# Detection and Reconstruction of Noisy Bistable Stochastic Time Domain Signals Using Machine Learning

Ben Hendrickson, Ralf Widenhorn, Paul R. DeStefano and Erik Bodegom

*Department of Physics, Portland State University, Portland, OR 97201*

***Abstract*– Bistable stochastic systems are characterized by random discrete jumps in what otherwise would be a constant signal, usually measured as current or voltage. Here, a method of bistable stochastic signal detection and white-noise-free reconstruction is presented. This method is built on machine learning techniques for classification and denoising of one-dimensional time-series. The model is trained on a simulated dataset in order to provide certainty in the fidelity of corresponding 'clean' and 'noisy' signals, and tested on a different set of simulated signals. In addition, experimental data collected from a digital image sensor are used to provide a qualitative description of the model's efficacy.**

***Keywords*—machine learning, bistable stochastic signals, random telegraph signal, convolutional neural network, denoising autoencoder, signal reconstruction**

## 1. Introduction

### *1.1. Bistable Stochastic Signals*

Bistable stochastic signals or stochastic switching signals are the result of nonlinear dynamic processes that occur across many domains of physical science. This category of phenomena is frequently modeled as a double potential well with an energy barrier of some height in the center. The system remains in one of the states for some time until an event, or random energetic fluctuations, prompts the system over the barrier to the other state. The system returns to the previous state in the same way.

This phenomenon has been observed and studied extensively in biomolecular dynamics [1-5] where the kinetics of single molecule chemical reactions [6], and the mechanics of ion transport in biological membranes are modeled as two-state systems [7-9]. Single molecule reaction observations provide more precise measurements than those taken from a large collection. Ion transport is an important basic cellular process that provides insight into disease mechanisms. Enabling easier extraction of key parameters from these signals may enable researchers to develop more cost effective techniques to advance their fields. Many quantum mechanical systems are defined and analyzed as two-state stochastic dynamic processes including studies into electron shelving [10], strongly coupled atom/resonator systems [11], and detection of spin resonance for a single electron [12]. These studies shed light into the dynamics between quantum mechanical systems and interacting fields, and allow nondestructive measurement of quantum spin states. Semiconductor devices are susceptible to metastable defects often caused by radiation exposure. Since these defects stochastically switch on and off, they produce bistable current signals known as random telegraph signals [13-17]. Analyzing the amplitudes and state lifetimes of these signals provides insight into the class and locations of these defects.

Bistability is common, but has important implications. It has been shown that the superposition of bistable sources produces $1/f$ noise, a phenomenon that has been observed in everything from electronic devices to quasars [18]. The amplitudes and state lifetimes have different meanings for each system, but each provides information on a fundamental process in nature. In this paper we will describe a generic method for characterizing bistable signals and we will apply the technique to a large data set from a digital image sensor.

### *1.2. Basic Mathematics of Noisy Bistable Stochastic Signals*

Bistability is defined by stochastic transitions between one of two states, defined here as state 0 and state 1, the low and high states respectively. Represented mathematically, the state $s$ at some given time $t$ is either $s(t) = 0$ or $s(t) = 1$. Here, we will assume that any bistable signal has two independent noise contributors, the state transitions and Gaussian or white noise from other sources e.g., measurement. Since these noises are assumed independent from one

another in our model, their respective variances add together to determine the total noise of the signal such that:

$$\sigma_{SIG}^2 = \sigma_{Gaussian}^2 + \sigma_{BST}^2$$

The magnitude of the signal at some time $t$ is written as:

$$x(t) = x_0 + \epsilon(t) + A * s(t)$$

where $x_0$ is the signal value of the bottom state, $\epsilon(t)$ is the dark current Gaussian noise contribution at time $t$, $s(t)$ is the system state at time $t$, and $A$ is the state separation, or bistable amplitude. While the characteristic state lifetimes of a bistable signal may depend on a variety of factors, they are typically modeled as decaying exponentials, with the likelihood of the system flipping from one state to the other increasing with time. The model presented here is built on that assumption in order to account for the stochastic nature of these signals.

### *1.3. Machine Learning Classification*

The goal in building a classification model is to take a set of data made of many categories and accurately separate it into its different types. This classification model was trained to differentiate noisy bistable signals from non-bistable signals. A signal is represented as a vector and passed through various layers of operators or functions to produce, in this case, a single output (zero for bistable signals or one for non-bistable signals). This is similar to the way that image classification is performed, and similar to machine learning classification methods previously used for one-dimensional digital signals [19-23]. A typical convolutional classification model [24] will include: convolutional, pooling, dropout, and fully connected layers, here, each is addressed in turn.

#### *1.3.1. Convolutional Layers*

Convolutional layers apply filters to extract prominent features that are representative of distinctive characteristics, such as state transitions. As the signal is passed forward through the network, each neuron (or, filter or kernel) is convolved with the signal creating a feature map that is the same size as the input [25]. Finally, an activation function is applied to each filter. This function ensures that each convolution is, in the end, a non-linear operation. The activation function used here is the rectified linear unit (ReLU) function [26] which returns a zero for negative inputs and the input value itself for positive inputs [27], or with $x$ the input:

$$f(x)_{ReLU} = \max\{0, x\}.$$

Convolutional layers that are stacked after the initial layer will operate upon the feature maps produced from the previous layers. The shapes of the filters, or weights of the neurons, are continuously changed during the training process by backpropagation, to be discussed later.

#### *1.3.2. Pooling Layers*

Pooling layers reduce the dimensionality of the vector by down-sampling the feature maps. Pooling layers typically appear directly following a convolutional layer. While there are a variety of pooling techniques, our classification scheme uses "max-pooling." Essentially, max-pooling is a form of compression that inspects a section of a feature map, say elements 7, 8, and 9, finds the largest value amongst the three, and tosses the other two values out. Pooling not only eases the computational stress of training a model by reducing the number of parameters, but also provides spatial invariance of important features [28].

#### *1.3.3. Dropout Layers*

Dropout layers turn off a percentage of neurons, or filters during training. This prevents filters from becoming dependent on the presence of neighboring filters to optimize the model. This interdependence leads to overfitting. An overfit model will perform very well on the data it is trained on, but will perform poorly on data in general [29].

#### *1.3.4. Fully Connected Layers*

The final layer in a classification model is a fully connected layer. Each neuron in this layer, as the name suggests, is connected to every output from the previous layer. This layer forms a vector where each element represents a confidence score corresponding to a distinct class. This model has a final layer of size one, where the one neuron represents the confidence of a signal containing bistability.

### *1.4. Classifier Training*

When the model is first initialized for training the coefficients of each filter, or the shape of each filter, are randomized. Then, one by one, members of the training set are passed through the network, and assigned a confidence of bistable versus non-bistable. Because this is supervised training the confidence score is checked against the given label for the signal, 0 for bistable and 1 for non-bistable signals. The error of the confidence score is calculated by using the binary cross-entropy loss function, defined below, and improved by updating the filter and activation weights by means of backpropagation [30].

#### 1.4.1. *Binary Cross Entropy*

The loss function used for classification is binary cross entropy, $E$. Here, $t$ is the target label, 0 for bistable, 1 for non-bistable. $y$ is the probability of the signal being non-bistable according to the model. Notice that if the target and probability are close to one another the error is close to zero [31].

$$E = -(t\log(y) + (1-t)\log(1-\mathrm{y}))$$

### 1.5. *Denoising Autoencoder*

Once the signal is run through the classification model, and if it is determined to have bistability, the signal has its white noise component suppressed by means of a denoising autoencoder (DAE). The autoencoder shares some features of the classifier, e.g., convolutional layers, pooling layers, etc. Rather than attempting to identify the kind of signal (bistable vs. non-bistable) it takes the noisy signal as an input and attempts to return the denoised one. In this case, the autoencoder takes a bistable signal with Gaussian noise, and returns a signal with suppressed noise.

To train our DAE, a noise-free bistable signal, $x$, is simulated. Then, Gaussian noise is added over the top to produce the noisy signal $\tilde{x}$. This signal is then encoded by running it through convolutional and pooling layers to extract pertinent features and compress it. The now encoded signal, or rather feature map, is then decoded by again running it through convolutional layers, but now using up-sampling rather than pooling. The up-sampling returns the signal to its original size by adding elements with value equal to zero. Adding these zeros forces the autoencoder to learn the important features of the non-zero values in order to 'fill in the gaps'. Finally, the signal is passed through a fully connected layer that produces a denoised reconstruction of the input signal $\hat{x}$ as seen in figure 1. Just like with the classifier, the result is measured against the ground truth, or in this case the original clean signal $x$ [32], by again using a loss function. For the autoencoder the loss function is a simple mean squares error comparison between each element of the clean signal $x$ and the denoised $\hat{x}$ [33-36].

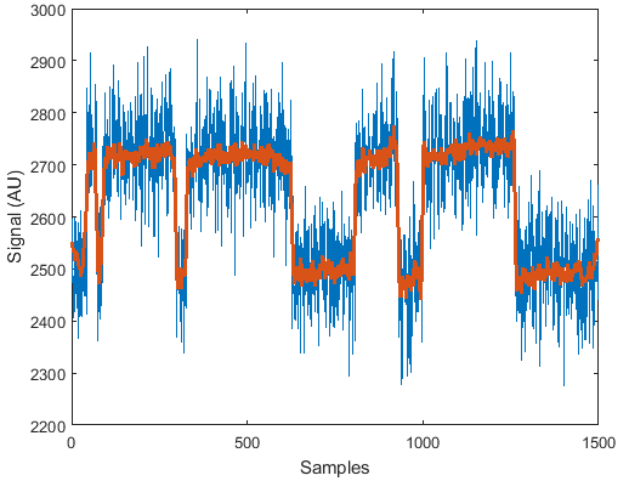


Figure 1: A stochastic bistable signal before (blue) and after (orange) passing through DAE. A significant increase in signal to noise is obvious.

## 2. Model Topology and Algorithm Methodology

This bistable signal detection and reconstruction schema was developed in Python and MATLAB using the concepts outlined in the previous section. All machine learning modeling was performed in Python, while the data preparation and reconstruction finalization was performed in MATLAB. This section outlines the specific choices made with respect to model architecture, and signal processing to carry out the goal of accurate detection and reconstruction.

### 2.1. *Classifier Summary*

The classification modeling network, shown in figure 2, was developed in Python using Keras [37] as a wrapper over TensorFlow [38]. The layers are structured as such: $Conv(32) \rightarrow Pool(3) \rightarrow Drop(0.5) \rightarrow Conv(64) \rightarrow Pool(3) \rightarrow Drop(0.5) \rightarrow Conv(128) \rightarrow MaxPool() \rightarrow Drop(0.5) \rightarrow Fully\ Connected(1)$. The convolutional layers have 32, 64, and 128 filters respectively with the size of each filter set to 12. Each uses the ReLU activation function. The first two pooling layers take the maximum value, while the last takes an average. The dropout rate is set to 50%. The final layer uses the sigmoid activation function. Training was carried out over five epochs. Figure 2 shows the number and size of the feature maps resulting from the convolution and pooling operations, as well as the final fully connected layer which contains the bistability confidence score.

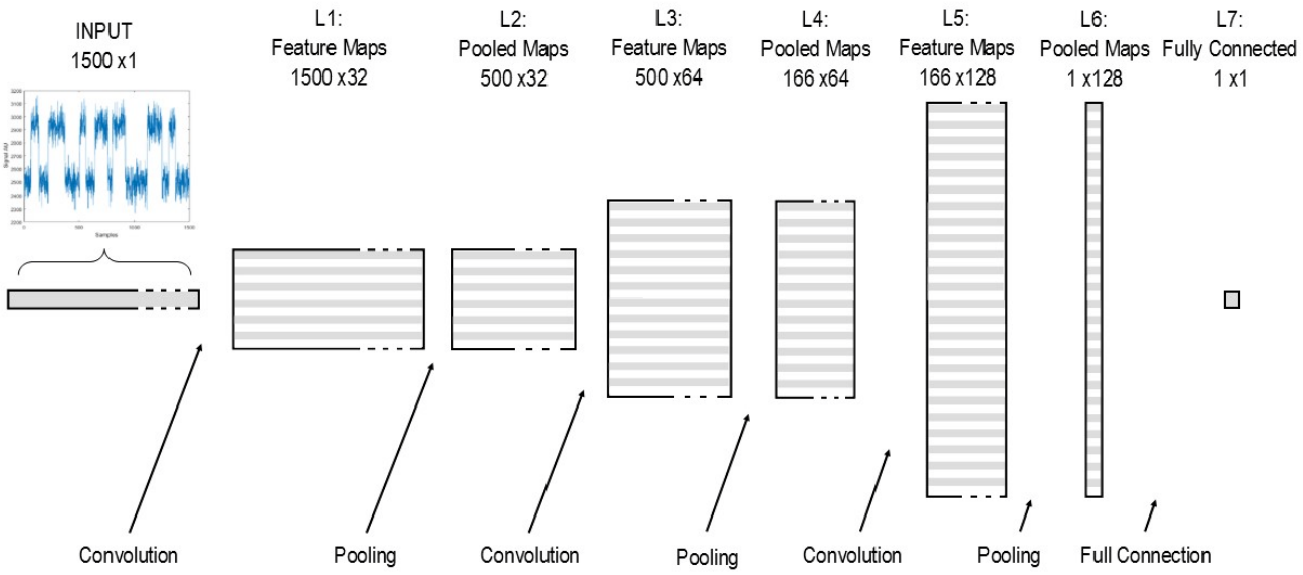


Figure 2: Topology of the bistability classification model. Each signal is passed through 32 convolutional filters to create 32 feature maps. The ReLU activation function is then applied. Those activated feature maps are then pooled down to size 500, passed through the next convolutional layer and activation function to creates a block of 64 feature maps of size 500. The process is repeated once more to create 128 feature maps of size 166 which undergo maxpooling where each feature map is reduced to a single value, its maximum. Those single value maps are then fully connected to the final layer, a single value, which represents the bistability confidence score.

### *2.2. Autoencoder Summary*

The denoising autoencoder model, shown in figure 3, was likewise built in Python using Keras as a wrapper over TensorFlow. Its layers are structured as such: Conv(64)→Pool(3)→Conv(32)→Pool(3)→Conv(32) →Upsample(3)→Conv(64)→Upsample(3)→ Fully Connected (1500). The convolutional layers have 64, 32, and 64 filters respectively while the size of each filter is again set to 12. Each uses the rectified linear unit activation function. The final fully connected layer uses a linear activation function. Training was carried out over five epochs. The squeezing and expansion of the denoising autoencoder, as well as the denoising effect can be seen in figure 3.

### *2.3. Training Considerations*

One of the more problematic aspects of stochastic bistability is that there are no well-defined limits on amplitude or state lifetime. If either of these key characteristics is sufficiently small it is difficult to distinguish whether or not a signal has bistable state transitions, let alone attempt to reconstruct it without Gaussian noise. It then becomes necessary to create a training set with realistic bistable signals that feature a wide variety of amplitudes and state lifetimes. Simulated signals and noise augmentation have been used previously for training networks related to variety of applications [39-43]. The training set created here has amplitudes from 1 to 450 arbitrary units (AU), spaced evenly by intervals of 1.5 AU, and state lifetimes spaced evenly from 1 to 300 samples, as shown in figure 4. Transitions between bistable states are determined by a decaying exponential probability so that they remain stochastic, but average out to the appropriate state lifetime. Lifetimes for the high and low states were set equal to each other for all bistable signals.

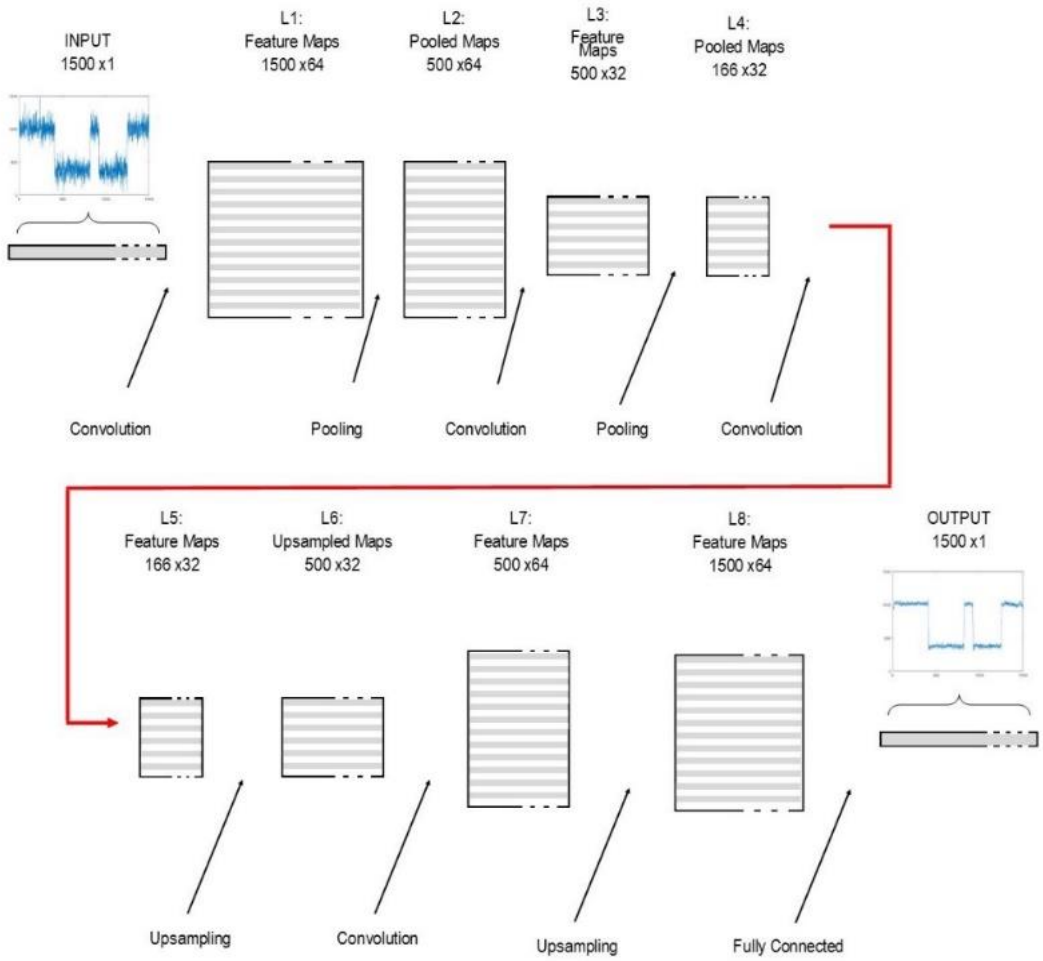


Figure 3: Topology of the denoising autoencoder. Each signal is passed through the convolutional layers similar to the classifier. After the 3rd convolution the feature maps are upsampled rather than pooled to expand them back to their original size. By upsampling, the model is forced to learn important features of the data in question, which leads to a denoised version of the input signal.

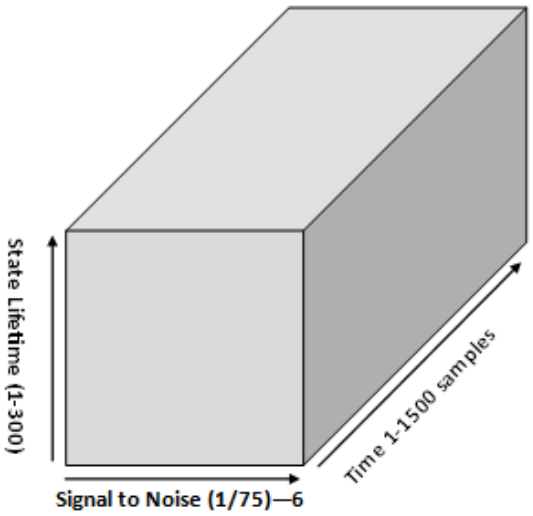


Figure 4: The structure of the training set

Each signal then has Gaussian noise added, with a standard deviation of 75 AU as shown in figure 5. A new quantity is defined for an approximation of the signal to noise ratio which is simply

$$SNR_{BST} = A/\sigma_{Gaussian}$$

where $A$ is the amplitude between states and $\sigma_{Guassian}$ is the Gaussian noise. The range of $SNR_{BST}$ for the training dataset spans from $\frac{1}{75}$ to 6. To train the classifier to separate bistable from non-bistable signals an additional collection of non-bistable signals, Gaussian noise only, were produced. . In total 180,000 signals were created, 90,000 with only Gaussian noise and 90,000 with Gaussian noise and bistable transitions.

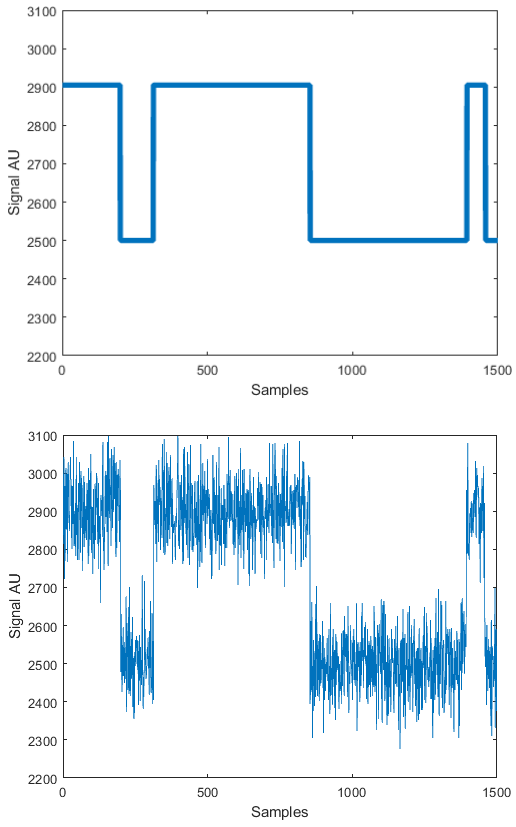


Figure 5: A simulated bistable signal before and after adding Gaussian noise with $SNR_{BST} = 5.33$.

Finally, before training the machine learning models the signals must be scaled, so the shape of the signal, not the magnitude determines the weights of the filters. It was determined that each signal should lie between zero and one, so each signal $x$ is subtracted by a value just below the minimum to create $x_s$

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

$x_s$ is then divided by a value just above its maximum to create $x_{sd}$

$$x_{sd} = \frac{x_s}{d}\ ;\ d = 1.01 * \max(x_s)$$

Since the model is trained on scaled signals, any real data processed by it must undergo the same scaling. In order to ensure the mean signal values remain unchanged this scaling must be reversible, so a key is maintained that records $s$ and $d$ for each signal $x$.

*2.4. Gaussian fit level finding*

Recall the total noise of a bistable signal is defined as: $\sigma_{SIG}^2 = \sigma_{Gaussian}^2 + \sigma_{BST}^2$ which shows the Gaussian noise and bistable transition noise are uncorrelated to one another. Therefore, a histogram of a bistable signal, before and after the autoencoder denoising, will be composed of the sum of Gaussian peaks, one for each state. The reconstruction of a bistable signal is completed by taking a histogram of the autoencoder result, and fitting [44] it as a sum of two Gaussians as shown in figure 6. The new clean signal, figure 7, is created by snapping each element of the autoencoder to whichever peak value from the fitted histogram (see figure 6) is closest to that element. From here the state separation amplitudes and state lifetimes are simply collected.

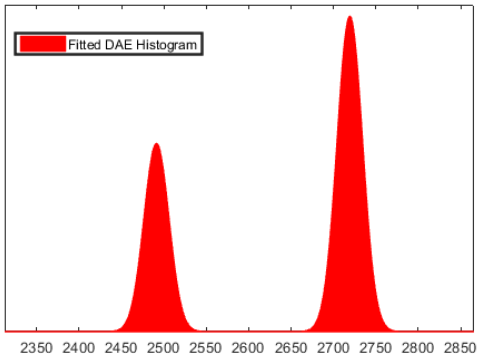


Figure 6: The fitted histogram of the autoencoder results. The final reconstruction uses the values where peaks occur.

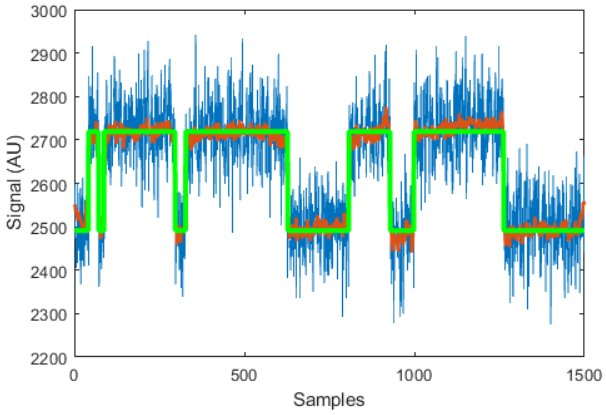


Figure 7: The final reconstruction of a stochastic bistable signal. There are a total of 2 values for the entire signal, with zero Gaussian noise.

## 3. Results and discussion

### *3.1. Simulated Dataset*

In order to measure the efficacy of the classification model, a validation test was carried out on two additional sets of simulated signals. This test inputs a sample signal to the model, and records the number of correct and incorrect inferences. Like the training sets, each is composed of 90,000 signals. The set of bistable signals has state lifetimes that span from 1 to 300 samples, and amplitudes such that the $SNR_{BST}$ runs from $\frac{1}{75}$ to 6. All signals are scaled as described above before running them through the algorithm.

The algorithm detected 83.5% of the bistable signals, and recorded zero false positives from the non-bistable test set. It works remarkably well on bistable signals that have a $SNR_{BST} > 1.5$ and lifetimes longer than about 20 samples as seen in figure 8.

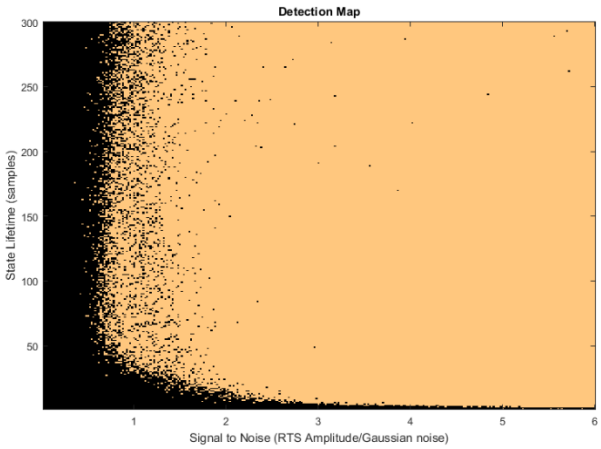


Figure 8: The bistable signal detection map. Black areas are where the detection model failed.

Each signal that passed detection was then scored on the quality of reconstruction by means of the sample correlation coefficient. This is a great advantage of testing on a simulated data set since each reconstruction can be directly compared to the original clean signal. The sample correlation coefficient is calculated as

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

where $C_{xy}$ is the correlation coefficient or score, $x_i$ is the value of the $i^{th}$ element in the reconstructed signal, $\bar{x}$ is the mean of the reconstructed signal, $y_i$ is the value of the $i^{th}$ element in the original clean signal, and $\bar{y}$ is the mean of the original clean signal. The coefficient lies between $-1$ and 1 where $-1$ is perfectly anticorrelated, 0 is uncorrelated and 1 is perfectly correlated. In practice negative scores are possible, but exceedingly rare. Nearly all bistable signal detections resulted in a highly accurate reconstruction as seen in figure 9 and punctuated by table 1. The mean correlation score for detected bistable signals is 0.978 [45].

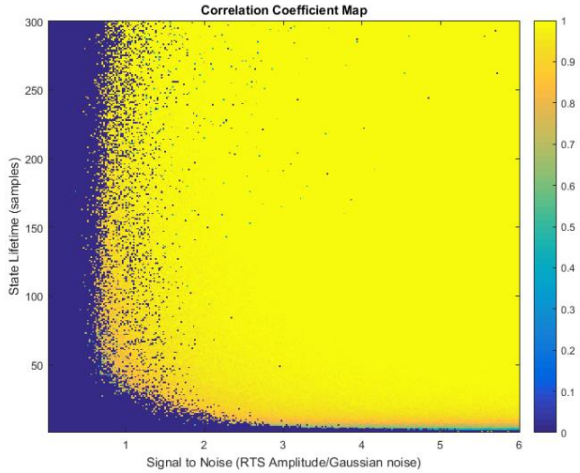


Figure 9: The sample correlation coefficient map. For the vast majority of signals that passed detection, reconstruction is near perfect.

| Correlation Range | Counts |
|---|---|
| $0 < C_{xy} < 0.4$ | 68 |
| $0.4 \leq C_{xy} < 0.6$ | 257 |
| $0.6 \leq C_{xy} < 0.7$ | 251 |
| $0.7 \leq C_{xy} < 0.8$ | 730 |
| $0.8 \leq C_{xy} < 0.9$ | 2,523 |
| $0.9 \leq C_{xy} < 0.99$ | 21,613 |
| $C_{xy} \geq 0.99$ | 49,659 |

Table 1: The correlation score counts highlight the quality of reconstruction for a great majority of pixels.

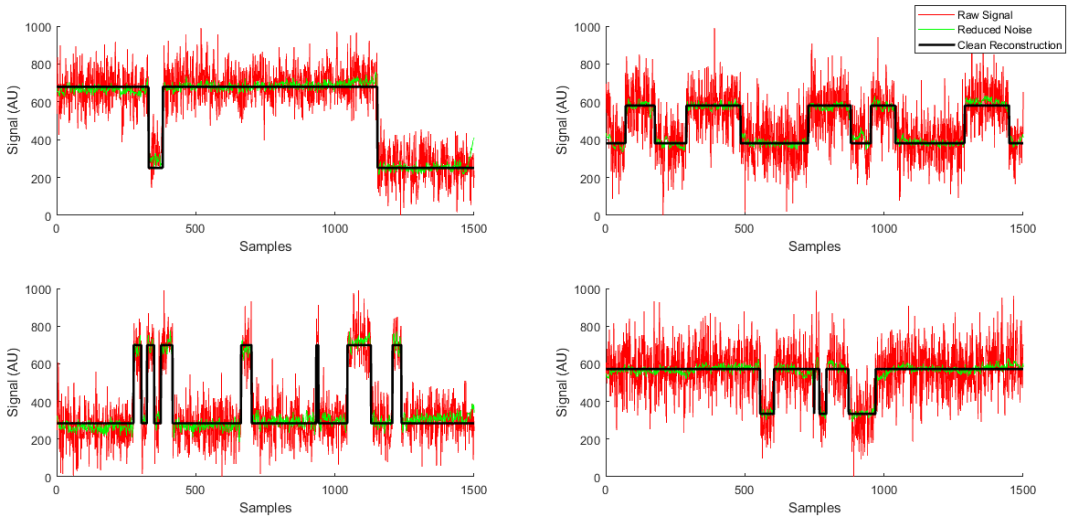


Figure 10: Reconstructions of four randomly selected RTS pixels.

### *3.2. RTS Image Sensor Data*

A commonly observed bistable phenomenon is Random Telegraph Signal (RTS) noise in silicon devices, particularly image sensors. RTS noise in sensors is typically the consequence of exposure to radiation. The bistability is produced by discrete changes in the generation rate of leakage current, known in image sensors as dark current [46]. RTS noise in image sensors has been previously analyzed in a number of ways. Usually a lengthy time series is created for pixels of interest by collecting many (a few thousand) frames at regular intervals. This series is then analyzed to identify RTS behavior and extract characteristics of interest. RTS is one of the major noise sources that remains difficult to mitigate in both CMOS and CCD image sensors.

To provide an example of the results this detection and reconstruction algorithm may yield frames were collected from a charge-coupled device (CCD) and stacked together. Then the temporal response from individual pixels was analyzed and reconstructed as if each were an independent device (see figure 10). The sensor used is a SITe SI-033AF frontside illuminated 1 mega-pixel CCD ($1024x1024$) [47]. Frames were taken in dark conditions with 10 second integration time at 305 K.

As illustrated by the four random RTS pixels shown in figure 10, the collected CCD dark current data showed that the method described here is capable of creating quality noise free reconstructions of bistable stochastic signals. It was expected, perhaps naively, that the wide range of state lifetimes and amplitudes characteristic of RTS signals would cause some issues, but none have arisen yet. While this result is promising, additional validation is required by testing it on different sources of data.

## 4. Conclusion

A machine learning based algorithm is presented for the reconstruction and analysis of stochastic bistable signals. The algorithm uses a convolutional classifier for the identification of state transitions, and a convolutional denoising autoencoder to increase the bistable amplitude signal to noise level. A histogram of the decoded signal values is taken and fit to the sum of two Gaussians. Finally, the signal is reconstructed by snapping each value of the decoded signal to the nearest peak location.

Quantitatively, the algorithm was shown to be successful by running it on a set of simulated bistable and non-bistable signals. It detected over 83% of the bistable signals, and only consistently failed on signals with a $SNR_{BST} < 1$. Reconstruction for signals that passed detection is exceptional, reaching an almost perfect correlation coefficient of 0.99 or greater for nearly 66% of detected signals.

Qualitatively, the algorithm proved capable on a set of data collected by taking dark frames with a CCD image sensor. In the case of image sensors in particular, additional steps need to be taken to address issues stemming from cosmic rays and thermal fluctuations, but once mitigated near perfect reconstruction of an RTS signal is expected.

## References


[1] E. Rhoades, E. Gussakovsky, G. Haran, Watching proteins fold one molecule at a time. Proc. Natl. Acad. Sci. USA. 100 (2003) 3197-3202.

[2] H.S. Chung, K. McHale, J. M. Louis, W. A. Eaton, Single-Molecule Fluorescence Experiments Determine Protein Folding Transition Path Times, Science. (2012) 981-984.

[3] L. Edman, Z. Földes-Papp, S. Wennmalm, R. Rigler, The fluctuating enzyme: a single molecule approach, Chem. Phys., Volume 247, Issue 1 (1999) 11-22.

[4] H.P. Lu, L. Xun, X.S. Xie, Single-Molecule Enzymatic Dynamics, Science. (1998) 1877-1882.

[5] X. Zhuang, H. Kim, M.J.B. Pereira, H.P. Babcock, N.G. Walter, S. Chu, Correlating Structural Dynamics and Function in Single Ribozyme Molecules, Science, (2002) 1473-1476.

[6] K. Velonia, O. Flomenbom, D. Loos, S. Masuo, M. Cotlet, Y. Engelborghs, J. Hofkens, A.E. Rowan, J. Klafter, R.J.M. Nolte, F.C. de Schryver, Single enzyme kinetics of CALB catalyzed hydrolysis, Angew. Chem. Int. Ed. Engl., 44 (2005) 560-564.

[7] R.C. Klipp, N. Li, Q. Wang, T.A. Word, M. Sibrian-Vazquez, R.M. Strongin, X.H.T. Wehrens, J.J. Abramson, EL20, a potent antiarrhythmic compound, selectively inhibits calmodulin-deficient ryanodine receptor type 2, Heart Rhythm, 15 (2018) 578-586.

[8] V. Krishnamurthy et al., Gramicidin Ion Channel-Based Biosensors: Construction, Stochastic Dynamical Models, and Statistical Detection Algorithms, IEEE Sensors Journal, 7 (2007) 1281-1288.

[9] S. Oiki, H. Shimizu, M. Iwamoto, and T. Konno, Single-Molecular Gating Dynamics for the KcsA Potassium Channel, Single-Molecule Biophysics, (2011) 147-193.

[10] D. R. Crick, S. Donnellan, D. M. Segal, and R. C. Thompson, Magnetically induced electron shelving in a trapped Ca+ ion, Phys. Rev. A, 81-052503 (2010) 1-4.

[11] S. Reick, K. Mølmer, W. Alt, M. Eckstein, T. Kampschulte, L. Kong R. Reimann, A. Thobe, A. Widera and D. Meschede, Analyzing quantum jumps of one and two atoms strongly coupled to an optical cavity, J. Opt. Soc. America B, 27 (2010) A152-A163.

[12] J. Jin, J. Guo, J. Luo, X. Li, and Y. Yan, Quantum trajectory analysis for electrical detection of single-electron spin resonance, Phys. Rev. B, 73 (2005) 125312.1-5.

[13] P.L. Leonard and S.V. Jaskolski, An investigation into the origin and nature of "Popcorn noise", Proceedings of the IEEE, 57 (1969) 1786-1788.

[14] J. Bogaerts, B. Dierickx and R. Mertens, Random telegraph signals in a radiation-hardened CMOS active pixel sensor, IEEE Trans. on Nucl. Sci., 49 (2002) 249-257.

[15] W.H. Chard and P.K. Chaudhari, Characteristics of burst noise, Proc. IEEE, 53 (1965) 652.

[16] E. Simoen, B. Dierickx, C. L. Claeys, and G. J. Declerck, Explaining the amplitude of RTS noise in submicrometer MOSFETs, IEEE Trans. Electron Devices, 39 (1992) 422.

[17] V. Goiffon, P. Magnan, P. Martin-Gonthier, C. Virmontois, and M. Gaillardin, New source of random telegraph signal in CMOS image sensors, International Image Sensor Workshop, Hokkaido, Japan, (2011)

[18] W.H. Press, Flicker noises in astronomy and elsewhere, Comments on Modern Physics, Part C - Comments on Astrophysics, 7 (1978) 103-119.

[19] Q. Wang, P. Du, J. Yang, G. Wang, J. Lei, C. Hou, Transferred deep learning based waveform recognition for cognitive passive radar, Signal Process., 155 (2019) 259-267.

[20] T. Ölmez, Z. Dokur, Classification of heart sounds using an artificial neural network, Pattern Recognition Letters, 24 (2003) 617-629.

[21] K. Basu, V. Debusschere, A. Douzal-Chouakria, S. Bacha, Time series distance-based methods for non-intrusive load monitoring in residential buildings, Energy and Buildings, 96 (2015) 109-117.

[22] Y. Chen, G. Zhang, M. Bai, S. Zu, Z. Guan, and M. Zhang, Automatic Waveform Classification and Arrival Picking Based on Convolutional Neural Network, Earth and Space Science, (2019) 1-77.

[23] Z. Xiong, M.K. Stiles, J. Zhao, Robust ECG Signal Classification for Detection of Atrial Fibrillation Using a Novel Neural Network, Computing in Cardiology, 44, 2017.

[24] T. Guo, J. Dong, H. Li and Y. Gao, Simple convolutional neural network on image classification, 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, (2017) 721-724.

[25] J. Li, Y. Si, L. Lang, L. Liu, T. Xu, A Spatial Pyramid Pooling-Based Deep Convolutional Neural Network for the Classification of Electrocardiogram Beats, Appl. Sciences, 8 (2018) 1590.

[26] Z. Hu, Y. Li and Z. Yang, Improving Convolutional Neural Network Using Pseudo Derivative ReLU, 2018 5th International Conference on Systems and Informatics (ICSAI), Nanjing, (2018) 283-287.

[27] M.D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, ECCV, volume 8689 of Lecture Notes in Computer Science, Springer (2014) 818–833.

[28] D. Ciresan, U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber, Flexible, High Performance Convolutional Neural Networks for Image Classification, International Joint Conference on Artificial Intelligence (2011) 1237-1242.

[29] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, ArXiv (2012)

[30] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86 (1998) 2278-2324.

[31] P. Sadowski, Notes on backpropagation, (2016). [Online]. Available: https://www.ics.uci.edu/

[32] C. Lu, Z. Wang, W. Qin, J. Ma, Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification, Signal Process., 130 (2017) 377-388.

[33] L. Gondara, Medical Image Denoising Using Convolutional Denoising Autoencoders, 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, (2016) 241-246.

[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful

representations in a deep network with a local denoising criterion. In Proceedings of the 27th International Conference on Machine Learning, (2010) 3371–3408.

[35] G. Hinton *et al.*, Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, IEEE Signal Process. Mag., 29 (2012) 82-97.

[36] X. Ye, L. Wang, H. Xing and L. Huang, Denoising hybrid noises in image with stacked autoencoder, 2015 IEEE International Conference on Information and Automation, Lijiang, (2015) 2720-2724.

[37] Chollet, François and others, Keras, (2015) [Online]. Available: www.keras.io

[38] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, (2015) [Online]. Available from www.tensorflow.org

[39] M. Carrillo et al, Time series analysis of gravitational wave signals using neural networks, J. Phys.: Conf. Ser. 654 012001, (2015).

[40] M. Takadoya, M. Notake, M. Kitahara, J.D. Achenbach, Q.C. Guo and M.L. Peterson, Crack-Depth Determination By A Neural Network With A Synthetic Training Data Set, Review of Progress in Quantitative Nondestructive Evaluation, 12, (1993) 803-810.

[41] N. J. Rodriguez-Fernandez, P. Richaume, Y. H. Kerr, F. Aires, C. Prigent and J. Wigneron, Global retrieval of soil moisture using neural networks trained with synthetic radiometric data, 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, (2017) 1581-1584.

[42] T. A. Le, A. G. Baydin, R. Zinkov and F. Wood, Using synthetic data to train neural networks is model-based reasoning, 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, (2017) 3514-3521.

[43] A. Witmer and B. Bhanu, HESCNET: A Synthetically Pre-Trained Convolutional Neural Network for Human Embryonic Stem Cell Colony Classification, 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, (2018) 2441-2445.

[44] T.C. O'Haver, Pragmatic Introduction to Signal Processing 2019: Applications in scientific measurement. Kindle Direct Publishing, (2019) p. 340.

[45] N. Bershad and A. Rockmore, On estimating signal-to-noise ratio using the sample correlation coefficient (Corresp.), IEEE Transactions on Information Theory, 20 (1974) 112-113.

[46] K. Ackerson et al., Characterization of "blinking pixels" in CMOS Image Sensors, 2008 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, Cambridge, MA, (2008) 255-258.

[47] SITe SI03xA 24 μm Charge-Coupled Device Family, SITe, Tigard, OR, (2003) [Online]. Available: http://www.not.iac.es/instruments/detectors/CCD1/S103xA_family.pdf