It Seems Random, But It's Chaotic: N-CATS, a Model for Cryptocurrency Price Prediction

JAYJAY JEONGJIN PARK

ACM Reference Format:
Jayjay Jeongjin Park. 2023. It Seems Random, But It's Chaotic: N-CATS, a Model for Cryptocurrency Price Prediction. 1, 1 (December 2023), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

Cryptocurrency price prediction is a very challenging problem in asset-pricing, and numerous deep learning methods were devised to predict the price. However, not only existing methods do not perform well in long term prediction, their methods do not take advantage of the market dynamic's information. In this project, a new model is created to better predict long term price behavior, utilizing prior knowledge on market dynamics. Out of all the models tested, the model that learned the market dynamics the best is the new proposed model, N-CATS. Code is available in https://github.com/jayjay-park/N-CATS.git

2 INTRODUCTION

Due to high volatility, predicting cryptocurrency price have been a difficult problem to solve. Especially, study conducted by Yang et al. point out how cryptocurrency price prediction is even more difficult problem to solve than stock price prediction through analysis with entropy[1]. A study conducted by Yi et al. shows that the cryptocurrency market is weakly market efficient, which just means that predicting crypto-asset price with just past trends of price is impossible[2].

However, some recent papers claimed the cryptocurrency market shows a chaotic behavior, which implies market is a type of a chaotic dynamical system. This introduces exciting new possibilities for how we can approach the challenge of modeling the behavior of the cryptocurrency market. That is, from just a simple regression problem, we can now use properties of a chaotic dynamical system as inductive bias in model development.

Some previous approaches in predicting crypto-asset price using Neural Networks include Long Short-Term Memory(LSTM)[3]. However, as it shown in the Result section and previous study[1], LSTM does not perform well nor is it robust enough for cryptocurrency price prediction.

Thus, in this project, a new method, N-CATS, attempts to learn both market's chaotic dynamic along with past trends of crypto-asset price, with the new auto-correlation loss function. Specifically, this study focuses on answering following research questions: 1. Can we verify if cryptocurrency market is indeed chaotic? 2. How does baseline models perform in predicting price? 3. How can we make a model learn market dynamics information? 4. How does the new model perform?

3 BACKGROUND INFORMATION

3.1 Measurements of a Chaotic System

3.1.1 Lyapunov Exponent. A chaotic system is a deterministic dynamical system that is extremely sensitive to initial point[4]. The chaotic system's sensitivity to initial point can be well explained with lyapunov exponent as lyapunov exponent measures the sensitivity to initial point.

When $Z(t)$ and $Z_0(t)$ are infinitesimally close trajectories, let $\delta Z(t) = Z(t) - Z_0(t)$ and $\delta Z_0 = Z(0) - Z_0(0)$, then λ is the Lyapunov exponent in the following equation.

$$|\delta Z(t)| \approx e^{\lambda t}|\delta Z_0| \quad s.t.\ t \in \mathbb{R}, \lambda \in \mathbb{R}$$

When $\lambda > 0$, system is defined as chaotic system as it means that the difference in two close trajectories exponentially diverge, thus becoming chaotic system.

In this project, lyapunov exponent is used to validate if the model learned the chaotic system or not, by seeing if it is positive number. Then, we can also compare how close the reproduced lyapunov exponent is compared to lyapunov exponent from the original time series data.

3.1.2 Auto-Correlation. Auto-correlation is the correlation between a time-series and delayed version of itself.

Author's address: Jayjay Jeongjin Park.

XXXX-XXXX/2023/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

When time-lag is define as τ, window-size is T, and x is a state variable, auto-correlation $C(x, \tau)$ is defined as the following.

$$C(x,\tau) = \mathbb{E}(x_t x_{t+\tau}) - \mathbb{E}(x_t)\mathbb{E}(x_{t+\tau}) = \frac{1}{T}\sum_{t\leq T} x_t x_{t+\tau} - \frac{1}{T}\sum_{t\leq T} x_t \frac{1}{T}\sum_{t\leq T} x_{t+\tau}$$

If a system's dynamic is chaotic, then auto-correlation should converge to 0 as value of τ increases[5].

3.1.3 Multi-Step Prediction. Although multi-step prediction is a more general term used in time-series modelling, if a model learned a correct type of chaotic system, then model necessarily will have reasonably low multi-step prediction error. Unlike one-step prediction, error accumulates, thus it gives more insight in how well model performs comapred to one-step prediction. With the learned model $\hat{M}$, then for $t \in [0, ..., N]$, we follow the following steps to compute multi-step prediction.

$Define\ T = [0, ..., N]$
for `t in T` **do**
  $\hat{x}_{t+1} = \hat{M}(\hat{x}_t)$
end for

In summary, in this project, statistical measures of chaotic system are used in two different way: 1. to assist with training by using auto-correlation, 2. verify if a model learned a true dyanmics or not by using lyapunov exponent, multi-step prediction.

3.2 Framework for Learning Time Series

3.2.1 Long-Short Term Memory. Long-Short Term Memory proposed by Hochreiter et al. attempts to solve long term dependency and vanishing gradient problem of Recurrent Neural Network[6]. It is consisted of three gates: input gate, i_t, forget gate, f_t, and output gate, o_t.

$$\begin{aligned} i_t &= \sigma(w_i[h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(w_f[h_{t-1}, x_t] + b_f) \\ o_t &= \sigma(w_o[h_{t-1}, x_t] + b_o) \\ s.t. \quad & \sigma = sigmoid() \\ & w = weight\ matrix \\ & h_{t-1} = output\ from\ the\ previous\ time \\ & x_t = current\ input \\ & b = bias \end{aligned}$$

To generate memory vector,

$$\begin{aligned} \hat{c}_t &= tanh(w_c[h_{t-1}, x_t] + b_c) \\ c_t &= f_t * c_{t-1} + i_t * \hat{c}_t \\ h_t &= o_t * tanh(c_t) \end{aligned}$$

$\hat{c}_t$ is a candidate for memory, c_t is memory at time t. When h_t is passed through softmax layer, we get the predicted output, y_t.

3.2.2 Neural Ordinary Differential Equation. Neural Ordinary Differential Equation is a framework proposed by Chen et al. for learning dynamical system[7]. When x is an initial state, h is a neural network that models the dynamics, and ϕ_h are the time integrator of h, Neural ODE is defined as such:

$$\frac{d}{dt}\phi_h^t(x) = h(\phi_h^t(x)) \quad t \in \mathbb{R}^+, x \in \mathbb{R}^d, \phi_h^t(x) \in \mathbb{R}^d$$

One of the few limitations of Neural ODE is that it is for learning deterministic system, which makes it difficult to learn stochastic or highly volatile system[8].

3.2.3 Neural Stochastic Differential Equation. To overcome such limitation of Neural ODE, Kidger et al. proposed a method which fits Neural Stochastic Differential Equation(Neural SDE) with GANs[9]. When $t \in [0, T]$, x is a state variable, and y_t is a solution, Neural SDE is defined as following.

$$\begin{aligned} dx_t &= \mu_\theta(t, x_t)dt + \sigma_\theta(t, x_t) \circ dW_t \\ y_t &= \alpha_\theta x_t + \beta_\theta \end{aligned}$$

In here, μ_θ is a neural network that approximates drift, and σ_θ is a nueral network that approximates diffusion.

In the current project, instead of using GAN to make Neural SDE fit, a new loss function is used to train latent Neural SDE.

4 EXPERIMENT

4.1 Dataset

The dataset used for the experiment was obtained from Kaggle Bitcoin Historical Dataset[1].

The Dataset includes every 1 minute historical price in 2021, including features like Unix timestamp, dates, symbol, opening, high, low, closing, Volume and Volume Base.

The task of this project is a univariate time series prediction; therefore, only closing price was used among

[1]https://www.kaggle.com/datasets/prasoonkottarathil/btcinusd/?select=BTC-2021min.csv

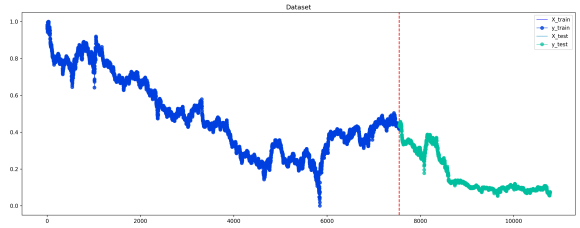

Fig. 1. Plot of Train, Test Dataset with Training-Test Data Split Shown

	Learning Rate	Epochs	Architecture	Feature
LSTM	$1e-3$	1000	1 LSTM layer, 2 Linear layer	window size of 10
Neural ODE	$5e-4$	1000	6 Linear Layer	
N-CATS	$5e-4$	800	2 Linear layer	latent dim of 64

Table 1. Summary of Hyperparameter Setting

multiple features. For the data pre-processing, min-max normalization is used. Then, data is splitted by 70:30 ratio, thus having training data size of 7546 and test data size of 3234.

4.2 Experiment Setting

For the baseline models, two types of model, Long-Short Term Prediction(LSTM) and Neural Ordinary Differential Equation(Neural ODE), that are frequently used for time series prediction are selected[3].

Following is the summary of hyper-parameter setting for the experiment. Training algorithm used is AdamW, with different learning rates per model. Different numbers of epoch are used as well. Rest of the details are included in the following table.

5 METHOD

5.1 Learning Problem

In this supervised learning problem, we define empirical risk minimization problem as the following: *Given* $S = \{z_i\}_{i=1}^m = \{(x_i, y_i)\}_{i=1}^m, x \in \mathbb{R}^d$,

$$\mathbf{R}(h) = \mathop{\mathbb{E}}_{S\sim D^m} \hat{\mathbf{R}}_s(h) = \mathop{\mathbb{E}}_{S\sim D^m} \frac{1}{m}\sum_{i=1}^{m} l(z_i, h)$$

$$MSE\_loss = l(x, h) = \|y_i - model(x_i)\|^2$$

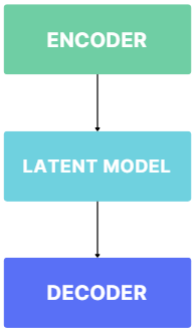

Fig. 2. Architecure of N-CATS

5.2 N-CATS: Neural Chaotic Auto-correlation for Time Series

5.2.1 Architecture. In an attempt to make a model learn market dynamics, N-CATS is proposed.

Like shown in the figure, N-CATS is consisted of three parts. Encoder that transform state variable to latent dimension, latent model that learns the dynamic in higher dimension, and lastly, decoder that returns prediction in original d dimension.

Latent model can be anything like LSTM, Neural ODE. But for this project, Neural SDE is chosen as latent model.

5.2.2 Learning Problem. The new model is then trained with new loss function. This new loss function is Mean Squared Error loss with norm difference of auto-correlation as regularization term. λ is a regularization parameter, and T is windown-size. Time-lag is defined as τ.

$$\mathcal{L}_{new\_loss} = \mathcal{L}_{MSE} + \lambda * \mathcal{L}_{autocorrelation}$$
$$s.t.\ \lambda \in [0, 1]$$
$$\mathcal{L}_{auto-correlation} = \|\mathbb{E}(x_t x_{t+\tau}) - \mathbb{E}(x_t)\mathbb{E}(x_{t+\tau})\|$$
$$= \|\frac{1}{T}\sum_{t\leq T} x_t x_{t+\tau} - \frac{1}{T}\sum_{t\leq T} x_t \frac{1}{T}\sum_{t\leq T} x_{t+\tau}\|$$

6 RESULTS

6.1 RQ1: Can we verify if crypto-currency market is indeed chaotic?

First, to approximate lyapunov exponent from time series, function[2] that implements algorithm of Rosenstein et al. is used[10].

[2]https://pypi.org/project/nolds/

Algorithm 1 Training the N-CATS

Input: $\{(x_t)\}_{i=0}^{M}$
$\lambda \in [0, 1]$
$\tau = [0, ..., V]$
for i in 1,2, ..., N **do**
e_t = Encoder(x_t)
l_t = Latent Model(e_t)
o_t = Decoder(l_t)
MSE_loss = $\|o_t - x_{t+1}\|^2$
AC_loss =
$\| \frac{1}{T} \sum_{t \leq T} x_{t+1} x_{t+1+\tau}$
$- \frac{1}{T} \sum_{t \leq T} x_{t+1} \frac{1}{T} \sum_{t \leq T} x_{t+1+\tau} \|$
- $\| \frac{1}{T} \sum_{t \leq T} o_t o_{t+\tau} - \frac{1}{T} \sum_{t \leq T} o_t \frac{1}{T} \sum_{t \leq T} o_{t+\tau} \|$
total_loss = MSE_loss + λ*AC_loss
total_loss.backward()
optimizer.step()
end for
Return: $\{o_i\}_{i=1}^{N}$

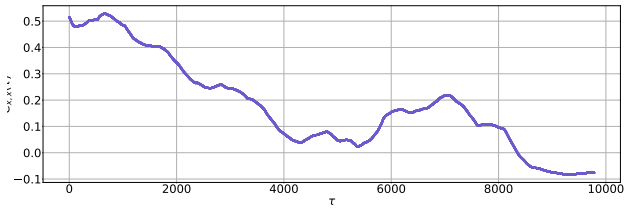

Fig. 3. Auto-correlation of Entire Dataset

The computed lyapunov exponent of the whole dataset is 0.001927. However, it is notable that as we don't exactly know the true dimension of the cryptocurrency market dynamics. Thus, when computing lyapunov exponent using Eckmann et al.'s method, with arbitrary dimension of 2, the lyapunov spectrum returned is [0.2607571, -0.1330105]. From this result, we can verify that cryptocurrency market is indeed chaotic system.

When plotting the auto-correlation of dataset, we can also observe that it is converging to 0, which is expected behavior for chaotic system.

6.2 RQ2,3: How does models perform in predicting price?

Regarding one-step prediction, LSTM did not perform that well as we can observe from the table and the one step prediction plot for the whole dataset. This is due to LSTM's vanishig gradient problem[11].

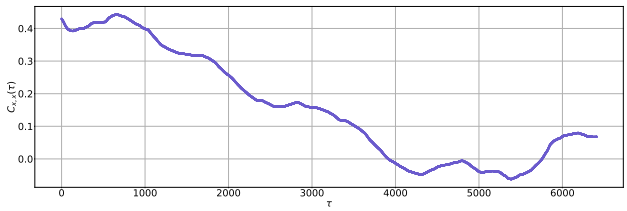

Fig. 4. Auto-correlation of Test Dataset

	Train Loss (One-Step) in MSE or AC Loss	Test Loss (One-Step) in MSE
LSTM	0.04117	0.11384
Neural ODE	$3.2348e-05$	**1.0721e − 05**
N-CATS	0.0022	0.00013

Table 2. One Step Prediction Loss Loss Table

	Multi-Step Prediction Loss	Norm Diff LE
Neural ODE	16.9741	0.0001
N-CATS	**6.5225**	**2.8197e − 05**

Table 3. Multi-step Prediction Error & Norm Difference of Lyapunov Exponent Table

Interestingly, for Neural ODE, test loss was low enough and the one step prediction plot looked good as well. However, when looking at the multi-step prediction, it is apparent that what Neural ODE learned is not the true chaotic system, which will be discussed in-depth in the next section. One of the possible reason why it is performing poorly on multi-step prediction is because Neural ODE is for learning deterministic system.

Overall, when looking at the plot, both Neural ODE and N-CATS performed well in one-step prediction.

Finally, for N-CATS, for training loss, new loss function was used, so it would not be fair to compare the training loss. However, for test loss, Means-Squared Loss, is not lower than that of Neural ODE. If N-CATS were to run on even longer epoch, the same 1000 epoch, then it is expected to decrease even lower and show similar performance as Neural ODE.

6.3 RQ4: Are models learning dynamics?

To assess if the model is learning the correct dynamics, following errors are computed.

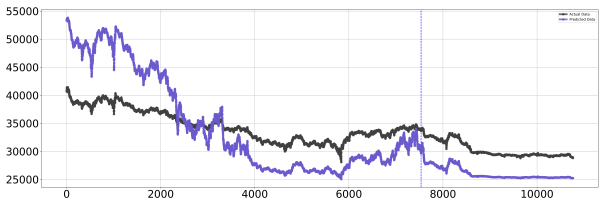

Fig. 5. One-Step Prediction Plot for LSTM from 0 to 10783 days

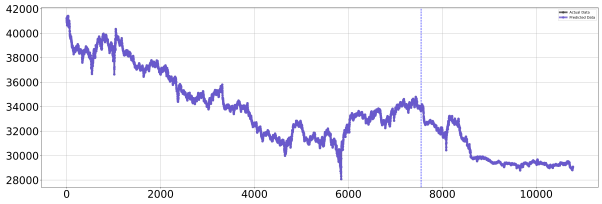

Fig. 6. One-Step Prediction Plot for Neural ODE from 0 to 10783 days

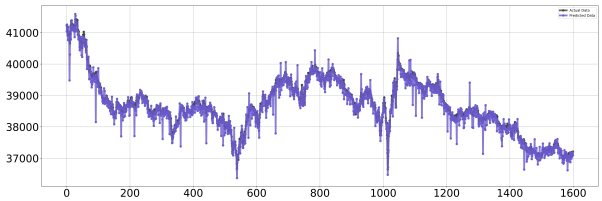

Fig. 7. One-Step Prediction Plot for N-CATS from 0 to 1600 days

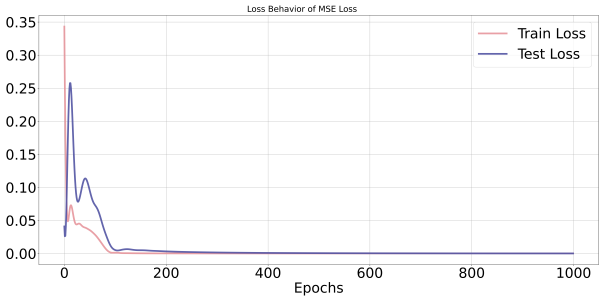

Fig. 8. Loss Plot for LSTM

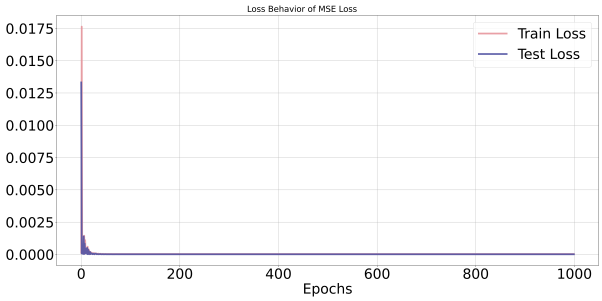

Fig. 9. Loss Plot for Neural ODE

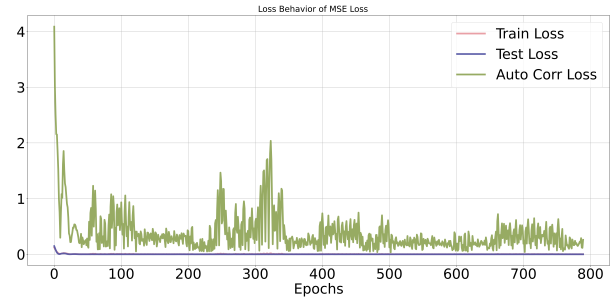

Fig. 10. Loss Plot for N-CATS

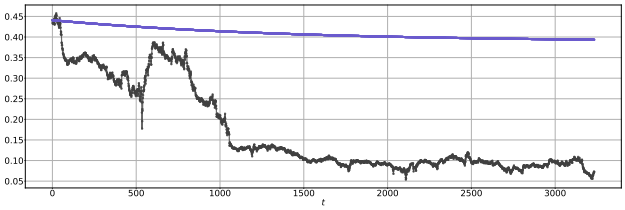

Fig. 11. Multi-Step Prediction for Neural ODE on Unseen Data

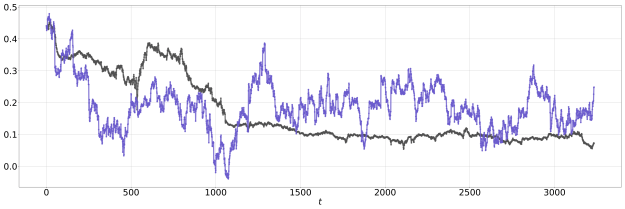

Fig. 12. Multi-Step Prediction for N-CATS on Unseen Data

$$Norm\_Diff\_LE = \|LE\_true\_data - LE\_predicted\_data\|$$

$$Multi\_Step\_Err = \|X - \tilde{X}\|$$

$$s.t.\ \tilde{X} = multi\_step\ predicted\ time\ series$$

When testing with lyapunov exponent and multi-step prediction error to assess if a model is learning the correct system, among the two models, N-CATS showed the smallest multi-step prediction error, 6.5225, and the smallest norm difference of lyapunov exponent, $2.8197e-05$.

As shown in Fig 11 and 12, Neural ODE is not really learning dynamic when N-CATS is at least behaving similar although it is not exactly the same. However, after conducting grid search and training in longer epoch, it is expected to learn a lot better in multi-step prediction.

7 FUTURE WORK

Some of the future work that can be done include 1. conduct grid search to find out optimal hyper-parameter setting for N-CATS, 2. Try out different kind of latent model, 3. Try to make model size smaller so that it is able to conduct experiment under the same condition. (for e.g., same 1000 epoch for training)

8 CONCLUSION

In this project, a new model that employs information on market dynamics for predicting cryptocurrency price is proposed. Model validation on whether or not it learned the dynamic is conducted using Lyapunov exponent, and multi-step prediction.

REFERENCES

[1] Liuqing Yang, Xiao-Yang Liu, Xinyi Li, and Yinchuan Li. Price prediction of cryptocurrency: An empirical study. In Meikang Qiu, editor, *Smart Blockchain*, pages 130–139, Cham, 2019. Springer International Publishing.

[2] Eojin Yi, Biao Yang, Minhyuk Jeong, Sungbin Sohn, and Kwangwon Ahn. Market efficiency of cryptocurrency: evidence from the bitcoin market. *Scientific Reports*, 13(1):4789, 2023.

[3] Sina E Charandabi and Kamyar Kamyar. Prediction of cryptocurrency price index using artificial neural networks: a survey of the literature. *European Journal of Business and Management Research*, 6(6):17–20, 2021.

[4] S.E. Jørgensen. Chaos. In Sven Erik Jørgensen and Brian D. Fath, editors, *Encyclopedia of Ecology*, pages 550–551. Academic Press, Oxford, 2008.

[5] VS Anishchenko, TE Vadivasova, GA Okrokvertskhov, and GI2014154 Strelkova. Correlation analysis of dynamical chaos. *Physica A: Statistical Mechanics and its Applications*, 325(1-2):199–212, 2003.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[8] Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

[9] Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural sdes as infinite-dimensional gans. In *International conference on machine learning*, pages 5453–5463. PMLR, 2021.

[10] Michael T Rosenstein, James J Collins, and Carlo J De Luca. A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 65(1-2):117–134, 1993.

[11] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn, 2018.