LSTM detection of Crypto-Pump and Dumps

Levi Butcher
lcb0012@mix.wvu.edu
CPE520 Final

I. Introduction

The popularity of Cryptocurrencies has been on the rise recently no doubt because of social media figure heads advertising their own investment into the coins like Elon Musk or Mark Cuban. However, the everyday person who doesn't completely understand the technology may not realize how volatile some of the crypto-coins are, resulting in many people being taken advantage of in a Pump-And-Dump scheme. A Pump-And-Dump scheme is where a individual or group will falsely get people to invest in a stock saying it's a wise investment to then turn around and sell of all of their own shares plummeting the price of the stock. Cryptocurrencies are perfect targets for these pump-and-dump schemes because they are not regulated by any government or organizational body making it not illegal. I seek to develop a method to identify the start of a pump-and-dump in order to warn people if it's a unwise investment as early as possible.

A. Goals

This project goals our two fold which are:

1) Identify a pump-and-dump using a time series of market data
2) Identify the pump-and-dump as early as possible. (Using only smallest amount of time series data)

B. Previous Works

This work will be based off of the work done by the authors of [1]. In their own paper they analyzed different pump and dump groups and learned how they operated. They collected data on when these pump-and-dumps took place and published this data for the good of the research community. This work uses their published dataset and uses their results as a baseline to analyze this work. In their paper they got the results shown in figure 1 which gives the best results using a random forest model with a time window of 5 seconds scoring a 84% f1 score which is very good.

II. Methodology

A. Model

My work is based on using a LSTM network to determine if a time frame of market data is the start of a pump-and-dump or not. A LSTM (Long Short Term Memory) is a Neural Network model based on RNN (Recurrent Neural Networks). LSTM's used the output of the previous cell as input for the next cell in time along with the normal sample input. Also LSTMs make use of cell state to determine what information is worthwhile to remember later on in time. This cell state solves a common problem found in RNNs which is vanishing gradient or exploding gradient which both make training a Neural Network basically impossible due to too small or large gradients, respectively. Figure 2 shows the flow of a LSTM cell.

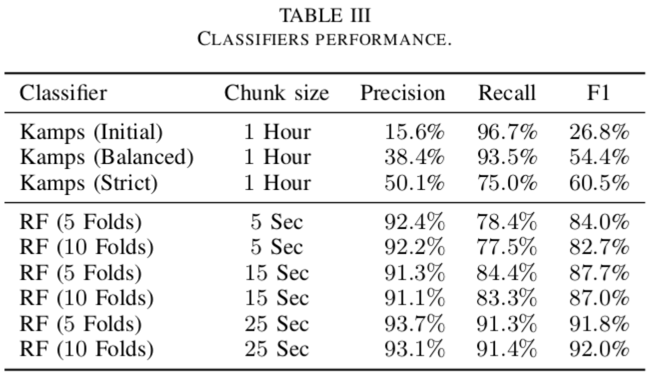

TABLE III
Classifiers performance.

Classifier	Chunk size	Precision	Recall	F1
Kamps (Initial)	1 Hour	15.6%	96.7%	26.8%
Kamps (Balanced)	1 Hour	38.4%	93.5%	54.4%
Kamps (Strict)	1 Hour	50.1%	75.0%	60.5%
RF (5 Folds)	5 Sec	92.4%	78.4%	84.0%
RF (10 Folds)	5 Sec	92.2%	77.5%	82.7%
RF (5 Folds)	15 Sec	91.3%	84.4%	87.7%
RF (10 Folds)	15 Sec	91.1%	83.3%	87.0%
RF (5 Folds)	25 Sec	93.7%	91.3%	91.8%
RF (10 Folds)	25 Sec	93.1%	91.4%	92.0%

Fig. 1. [1] Paper Results

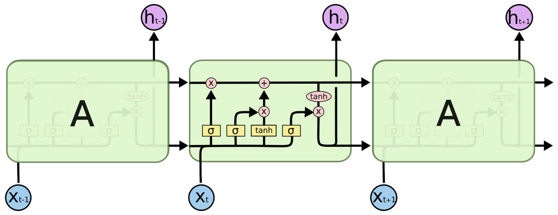

Fig. 2. LSTM Cell Flow from [2]

My work uses the LSTM model in two different ways. Method 1 shown in 3 is using a LSTM followed by a sigmoid activation function as a prediction of whether or not a time slice is a pump and dump. Binary Cross-Entropy is used as the loss function in that method.

Method2 using a LSTM Autoencoder to determine if data within a time frame is a pump-and-dump which is shown in figure 4. This model uses a LSTM to perform dimensionality reduction on a sample then decodes that data back to it's original form. The premise being that anomalies in the data won't be able to be recreated as well as normal data is meaning anomalies have higher loss. With this method a threshold has to be used to determine whether or not a sample is a pump-and-dump based on if it's recreations loss is greater then the threshold.

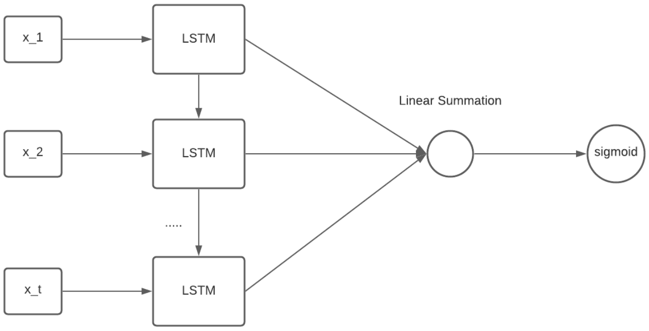

Fig. 3. Model 1 Design

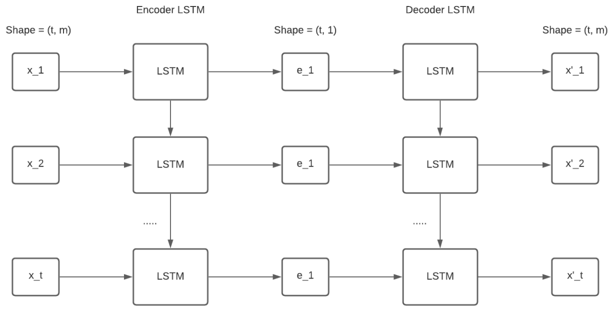

Fig. 4. Model 2 Design

B. *Preprocessing*

The dataset this work uses comes as market data from the Binance Cryptocurrencies platform. From the start of a pump-and-dump data is pulled 7 days prior and 7 days after and stored as a CSV file. There are many transactions that happen at the same time which need to be aggregated and stored as one record of buying or selling. Then the data is sliced into size K chunks, k represents the amount of seconds we are analyzing in predicting a pump and dump. If a given chunk is within k distance from the start of the pump-and-dump then it's labeled as a pump-and-dump the rest are labeled as normal. This form of labelling provides very few pump-and-dump labels for training and will be discussed in the Results section of this work. Before training the chunks are normalized to values between 0 and 1.

C. *Model Configurations*

Both models were implemented in python3 using the Pytorch library. Both had a learning rate of 0.01 and a momentum set to 0.9. Each model was run using the following time slice sizes in seconds: 5,15,25,60. Each model was ran for 5 epochs using batch gradient descent with a batch size of 100. For Model 2 a validation set is used to explore what the best theshold for scoring the highest f1 score is. The following start threshold values are used: [0.01, 0.001, 0.0001] and each one is multiplied by the square root of values going from 0 to 100.

All code is available publicly on this Github Repository.

III. Results

The results for method 1 are shown in figure I and the results for method 2 are shown in [**LSTMAutoresults**]. Method 1 performs poorly in all different values of k because there is too little examples of labeled pump-and-dump examples to perform training, the resulting model ends up always predicting 0 no matter the the data shown to it. Method 2 shows a little more promising in being able to predict a pump-and-dump however for the most part it always predicts normal. Method2 requires a search of best threshold for scoring the best recall score which is done using a standard iterative search approach. The threshold chosen is very finicky though because of the small amount of loss in the Autoencoder and a library using a grid search could give better results for the threshold.

TABLE I
LSTM Results

Chunk Size	True %	Accuracy	Precision	Recall	F1
5	0.066	99.93	0	0	0
15	0.268	99.73	0	0	0
25	0.529	99.47	0	0	0
60	1.459	98.54	0	0	0

TABLE II
LSTM AutoEncoder Results

Threshold	Chunk Size	True %	Accuracy	Precision	Recall	F1
0.0002	5	0.06	99.74	0.0076	0.0215	0.0113
0.0002	15	0.26	99.30	0.0256	0.0433	0.0322
0.0002	25	0.52	99.24	0.0327	0.0151	0.0207
0.00059	60	1.43	95.67	0.0430	0.0953	0.0593

IV. Conclusion

My methods did not perform well, especially in comparison to the work performed in [1], however, the LSTM Autoencoder does show the potential of getting a working model. With more time spent in fine tuning the model such as finding a better threshold, exploring a MLP (Multi-layer perceptron) between the encoder/decoder, or increasing the size of the encoder output could lead to better results. Ultimately, this work shows the potential of time series based models in detecting pump-and-dumps and with more time spent on this subject could show remarkable results for this problem.

V. Other Figures

A. *Training Loss*

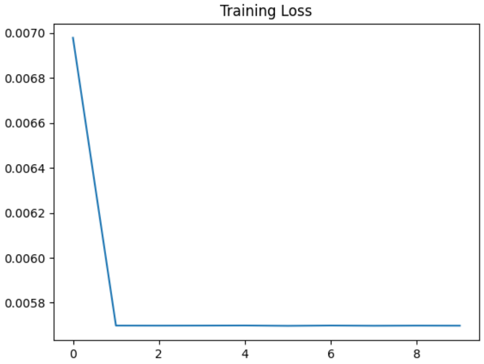

Fig. 5. Chunk Size 5 Training Loss

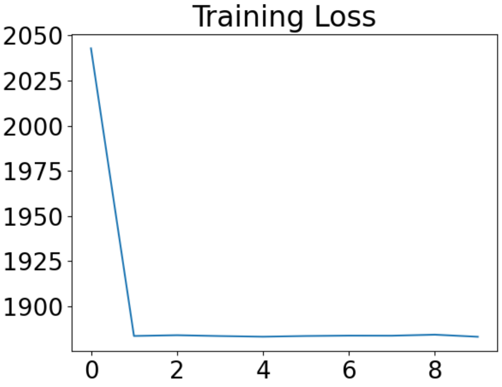

Fig. 6. Chunk Size 15 Training Loss

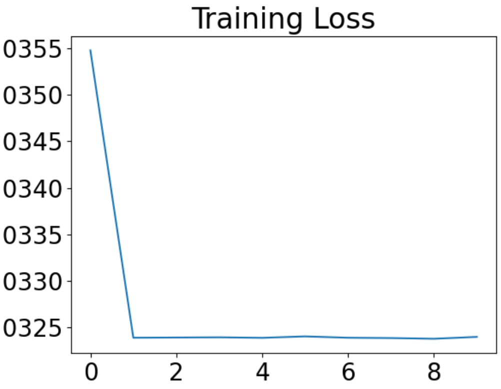

Fig. 7. Chunk Size 25 Training Loss

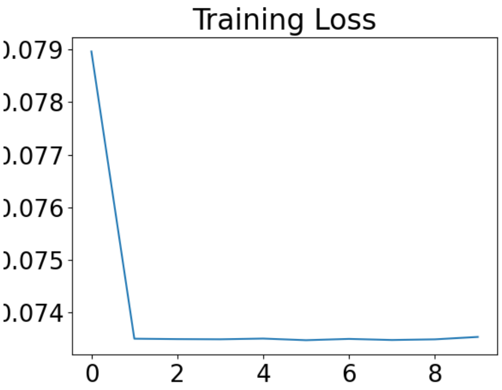

Fig. 8. Chunk Size 60 Training Loss

1) *Model 1:*

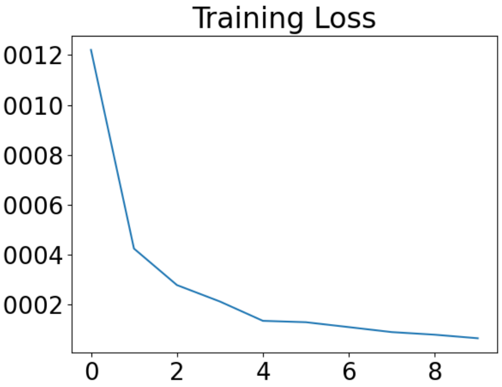

Fig. 9. Chunk Size 5 Autoencoder Training Loss

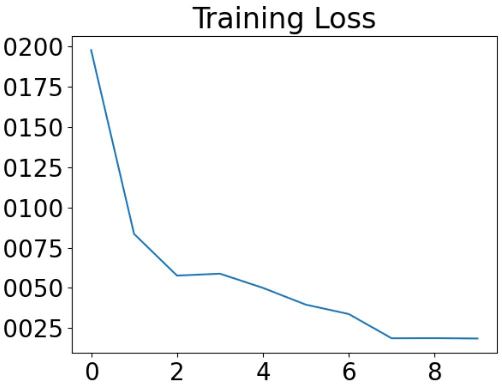

Fig. 10. Chunk Size 15 Autoencoder Training Loss

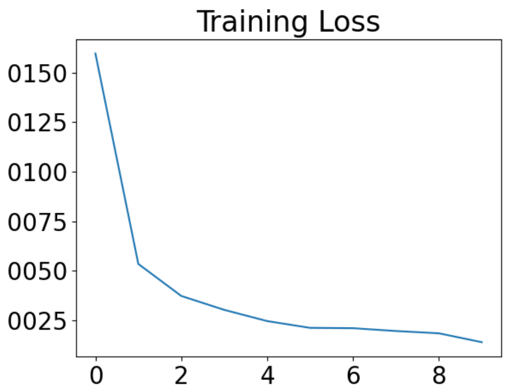

Fig. 11. Chunk Size 25 Autoencoder Training Loss

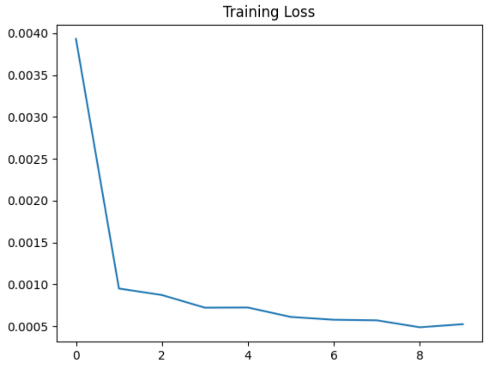

Fig. 12. Chunk Size 60 Autoencoder Training Loss

2) *Model 2:*

B. *Confusion Matrix*

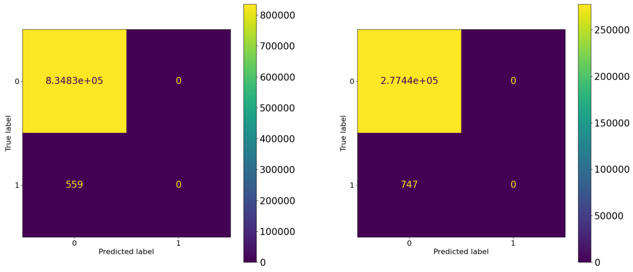

Fig. 13. Chunk size 5 and 15 for Model 1

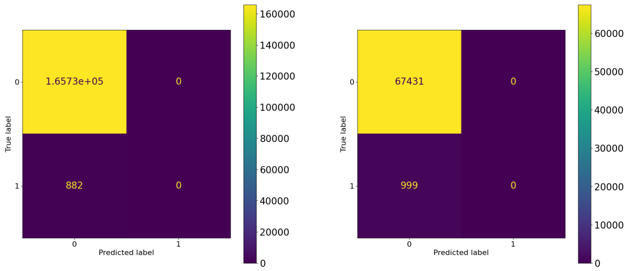

Fig. 14. Chunk size 25 and 60 for Model 1

1) *Model 1:*

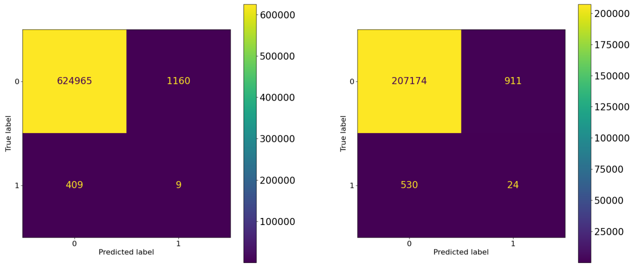

Fig. 15. Chunk size 5 and 15 for Model 2

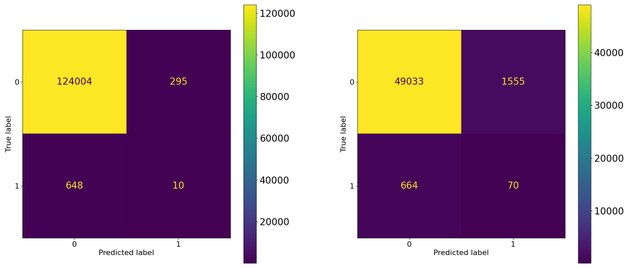

Fig. 16. Chunk size 25 and 60 for Model 2

2) Model 2:

References

[1] Massimo La Morgia et al. *Pump and Dumps in the Bitcoin Era: Real Time Detection of Cryptocurrency Market Manipulations*. May 2020. DOI: 10.1109/ICCCN49398.2020.9209660.

[2] Christopher Olah. *Understanding LSTM Networks*. Aug. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.