waveGan

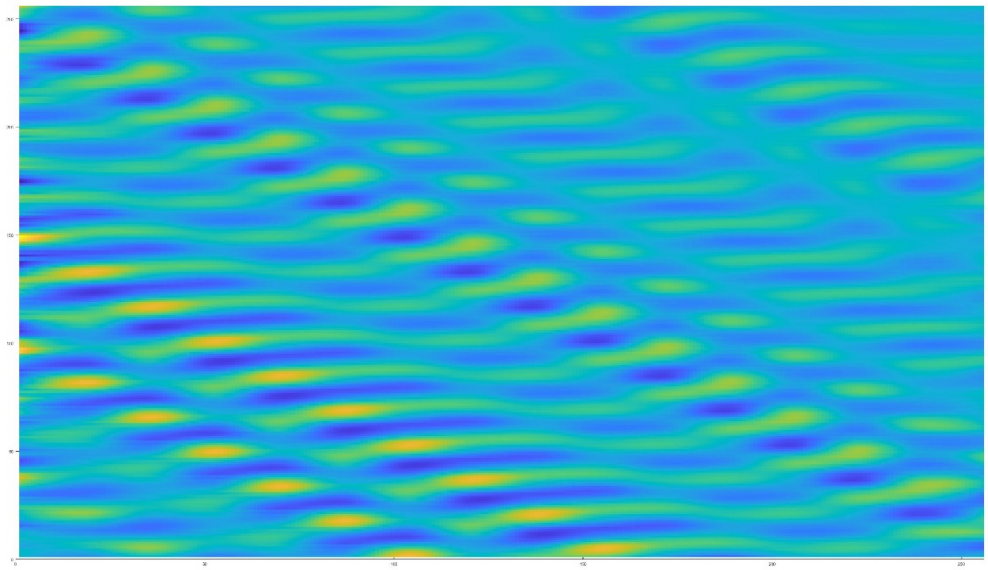
~ wavelet ~ augmented ~ vector ~ extrapolation ~ generative ~ adversarial ~ network ~

Jashua Luna

Thank you to everyone who made this possible. You know who you are.

Table of Contents

List of Figures



Finite States

A butterfly flaps its wings today, tomorrow there is a tornado. It is therefore wise for meteorologists to observe butterflies. But what if there are lots of butterflies? What if there are lots of butterflies in a tornado, during an earthquake? What if the only measuring device is a seismometer? Could a meteorologist with only this seismometer's readings, reliably predict next month's weather? With wavelets and neural networks, yes.

In a closed system, information about future states is contained in past and present state-information. This means that for any finite state model there exists a function that can, with information of the past and present, accurately predict future states. Some systems are more easily modeled than others; in general, the accuracy with which a system's future states can be predicted is inversely proportional to the system's entropy. The price of bitcoin is highly volatile. It frequently experiences vast changes and is considered generally hard to model. BTCUSDT is therefore an excellent example for this investigation. With wavelets and neural networks, a transfer function will be obtained that can with information on past and present states, accurately predict system-states. Before passing through a neural network, BTCUSDT data's entropy is increased by a complex wavelet transform of a sample and by maintaining a windowed record of the sample's transform through time.

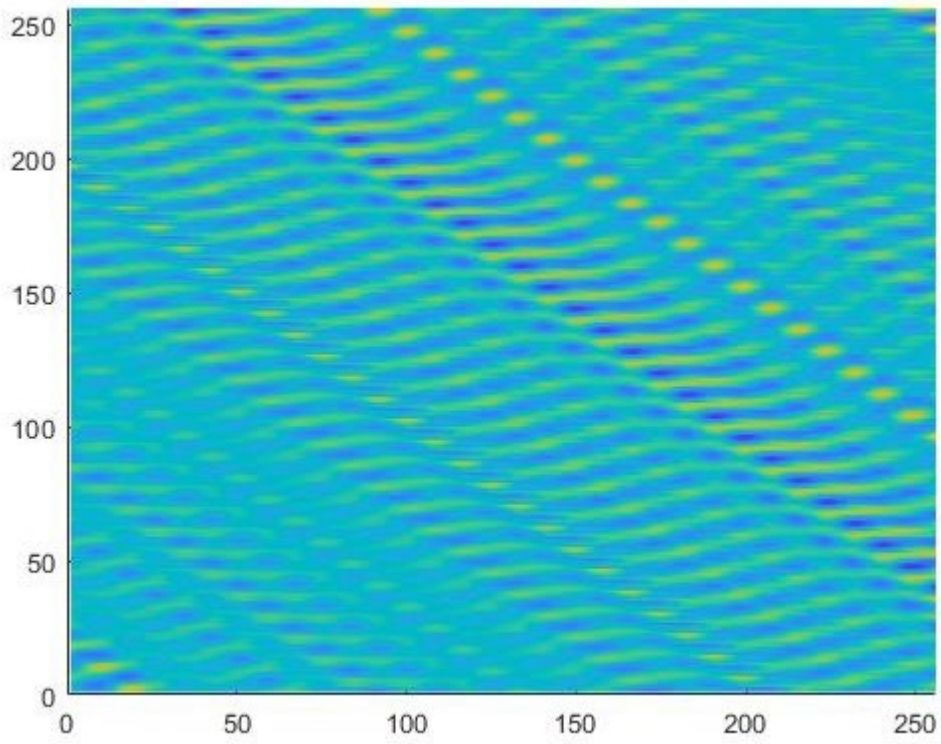
Entropy

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot log\big(P(x_i)\big)$$

$$P_k = \int_{l_k}^{u_k} f(x)\, dx$$

$$H(X) = -\sum_{k=1}^{n} P_k \cdot log(P_k) + logW$$

Shannon entropy H quantifies entropy as a function of how likely system states are to occur. For a discrete variable x with known entropy $P(x)$. For continuous variables with unknown probabilities, entropy can be heuristically



determined with histograms. The probability of a function state is approximated by the probability that the random function has a value within a specified range. For intervals of equal width W between l_k and u_k. One thousand samples of BTCUSDT containing 256 minutes of candle intervals had average entropy 910.45.

Random Walks

The price of Bitcoin at time t is $\mathfrak{V}(t)\ \{\ t, \mathfrak{V}(t)\ \in \mathbb{R}:\ \ t, \mathfrak{V}(t) > 0\ \}$. In this experiment, a function to determine the price of bitcoin at time t + Δt from the price at time t was determined. The function is described as $\mathfrak{V}(t)_{t_1 t_2} = f\big(\Theta, \mathfrak{V}(t)_{t_3 t_4}\big)\ \{t\ \in \mathbb{R}:\ t \geq 0,\ t_2 > t_1, t_1 > t_3 <\ t_4\}$ for some parameters Θ. This function and its parameters were determined analytically by minimizing loss on a hierarchical neural network.

$$\Theta = argmin\ \mathcal{L}[\mathfrak{V}(t)_{t_1 t_2}, f\big(\Theta, \mathfrak{V}(t)_{t_3 t_4}\big)]$$

Since the price of bitcoin has random-walk properties, samples taken from $\mathfrak{V}(t)$ are likely to have different means and variances. Conversely, if samples are taken from , $\mathfrak{V}'(t)$, mean will approximately be zero and samples will have similar variances. For a sample X with n elements:

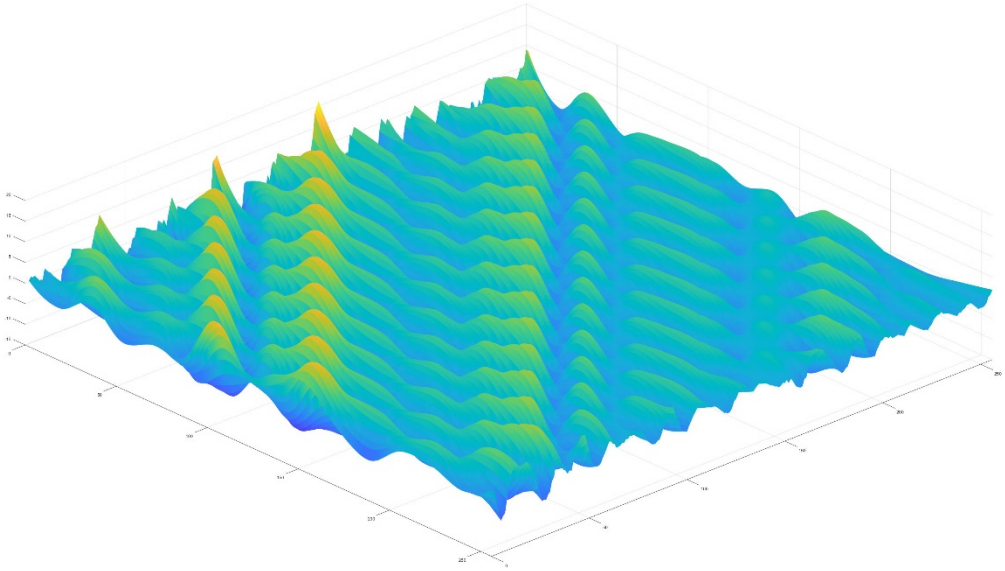
$$\lim_{n\to\infty} \mathbb{E}[\mathfrak{V}'(X)] = 0$$

$$\mathfrak{V}(t_2) =\ \mathfrak{V}(t_1) + \int_{t_1}^{t_2} \mathfrak{V}'(t)dt$$

$$\mathfrak{V}(t)_{t_1 t_2} = f\big(\Theta, \mathfrak{V}(t)_{t_3}, \mathfrak{V}'(t)_{t_3 t_{4,}}\big)$$
$$\mathfrak{V}'(t)_{t_1 t_2} = f\big(\Theta, \mathfrak{V}'(t)_{t_3 t_{4,}}\big)$$

$$\Theta = argmin\ \mathcal{L}[\mathfrak{V}'(t)_{t_1 t_2}, f\big(\Theta, \mathfrak{V}'(t)_{t_3 t_4}\big)]$$

Data samples' dimensionality was increased, by windowing the complex wavelet transform of samples with n elements, between time zero and time n. This creates multiple redundant sources of information for the network model to analyze.



Algorithm 1. Increasing Sample Dimensionality

Iterate *While* $t < t_o + w_s$

1. $S_x \leftarrow \mathcal{D}(t, t + w_s)$
2. $S_y \leftarrow \mathcal{D}(t + l_x, t + l_x + w_s)$
3. $X \leftarrow \begin{bmatrix} X \\ \mathbb{C}_{wt}(S_x) \end{bmatrix}$
4. $Y \leftarrow \begin{bmatrix} Y \\ \mathbb{C}_{wt}(S_y) \end{bmatrix}$
5. $t \leftarrow t + 1$

$\mathcal{D}$	BTCUSDT sampled once per minute
S_x, S_y	Vectorized samples of $\mathcal{D}$ containing w_s elements, from $t_o$to t_n.
$\mathbb{C}_{wt}(x)$	Complex wavelet transform (Dualtree).
t	Sample time, initialized as t_o.
l_x	Extrapolation Length
X, Y	Input and output sample arrays
w_s	Window size

As can be seen in figures 1-3, structural patterns appear when the data is arranged in this manner.

The resultant arrays X and Y are three dimensional. The size of the second and third dimensions of these arrays depends on the window size, choice of wavelet and level of deconstruction. Samples were interpolated such that dimensions were $[w_s, w_s, lvl]$, for lvl levels of wavelet decomposition. The way these samples were obtained means that some intersecting, perpendicular lines, of equal length represent the same information. For example, the information contained in $X(w_s, :, :)$ is the interpolated wavelet transform of the sample between $t_o + w_s$and $t_o + 2 \cdot w_s$. $X(:, w_s, :)$ is a history of the first value of the interpolated wavelet transform between $t_o + w_s$and $t_o + 2 \cdot w_s$. While $X(w_s, :, :) \neq X(:, w_s, :)$, the two encode the same information. This redundancy can be exploited by the network model. Furthermore, X and Y are complex valued. Selecting a complex-valued filter, orthonormal in Hilbert space, results in a decomposition where the complex portion roughly equals the real, with a 90-degree phase shift. Accordingly, the samples are separated into their real and complex parts and arranged real, complex, real… The final samples have dimensions $[w_s, w_s, 2 \cdot lvl]$. One thousand samples of X had an average entropy of 5.63e5.

Networks

The network model and loss functions were pulled and adapted from Isola, Zhu, Zhou and Efros' paper, Image-to-Image Translation with Conditional Adversarial Networks; also known as pixels to pixels. The proposed network is a "U-net" encoder decoder network with skipped connections and the loss functions are a combination of l1 and l2 loss, with the addition of a discriminator network. Because training samples had $2 \cdot lvl$ channels, the pixels-to-pixels model was modified to use grouped convolutions. The exact sizes of the convolution/transposed convolution layers tested were:

Grouped Convolution Sizes:

4	4	2	8	7
4	4	14	16	4
4	4	16	32	4
4	4	32	64	4
4	4	64	128	4
4	4	256	256	2
4	4	512	512	1

Transposed Grouped Convolution Sizes:

4	4	512	512	1
4	4	256	512	2
4	4	128	256	4
4	4	64	192	4
4	4	32	96	4
4	4	16	48	4
4	4	7	60	2

The dimensions specified have format $[F_x, F_y, C_g, F_g, N_g]$ for filters of size $f_x \cdot f_y$, C_g channels per group, F_g filters per group, and N_g groups. In between the grouped convolutions, there were hyperbolic tangent activations and batch-normalizations. The discriminator's architecture was also pulled from pixels to pixels and is PatchGan. This discriminator model evaluates generator output in localized regions. This promotes details and prevents blurring. L1 and L2 losses were combined in the Generator's loss function as recommended in pixels to pixels.

$$\mathcal{L}_D = \lambda_1 \cdot \mathbb{E}\big[log\big(\Phi_D(Y,X)\big) + log\big(1 - \Phi_D(\Phi_G(X),X)\big)\big]$$

$$\mathcal{L}_G = \lambda_1 \cdot \mathbb{E}\big[log\big(\Phi_D(Y,X)\big)\big] + \lambda_2 \cdot \mathbb{E}[|Y - \Phi_G(X)|]$$

$$\{\lambda_1, \lambda_2 \in \mathbb{R} : \lambda_1, \lambda_2 \neq 0\}$$

(Chun)

Because "U-net" is symmetrical, the outputs are the same size as inputs; to recover the extrapolated signal from this network, dimensionality must be reduced. The easiest way to do this would be to, by interpolation, resize and rearrange the data in $\hat{X}(w_s, :, :)$ so that it matches the dimensions of the specified complex wavelet transform for a sample sized w_s , then to take the inverse complex wavelet transform of that. This method, however, discards most of the extrapolated data and therefore does not make effective use of the information available. A second network could be trained to reduce the dimensionality of the first network's output in a way that maximizes the accuracy of the inverse complex wavelet transform of the second network's output. Ideally:

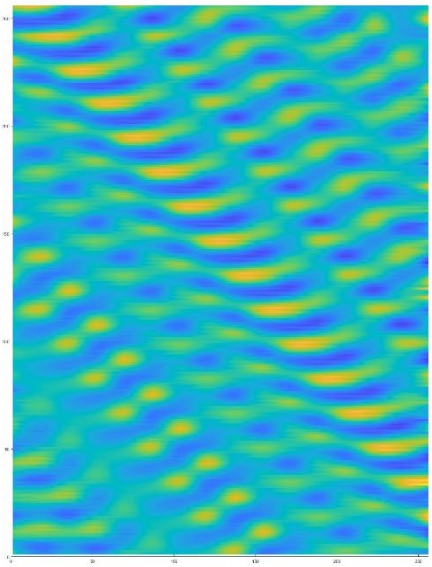
$$\hat{X} = i\mathbb{C}_{wt}\left(\Phi_2\big(\Phi_1(X)\big)\right), \hat{X} \cong S_y$$

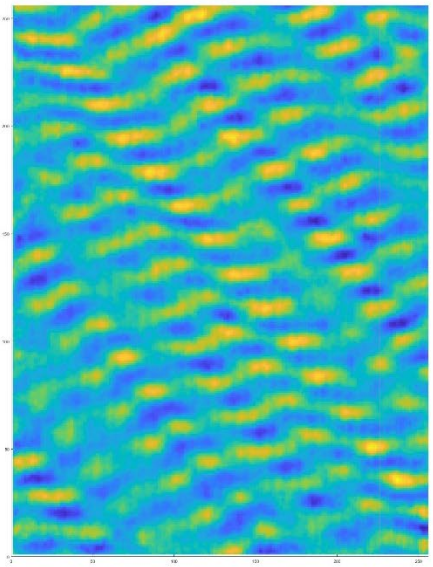
This network could be trained to minimize error between the expected signal and the inverse complex wavelet transform of its output.

$$\Theta = argmin\, \mathcal{L}[S_y, X]$$

$$\mathcal{L} = \mathbb{E}\left[\lambda \odot \left|S_y - i\mathbb{C}_{wt}\left(\Phi_2\big(\Phi_1(X)\big)\right)\right|\right]$$

Since accuracy in extrapolated terms might be more important than elsewhere, bias can be introduced with vectorized coefficients λ .





$$RMSE = \frac{1}{N}\sum_{n=1}^{N}\sqrt{\frac{1}{w_s \cdot w_s \cdot 2 \cdot lvl}\sum_{x=1}^{w_s}\sum_{y=1}^{w_s}\sum_{z=1}^{2\cdot lvl}\left(Y_{x,y,z} - \hat{X}_{x,y,z}\right)^2}$$

Results

Several *pixel-to-pixel* networks (with grouped convolutions) were trained on varying sets of 1000 samples. They were trained with adaptive moment estimation using varying learning rates and batch size 32. With a window length of 256 minutes, three extrapolation lengths were considered: 30, 60 and 128 minutes. On average, trained networks had root mean squared error (rmse) of 0.21, 0.25 and 0.32 for extrapolation lengths 30, 60 and 128, respectively. Error was determined by the element wise subtraction of the network's output from the corresponding reference output. For N samples, RMSE was calculated according to the equation listed above. The second network, whose inverse complex wavelet transformed output would approximately equal the reference signal, was not trained as it was sufficient for the purposes of this investigation to only evaluate the first's performance. Results from training were not as expected, RMSE values were higher than anticipated and certainly larger than would be acceptable if reliable price-predictions were to be made. For more accurate results, a different network architecture could be implemented, or settings changed. Also, at this point it is unknown if the addition of the second network would improve prediction accuracy; furthermore, the complex wavelet transform's scaling coefficients were discarded before training and the samples max-min scaled such that $\{X, Y \in \mathbb{R} : -1 \leq X, Y \leq 1\}$. From the trained networks (as illustrated above), it can be concluded that to an extent, highly entropic systems, like the price of bitcoin can be accurately modeled by some unknown function, in this case, an analytically determined combination of wavelets and neural networks. If this were to be extended for practical use, scaling coefficients could be also predicted by extending the existing networks, or adding another in parallel; it should be noted that the Dualtree wavelet transform's scaling coefficients did not vary much for the time window used (256 min). These could probably be extrapolated accurately with a linear regression.

Conjecture

It was demonstrated that a transfer function that can predict future states from current and past states can be analytically determined. However, the dataset that was used in training has some unique properties which might make similar results difficult to obtain in other scenarios. The most important being that the price of bitcoin is measured independently of the data collection for this experiment. This means that it is negligibly changed, if at all, by the collection of these samples for network training; The price of bitcoin is unaffected by the training of this network. If the information produced by this trained network were used to make changes in the system it seeks to predict, the accuracy of its predictions would likely decrease quickly. This is because the system the function models is uninfluenced by the function; addition of information produced by the function would alter the system, possibly in a radical way which would produce changes unknown to the function model thus decreasing prediction accuracy. Consider the ideal transfer function which can predict a system's future states with 100% accuracy.

$$X_1 = \Phi_{\natural}(X_o)$$

$$X_{k+1} = \Phi_{\natural}(X_k)$$

$$X_{k+1} = X_k + \Phi'_{\natural}(X_k)$$

$\natural$ denotes natural, uninfluenced by the observer. This function can reliably predict a system's future states from its current and past states. The instant that information from this function were used to change the system, the system's state would be modelled by the combination of the original predicted state, and the product of the predicted state and some unknown influence factor H.

$$\Phi_N(X_k) = H \cdot \Phi_{\natural}(X_k) + \Phi_{\natural}(X_k)$$

This action would change the system such that $\Phi_{\natural}$ would no longer be an accurate reflection of the system's state. A new function which includes observer influence, denoted by N, would become the ideal model. As the observer makes changes in the system, there would continue to be influence from the original function. System states at this stage are modelled as a combination of the new function and the natural.

$$\Phi_N(X_k) = H \cdot \Phi_N(X_k) + \Phi_{\natural}(X_k)$$

As the observer continues to make changes in the system, natural processes will become user influenced processes. If enough actions are made by the observer, the system change unaccounted for by the N function would approach zero. Eventually, future system states can be modelled as a function of influenced states, influence factor and a random term represented by gamma.

$$X_{k+1} = H \cdot f'\big(f'(X_{k-1})\big) + \Upsilon_k$$

This resembles the general form of a Kalman filter, and it might be useful to explore these similarities. A network might train on data it creates, progressively becoming more accurate, or it might be that this random term is insurmountable, and there is no way to simultaneously make changes to a chaotic system and maintain prediction accuracy. Some other time perhaps.

Files

For those interested, project files can be found at [https://github.com/not-JASH/waveGan.git]

Bibliography

Akram, Muhammed Adeel, et al. “A State Optimization Model Based on Kalman Filtering and Robust Estimation Theory for Fusion of Multi-Source Information in Highly Non-Linear Systems.” *Sensors*, vol. 19, 1687, 9 Apr. 2019.

Beale, Mark Hudson, et al. *Deep Learning Toolbox Reference*. Mathworks, 2020.

Isola, Phillip, et al. *Cornell University Online Archive*, Computer Science, 21 Nov. 2016, arXiv:1611.07004v3 [cs.CV] 26 Nov 2018.

Kingma, Diederik P, and Jimmy Lei Ba. “Adam: A Method For Stochastic Optimization.” *Cornell University Online Archive*, Machine Learning, 22 Dec. 2014, arXiv:1412.6980v9 [cs.LG] 30 Jan 2017.

Leung, Yui Chun. “Matlab-Gan.” *GitHub*, Repository, 3 Apr. 2020, github.com/zcemycl/Matlab-GAN.git.

McHardy, Sam. “Python-Binance.” *GitHub*, Repository, 5 Jan. 2021, github.com/sammchardy/python-binance.git.

Selesnick, Ivan W, et al. “The Dual-Tree Complex Wavelet Transform.” *IEEE Signal Processing Magazine*, Nov. 2005, pp. 123–15.

Wallis, Kenneth. “A Note on the Calculation of Entropy from Histograms.” *Munich Personal RePEc Archive*, 10 Jan. 2014, mpra.ub.uni-muenchen.de/52856/.
