy} < 0.4$	6	5,342	527
$0.4 \leq C_{xy} < 0.6$	162	3,746	1,055
$0.6 \leq C_{xy} < 0.7$	674	2,772	647
$0.7 \leq C_{xy} < 0.8$	2,404	3,755	1,392
$0.8 \leq C_{xy} < 0.9$	5,271	6,460	4,209
$0.9 \leq C_{xy} < 0.99$	42,660	47,600	33,556
$C_{xy} \geq 0.99$	8,143	8,274	35,507

Thanks for listening

- Questions?

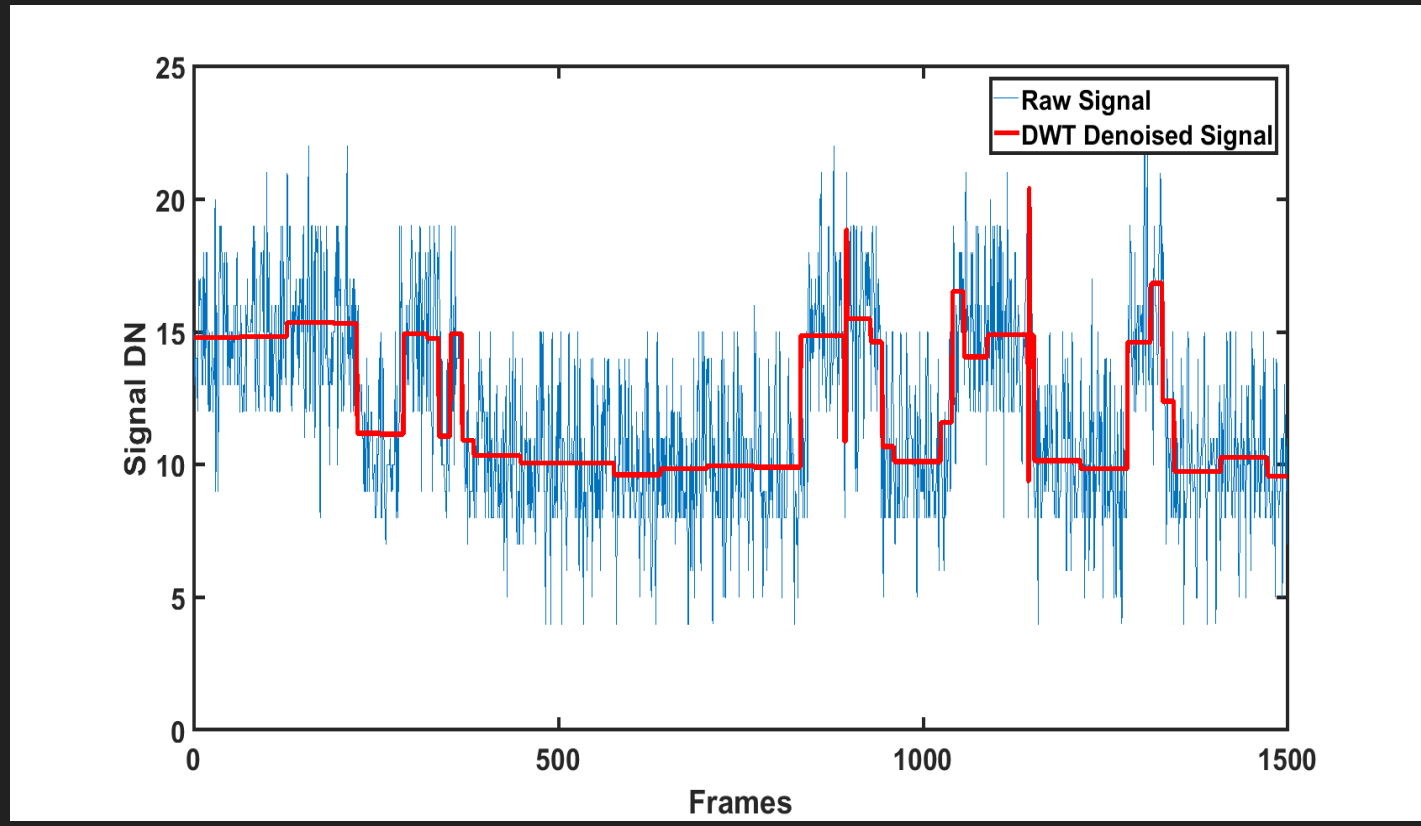
Haar Wavelet Analysis – DWT

- The DWT is similar to a microscope because it is repeatable
  - The trend sequence is treated as the new ‘mother’ signal

- Each time a subsequent transform is performed the ‘daughter’ sequences are of half size
  - The new ‘daughter’ sequence represents twice as many values from the original signal

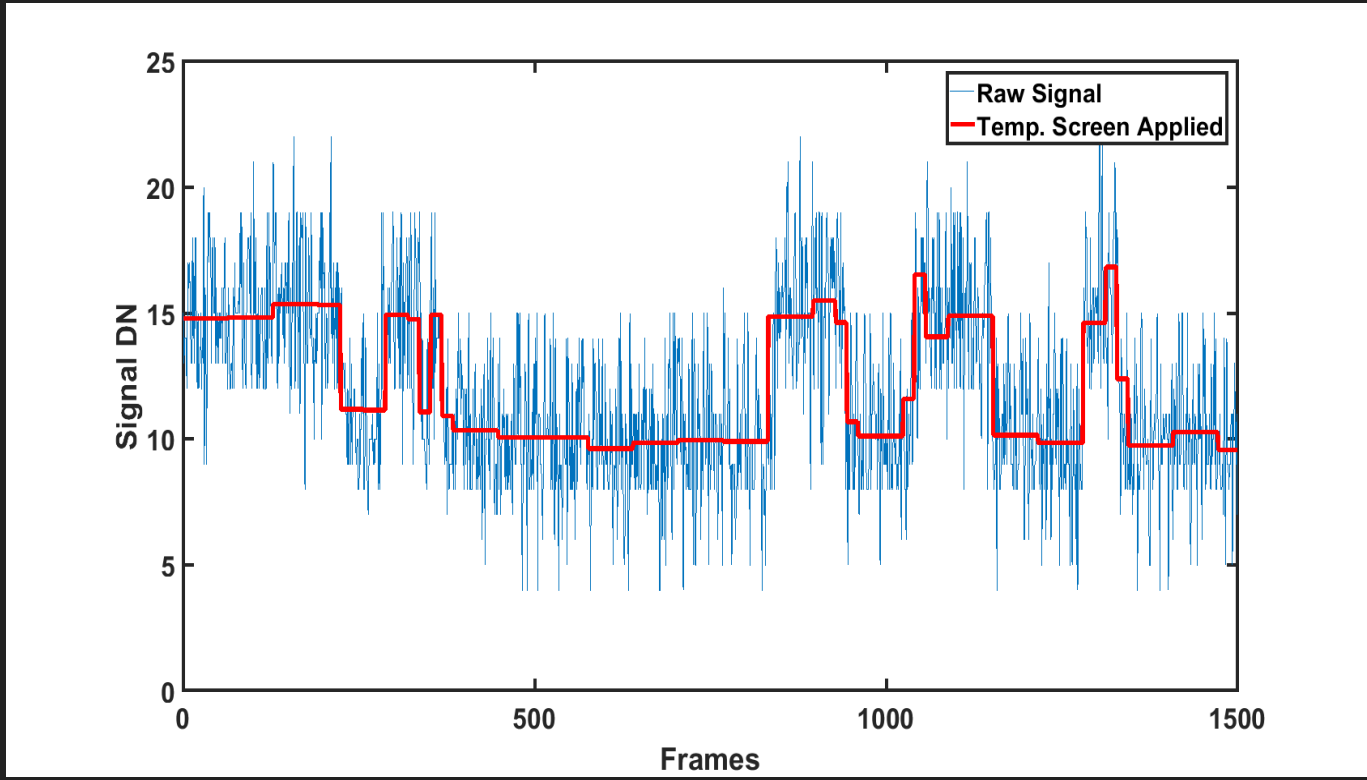
DWT Denoising Method

- The signal is run through the DWT denoising method as described
- The white noise is greatly reduced, but a few transients remain



DWT Denoising Method

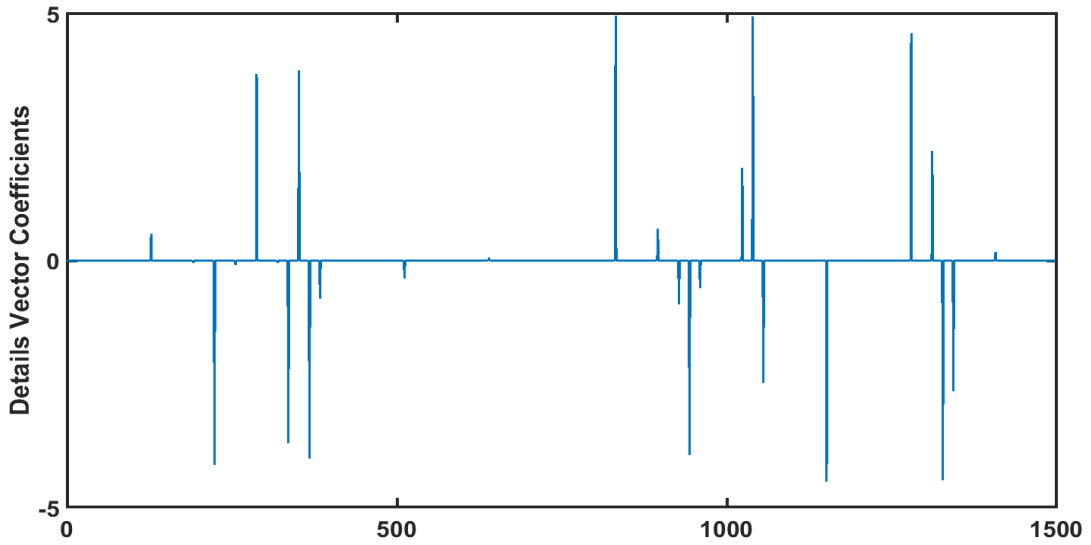
- To remove transients, a simple running comparison is implemented to verify the stability of a transition
- When a change in signal occurs at frame k, its value is compared to the next l frames where $l = 10$
- If the value is unchanged the transition is considered stable and left alone
- If the value changes is considered a transient, and is changed to the value at frame $k - 1$

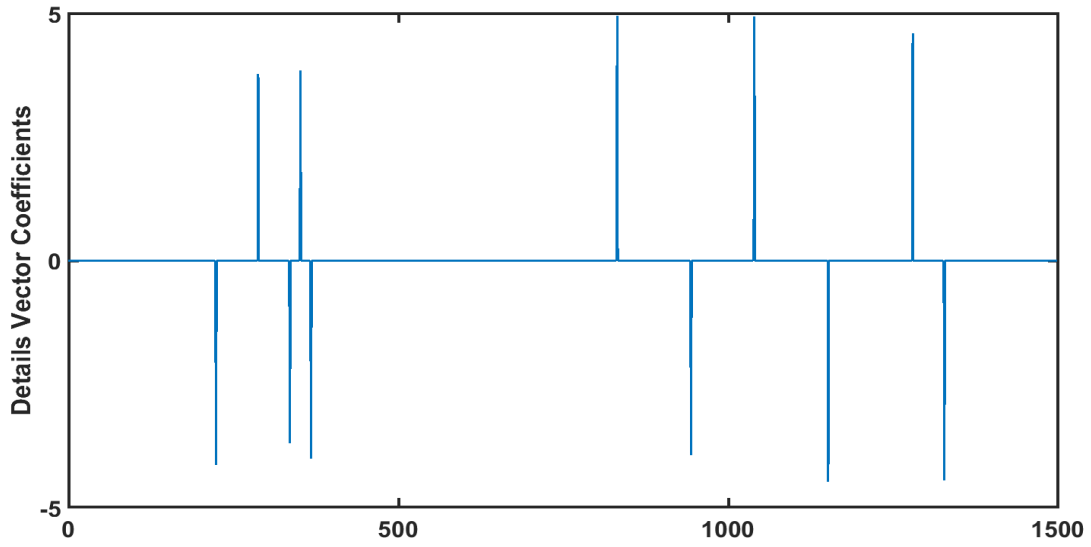


DWT Denoising Method

- Nearly all of the white noise is removed, but a few small changes remain
- A new details sequence is creating by subtracting each frame value by the previous frame value
- The new details sequence s is of $N-1$ where N is the size of the original signal
- Because the noise is already suppressed, the threshold need not be so discriminatory, as such

$$T_s = s_{MAX} * u_{0.75}$$





Machine Learning Method

- Signal classifier works similarly to image classifiers
- Developed in Python using a Keras wrapper over Tensorflow
- The model is trained on a set of 160,000 signals, half RTS and half non-RTS
- Convolutional layers extract prominent features and use them to differentiate RTS from non-RTS
- The convolutional layers use the 'relu' activation function while the final layer uses a sigmoid to force a choice between zero and one

$$Conv(32,12) \rightarrow Pool(3) \rightarrow Drop(0.5) \rightarrow$$

$$Conv(64,12) \rightarrow Pool(3) \rightarrow Drop(0.5) \rightarrow$$

$$Conv(128,12) \rightarrow Pool() \rightarrow Drop(0.5) \rightarrow$$

$$Fully\ Connected(1)$$

Machine Learning Method

- The autoencoder squeezes the noisy signal, extracting and prioritizing prominent features with convolutional and pooling layers
- It then expands the signal to original size and creates gaps with upsampling
- The model is trained to fill the gaps with a series of 'wrong' noisy signals and a corresponding set of 'correct' clean signals
- Each convolutional layer used the 'relu' activation function while the last layer uses the linear function to avoid zero values in the reconstructed signal

Layertype(NumFilters,KernalSize)

$$Conv(64,12) \rightarrow Pool(3) \rightarrow$$

$$Conv(32,12) \rightarrow Pool(3) \rightarrow$$

$$Conv(32,12) \rightarrow Upsample(3) \rightarrow$$

$$Conv(64,12) \rightarrow Upsample() \rightarrow$$

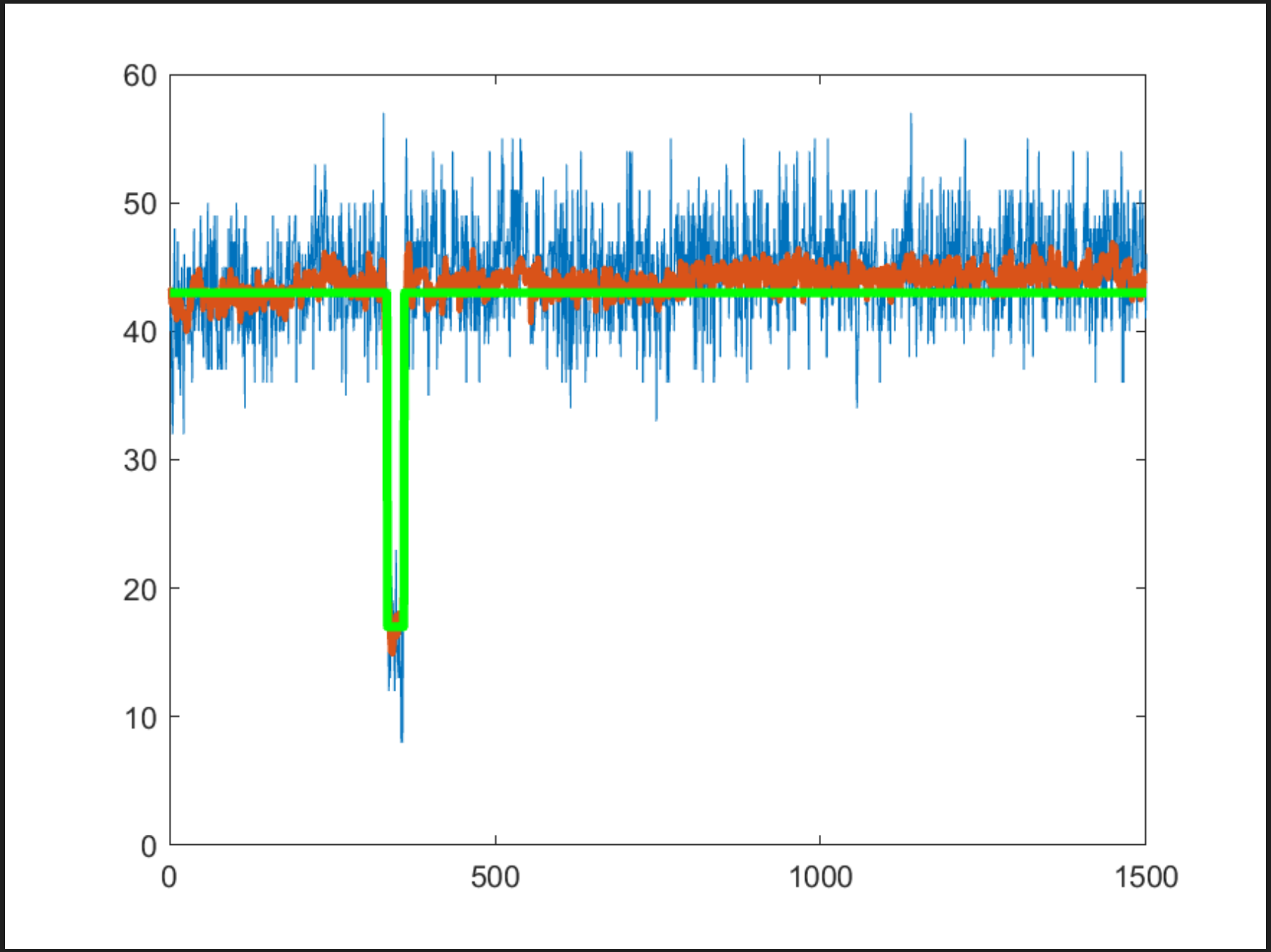
$$Fully\ Connected\ (1500)$$

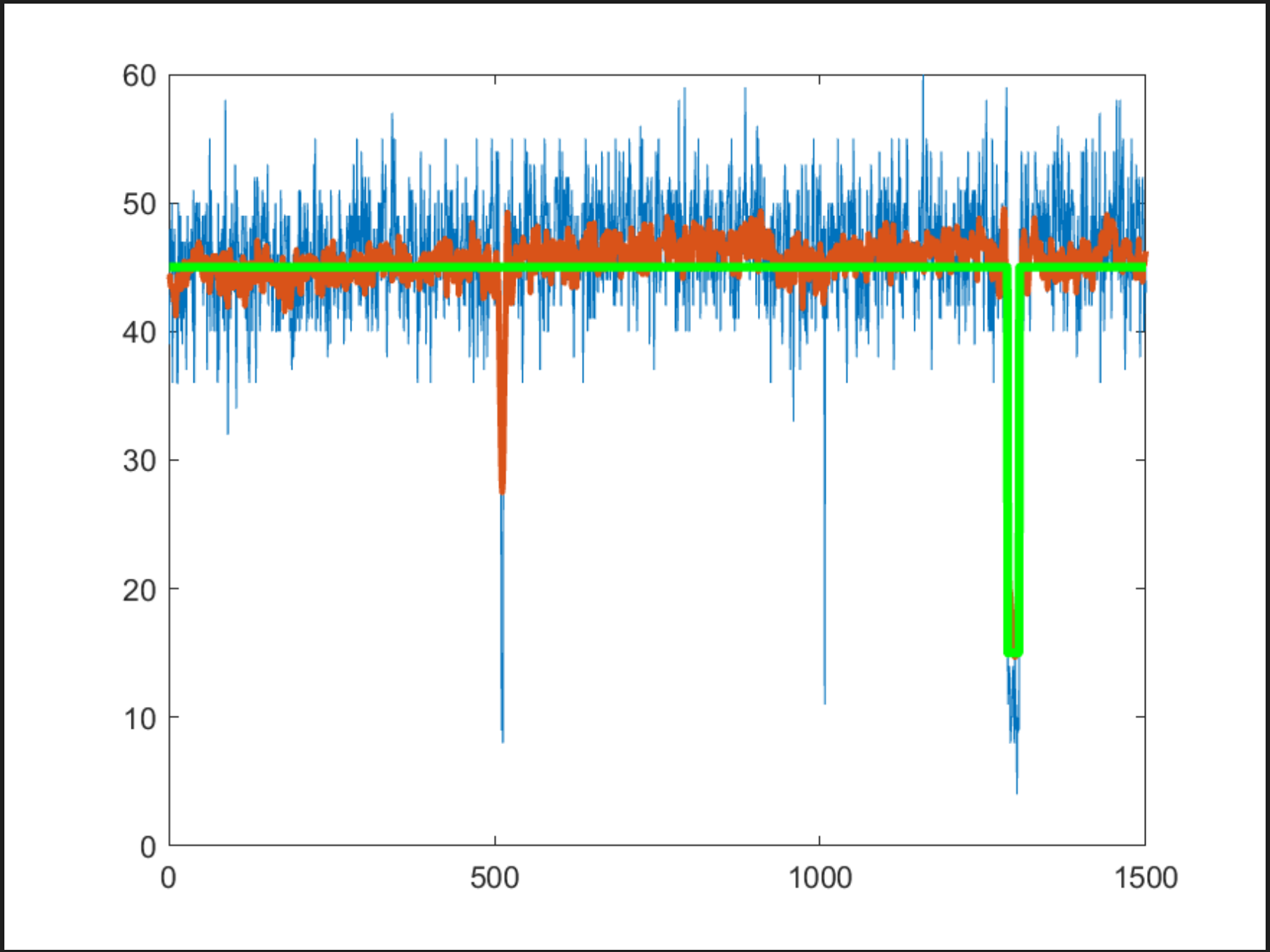
Machine Learning Method

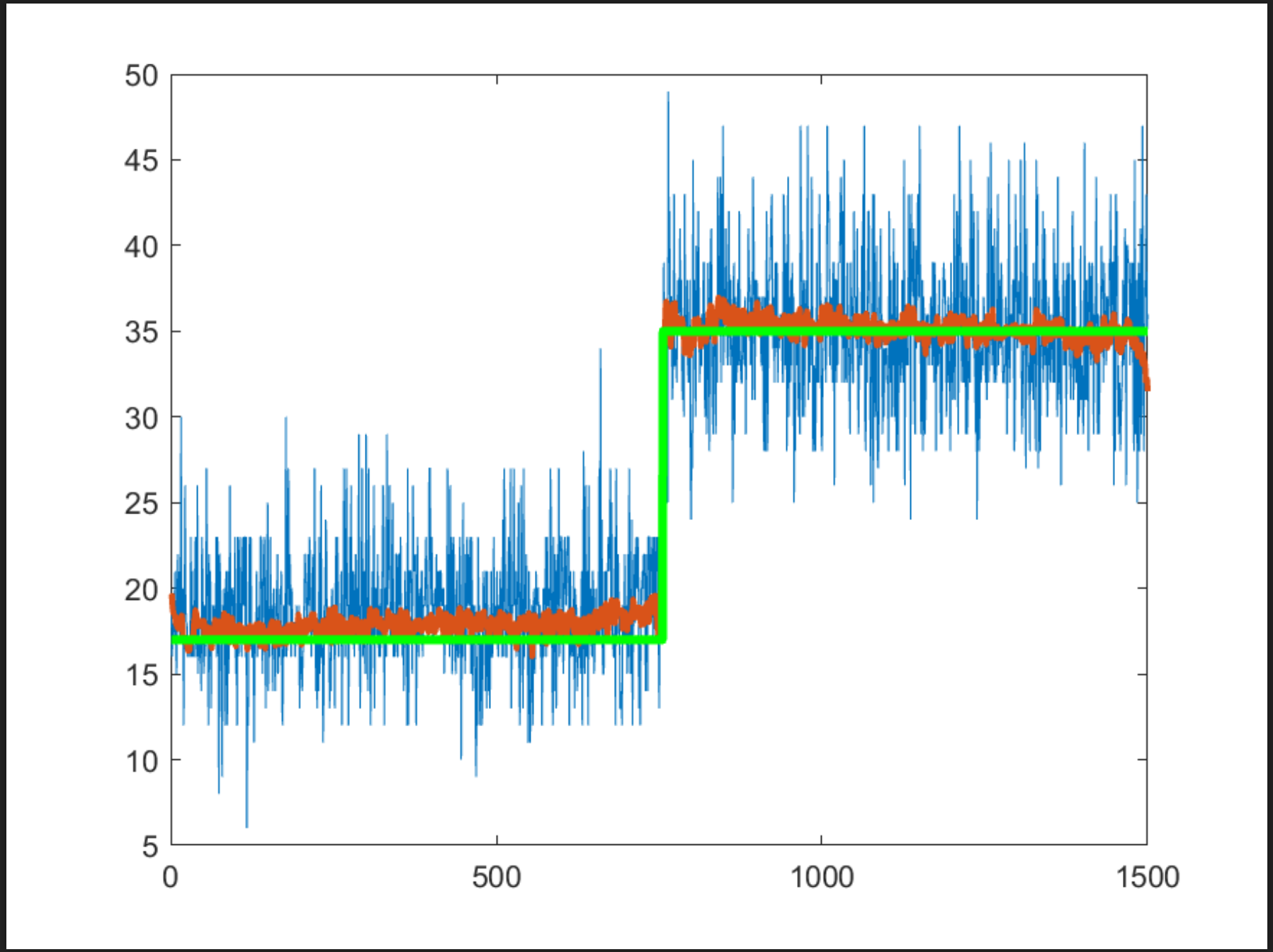
- The ML method requires special data preparation to work properly
- The training and input signals need to be rescaled between zero and one so that only the shape of the signals (RTS), not the magnitude, offers the defining signal characteristics
- Each value in the signal x is subtracted by a number just below the minimum of the signal creating a new vector x_s
- Then, x_s is divided by a number just above its maximum to create the scaled vector x_{sd}
- Because the scaling must be reversable, a key is maintained of scaling constants for each signal

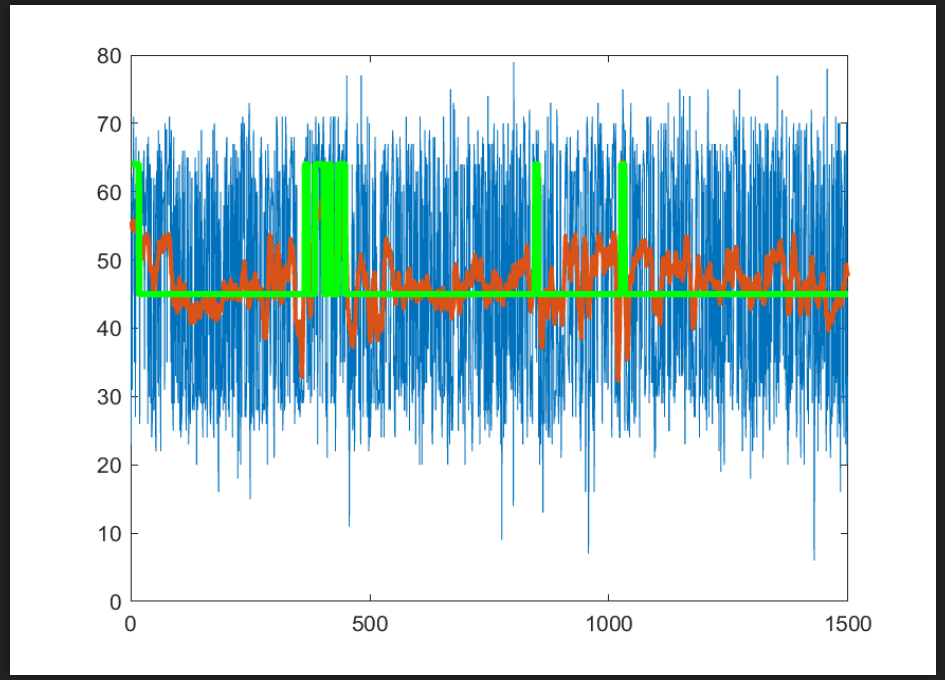
$$x_s = x - (0.99 * \min(x))$$

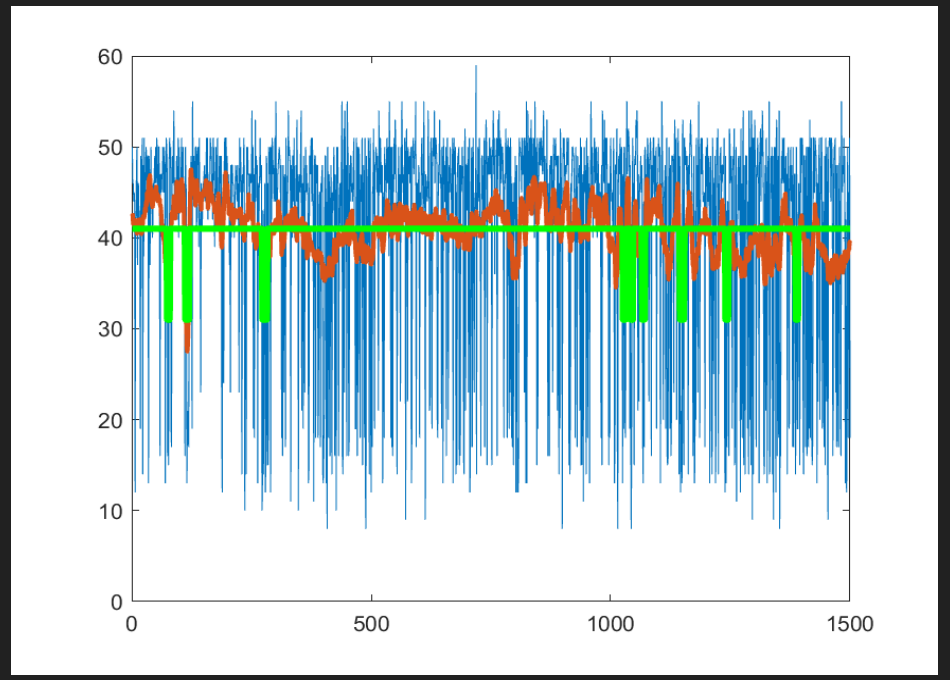
$$x_{sd} = x_s / (1.01 * \max(x_s))$$

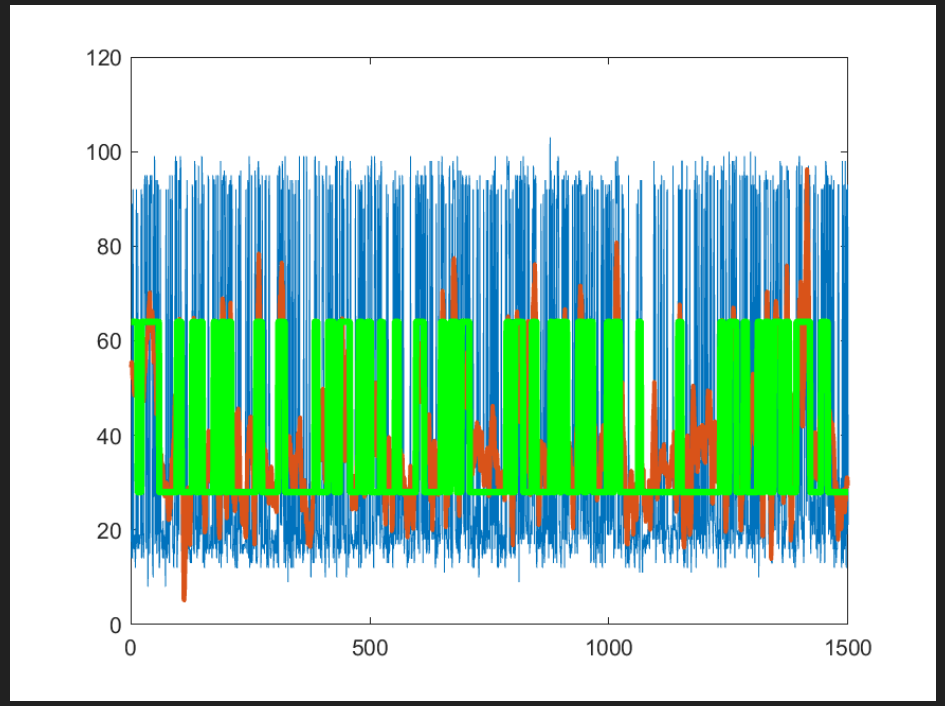

60
50
40
30
20
10
0
0
500
1000
1500

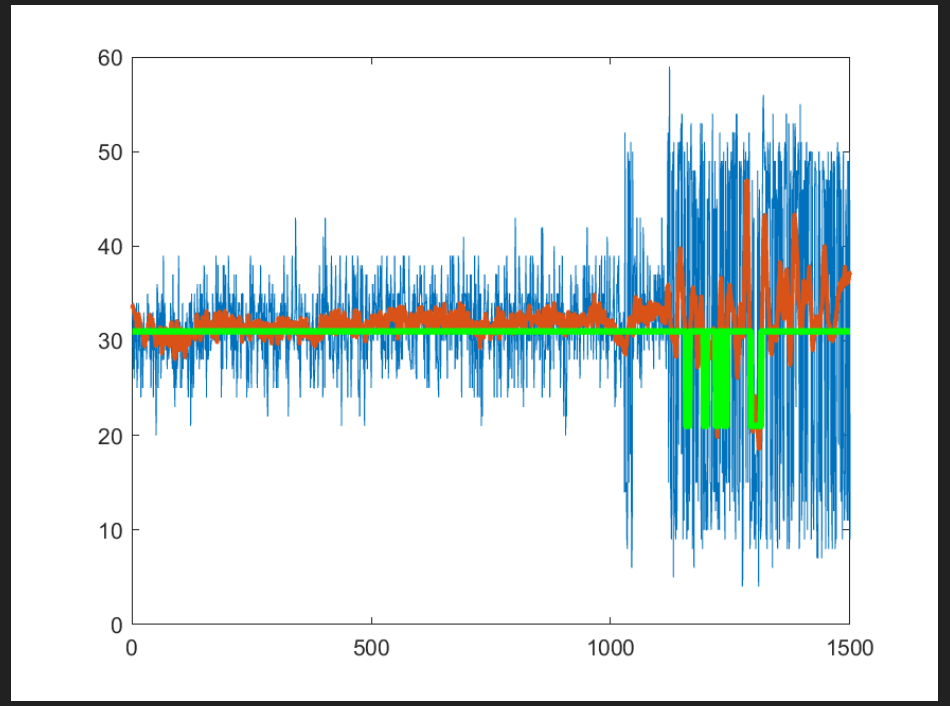

60
50
40
30
20
10
0
0
500
1000
1500


50
45
40
35
30
25
20
15
10
5
0
500
1000
1500


80
70
60
50
40
30
20
10
0
0
500
1000
1500


60
50
40
30
20
10
0
0
500
1000
1500


120
100
80
60
40
20
0
0
500
1000
1500


60
50
40
30
20
10
0
0
500
1000
1500