CRYPTO ROBO ADVISOR

Non Fungible Trio

Rubin Thomas, Maanasi Garg, Venkata J C Edala

1 PROBLEM

The central problem of this paper is to analyse the cryptomarket and see if a deep learning model can predict the fluctuation of prices in conjunction with sentiment data. To this end, we hope to construct a few models of varying complexity to help learn the patterns (if any) and help eliminate some of the uncertainty of this market. We believe that crypto currencies would have a prominent position in the future of finance and this paper would shed light into how people view the currency.

2 INTRODUCTION

It is well known that the market is a volatile beast. The crypto market is specifically more volatile and often guided by public sentiment.

Our interest in cryptocurrencies stem from the limitation of conventional currencies. Conventional credit and cast is often used to make transactions but it is overly controlled and only sustains small transfers of money. To this end, we believe cryptocurrencies have an important role to play in our future.

Researches have long been interested in predicting the prices of cryptocurrencies using simple models like linear regression and ARIMA. In our project we tried three models of differing complexities to see how well it would do. The models we used are: MLP, Transformers and LSTM.
For the data we have used prices from different exchanges over the years and additional data from google trends, reddit and twitter.

3 DATA ENGINEERING

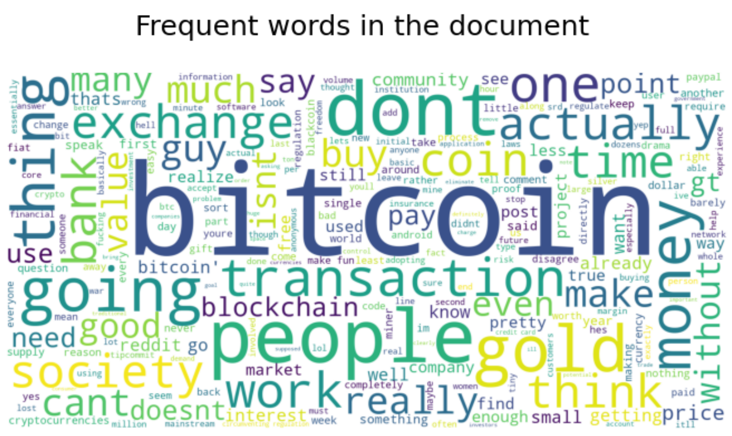


Fig 3.1 Reddit Bitcoin Comments word cloud

3.1 Data Extraction

For historical bitcoin price data, we extracted bitcoin's historical daily trading volume, and open, close, high, and low prices from the Bitstamp exchange. For daily sentiment data, we extracted for that day the number of Reddit that mentioned bitcoin, total number of likes on bitcoin related Reddit posts, total number of Twitter posts mentioning bitcoin, Google interest index value for the bitcoin keyword, and a weighted sentiment score on Reddit posts mentioning bitcoin.

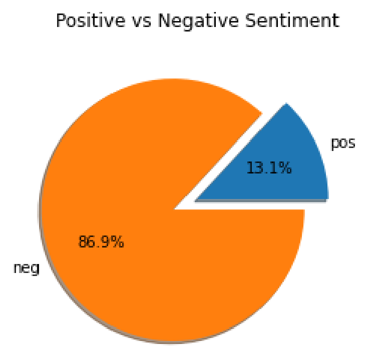


Fig 3.2 Sentiment Ratio in Reddit comments

The weighted sentiment score was calculated by passing each Reddit post mentioning bitcoin form that day into a pre-trained Hugging Face transformer sentiment analyzer model, and then taking a weighted average of the model's output sentiment scores based on the popularity of that post. This transformer model outputted a positive or negative label, along with a confidence score that we used as the numerical value of the sentiment score. Although this model was pre-trained on text that did not include Reddit posts, we manually looked through a good number of the sentiment labels to make sure they accurately reflected the post.

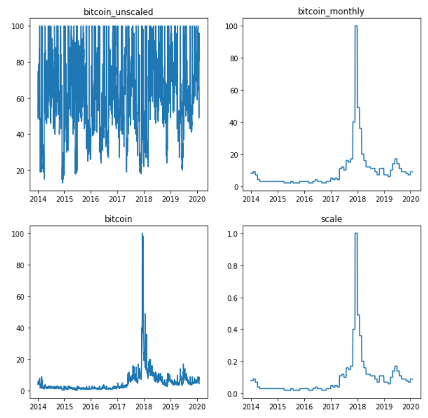


Fig 3.3 Google trends data for bitcoin

3.2 Data hyperparameters

Each of the models was tested across various data-specific hyperparameters and model-specific hyperparameters. The data-specific hyperparameters were common across all sections, and are defined here.

Data format: Different transformations were used on the training data before passing it into the models. Using "absolute data" for a model refers to using the raw, absolute pricing and sentiment data numbers as training data. Using "percentage data" refers to extracting day to day percentage changes in the pricing and sentiment data numbers as training data. Using "difference data" refers to extracting day to day absolute differences in the pricing and sentiment data numbers as training data.

Normalization technique: Different normalization techniques were used on the training data post transformations. These techniques included 1) min max scaling, which scales the data on a 0-1 scale, 2) standard scaling, which centers the data around a mean of 0 and standard deviation of 1, and 3) the lack of any normalization.

Sequence length: The number of days of historical price/sentiment data that is used to predict the future price. In other words, if sequence length is 14, then each training data item contains 14 days of historical price/sentiment data, and is used to output one value for the future price.

Prediction day: The number of days in the future that the model attempts to predict the price for.

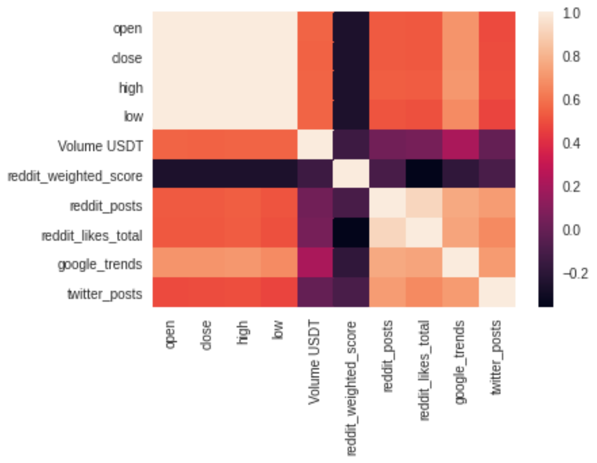
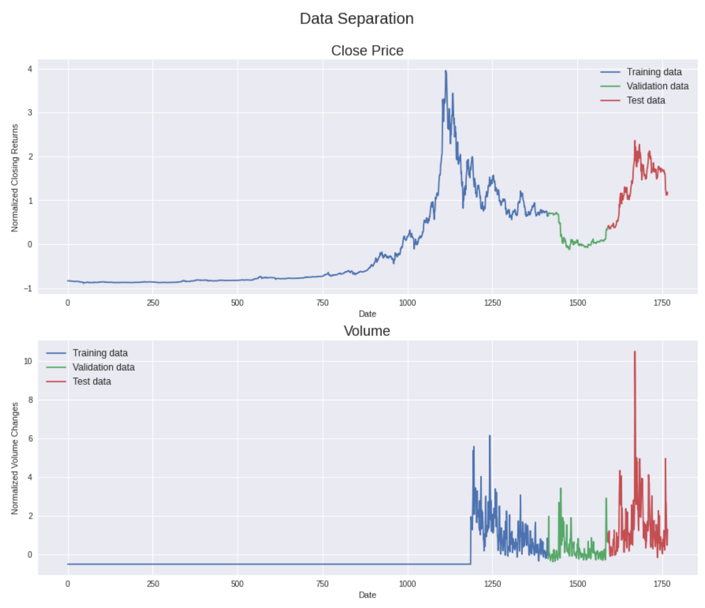


Fig 3.4 Correlation matrix of the features

3.3 Data splitting

For every model, we split the data into 80%, 10%, and 10% for the training, validation, and testing datasets, respectively. The time series data is not shuffled before splitting it into the training, validation, and testing datasets. In other words, the data from 2014 to around mid 2018 was used for training, mid 2018 to early 2019 for validation, and early to late 2019 for testing. This split up will best test whether historical correlations will carry over to the future. Shuffling the data would have probably led to artificially inflated testing results.



4 MODELS

We experimented with three models, MLP, LSTM and Transformers

4.1 MLP MODEL

The first model that we tried was the simplest model option: a MLP model. Since a MLP model does not have any inbuilt time-series attention mechanism, it is the simplest model that could be used for our task. After experimentation and hypertuning, we decided on using ReLU activation, MSE loss, Adam optimization, and a learning rate of around 3e-5.

4.2 TRANSFORMER:

The next model under consideration was a transformer model. A transformer model is a neural network that uses self-attention to give higher weights to the relevant parts of the time series data that make it easier for the prediction. We will be using a Multi head attention layer which will consist of Single head attention layer all constructed using Keras from Tensorflow.

As for the input data we first use a time2vector which will be discussed in the following sections, which is then concatenated to the actual data. The transformer also has multiple dropouts and batch normalization. After the self attention layer, the data is passed through a few conv1d layers to output a value of the required size.

4.2.1 Architecture

Time2vector:
One of the more novel changes to these transformers is that the ordering of the different days in the sequence is encoded in two extra fields: periodic and non-periodic time features. The equation to derive these two feature are as below:

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \leq i \leq k. \end{cases}$$

time2vector non-periodic and periodic

The intuition behind this feature is that the non-periodic feature captures the movement in time whereas the periodic feature holds the information about the seasonality in the data. This function is performed over the mean of every other feature in each instance of the sequence.

Self Attention heads

The attention heads are very similar to the ones commonly used in most literature for NLP models. We have three inputs: Query, Key , Value. All three of these inputs are the sequence itself, given that we are in need of a self attention head.

Our multi head attention is a list of single attention heads. The outputs of these heads are then concatenated before passing it through a fully connected layer

Transformer Encoder layer

The transformer encoder layer gets its input from the previous attention heads. The inputs are then run through a few Conv1D layers with kernel size 1. This effectively makes it a fully connected layer. There are "ReLU" activations between these layers. There are dropouts (15%) and batch normalizations as well that only function in the forward pass.

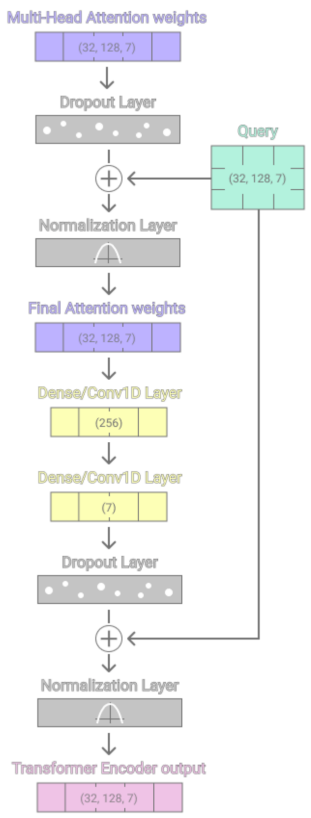


Fig 4.2 : Transformer Encoding layer

4.2.2 Training

The complete data while training included the difference time-series data for price (absolute), the differential price and the sentiment scores (reddit, google trends, twitter). The training was done with both the entire data as well as part of the data to see how well it would learn. It was also found that using a standard scaler gave better results so all the training for this model was done with such a normalization.

Training was done for 35 epochs which takes around 2-5minutes.

4.3 LONG SHORT TERM MEMORY MODEL (LSTM):

LSTM models are chosen since they are powerful sequence predictions because they are able to store past data. The assumption was that LSTM would be better at extracting the long term patterns that predict the price.

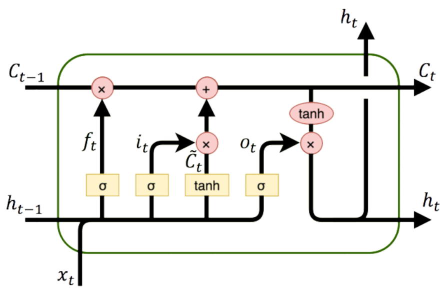


Fig 4.3 : LSTM Layer Architecture

4.3.1 Architecture

The architecture was made with the help of Keras. After experimenting with a different number of LSTM layers, it was found that 2 layers were optimal for our purposes. Using "ReLU" activation in between the layers seemed to worsen performance. A dropout layer of 15% was added in between the layer for just the forward pass. There is finally a dense layer at the end to shape the output in the format we require.

4.3.2 Training

The training for the again consists of the whole dataset used for the transformer with different runs having complete and partial data.

The model was run for 50 epochs and the losses converged quickly. Mean Squared Error was the loss function used, with Adam Optimizer and Metrics as

Mean Absolute Error and Mean Absolute Percentage Error

5 RESULTS

The following are the results for the different models after using different section of data for training

5.1 MLP

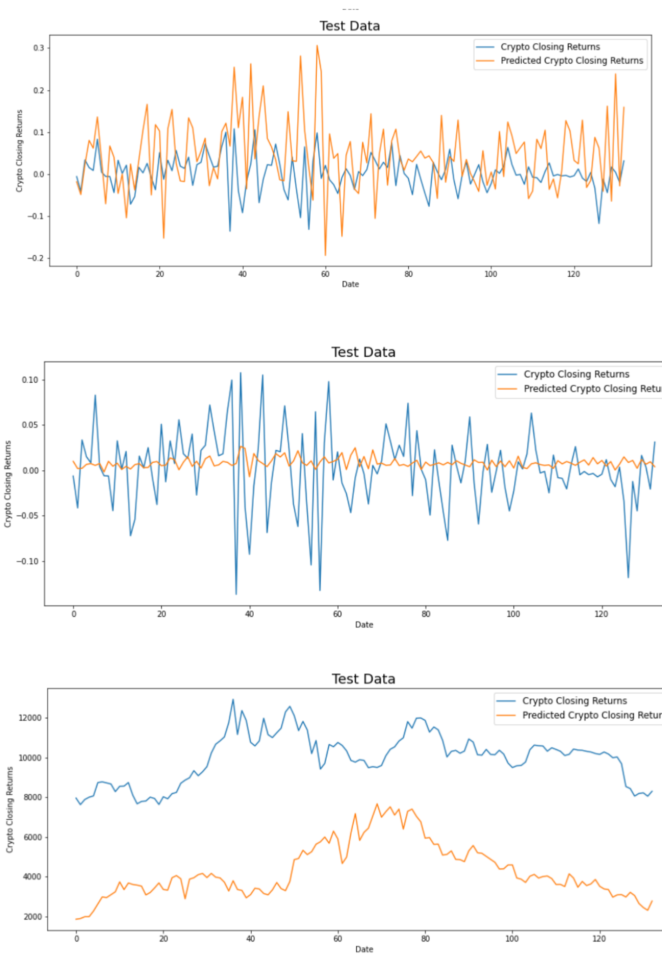


Fig 5.1: a (top), b (mid), c (bottom)

We ran over 500 iterations of the MLP model using different configurations of the data hyperparameters listed above, and model hyperparameters, particularly batch size (8 vs 12), architecture ([50,30,10] vs [120,30,10] vs [70,20]), and batch normalization. The results were generally poor, with the MLP model consistently over predicting, under predicting, or very chaotically predicting the returns. Initially, when we got very chaotic predictions from the MLP model **(Fig 5.1 a)**, we tried reducing the number of parameters with a smaller and narrower architecture, and adding batch normalization. The architecture change did not fix the chaos, and batch normalization created the opposite effect of very suppressed predictions (**Fig 5.1 b)**. Any configurations that had reasonable volatility were poor at capturing timing and direction of the price movements (**Fig 5.1 c)**. We concluded that the MLP model might be too simple for this task, and moved on to trying more complex models that are designed for time-series data.

5.2 Transformer

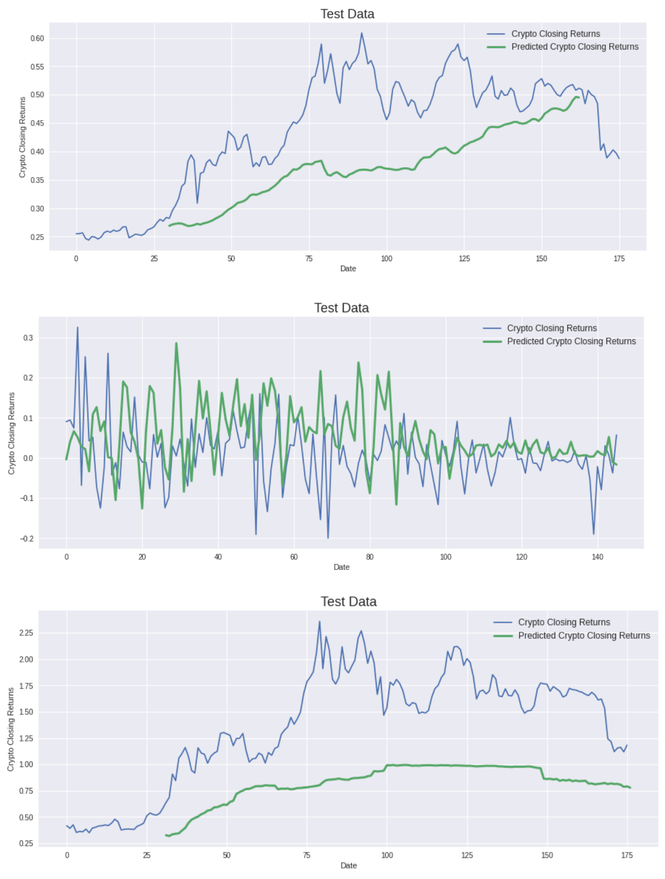


Fig 5.2: a(Top), b(mid), c(bottom)

The transformer model was run over multiple sections of the training data. One of the notable runs was when we ran it over just the data with economics features (absolute) (**Fig 5.2 a**). On this run none of the sentiment data was used and the model seemed to perform fairly well. The challenge was that it was unable to predict the actual value instead it only predicted the trend. This led us to the second try (**Fig 5.2 b**) where we just used the percentage price point to predict the percentage increase or decrease in prices. Unfortunately, this was too volatile for the model to do too good of a job on. The last graph (**Fig 5.2 c**) was using absolute price data, percentage price data and sentiment data. Although the noise from the sentiment data makes it difficult for the model to predict the exact value of the price, it does a much better job of predicting the trend.

5.3 LSTM

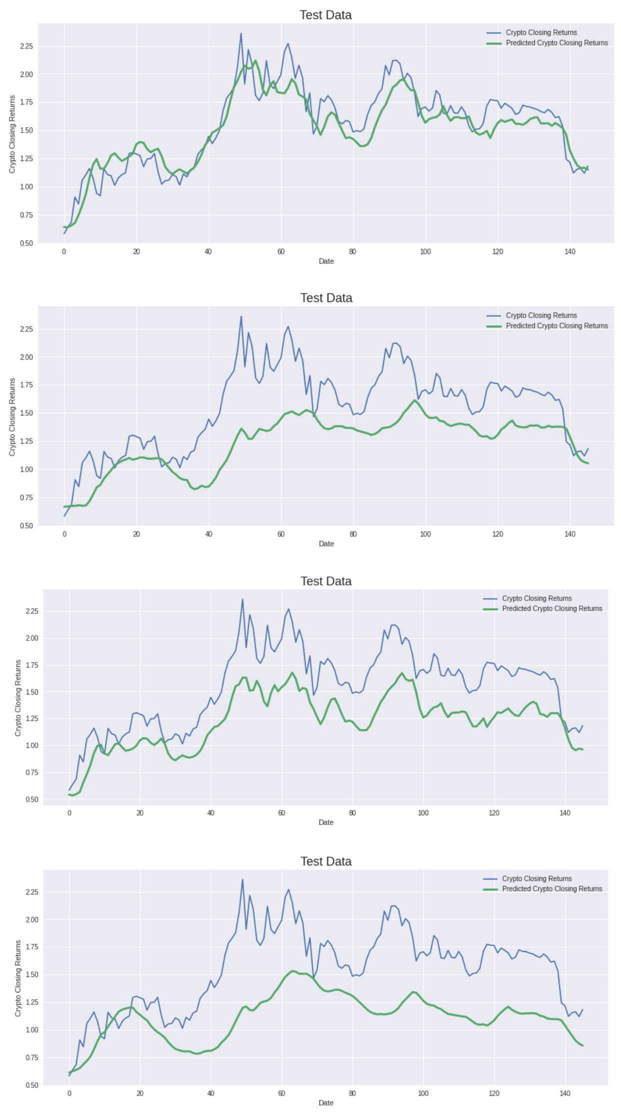


Fig 5.3: a(Top), b(top-mid), c(bottom-mid), d(bottom)

The first two graphs above are for the next day predictions. The one without the sentiment analysis (**Fig 5.3 a**) appears to perform better than the one with the sentiment analysis data (**Fig 5.3 b**). This is similar to the results we obtained from the transformers and might suggest that the sentiment data is adding too much noise. Other than this, this seems to be our best prediction.

The next two are for a 7 day away prediction. Again we see a similar trend where the one without sentiment data (**Fig 5.3 c**) outperforms the one (**Fig 5.3 d**) with that data.

5.4 Overall

As expected, our MLP model, which was our baseline performed the worst. The transformer was able to extract more pattern from the chaos but it too was not sufficiently good. The LSTM model on the other hand performed quite well.

Model	Prediction Day	MSE		
		Train	Val	Test
LSTM With Sentiment	1	0.053	0.0477	0.219
^	7	0.004	0.03	0.2564
LSTM Without Sentiment	1	0.005	0.05	0.03
^	7	0.012	0.03	0.135
Transformer With Sentiment	1	0.0350	0.0813	0.6421
^	7	0.0332	0.0916	0.6260

Table : MSE loss over LSTM/ Transformer models

Model	Day	MAE			MAPE		
		Train	Val	Test	Train	Val	Test
LSTM Sentiment	1	0.036	0.16	0.40	11.67	729	23.83
^	7	0.039	0.15	0.44	14.09	489.1	26.65
LSTM Without Sentiment	1	0.044	0.18	0.16	19.9	969	10
	7	0.07	0.16	0.331	20.76	731.2	20.56
Transformer With sentiment	1	0.129	0.24	0.747	58.57	832.3	47.34
	7	0.113	0.25	0.741	47.79	838.9	45.3

Table : MAE/ MAPE loss over LSTM/ Transformer models

6 CONCLUSION

6.1 Performance of Model

Even though the model performs the trends of the price increase quite well even a week from the predicting date, there is still a lag in the pattern it produces. The model might perform better if we try to predict something like a moving average of prices, but that would defeat the purpose of predicting prices in the stock market. In conclusion, even though it is a worthwhile endeavour try these models in the market, we believe it will be of little use. This is due to the fact that the cryptocurrencies are a fat tailed distribution and certain days (the end of 2018), account for most of the variation in the time series data. It is exactly these black swan events that are more important than any of the in between points. So any technique that uses correlation to predict is not going to do too well, at least not for day trading.

6.2 Reasons for failure and ways to improve

One potential reason for failure is that the pre-trained Hugging Face transformer was not effective at analyzing the sentiment of Reddit posts. Reddit users tend to use a lot of slang, jargon, and generally non-colloquial language that the transformer might not recognize. Since we did not have true sentiment labels for the Reddit posts that we collected, and since there were >4M posts, we did not have the time or capability to label the posts and use the true labels to tune/retrain the sentiment model. On top of that, running the Reddit posts through the pre-trained transformer model itself took 7-8 hours, so tuning or re-training might have taken an unrealistically long amount of time. If we did not have these constraints, we would have ideally loved to use a sentiment analyzer model that is trained to our dataset.

Another potential reason is the general lack of data, in terms of quality and quantity. Bitcoin has only been traded since ~2014, which means we only had around 2,000 days of price and sentiment data to use for all of model training, validating, and testing. We couldn’t even use 2020 data, since we could not find scraped Reddit bitcoin posts from 2020. We considered upsampling our data and using hourly data instead of daily data - however, we did not have access to some of our sentiment data points on an hourly basis. We also thought that the volatility in hourly prices would just add more noise to the model, since intraday volatility that generally does not have much correlation with the overall price trend.

Some additional data points we could maybe try adding include S&P 500 data, in case stock market activity affects bitcoin pricing, and time proximity to the next bitcoin halving.

6.3 Social impact:

Our motivation for doing this project was to build a personal cryptocurrency robo advisor that could empower the common retail investor to make better investing decisions and grow their personal wealth. Investing is an essential strategy to maintaining and growing personal wealth, and there is an increasing gap of investing knowledge between institutional investors that are available to the wealthy, and the common retail investor. Although we were not able to achieve results that we would trust for investment decisions, we hope that deep learning eventually empowers less wealthy individuals to make intelligent investing decisions.

On the other hand, bitcoin mining is known to contribute heavily to greenhouse gas emissions, due to the amount of electricity that is used to solve the computational puzzles required for mining. According to Digiconomist's Bitcoin Energy Consumption Index, bitcoin has a carbon footprint comparable to that of New Zealand, producing 36.95 megatons of CO2 annually. However, there is increasing awareness of using renewable energy sources for mining. For example, half of global bitcoin mining takes place in Sichuan China, but 95% of this mining is done using renewable hydroelectricity, rather than nuclear or coal based energy. Hopefully, bitcoin is increasingly mined using renewable energy and its greenhouse gas footprint continues to go down.

7 GENERAL DISCUSSION

Data limitations were probably our biggest constraint in training this model, and potentially our biggest obstacle to beating our baseline model. The Bitcoin data was chosen because it is old and is available for the required period of time along with reddit data whereas the other coins have limited data as they are newer. But the successful model can be used for predicting any crypto currencies given the sufficient market and sentiment data.

We know that Bitcoin is receiving lots of criticism for its bad environmental strategy (700KWh energy consumption per Transaction) and highly marketelized nature, but they are other coins out in the market which are relatively environmentally friendly XRP (0.0079KWh), Dogecoin (0.12KWh) etc. We chose bitcoin, not because we support it, but only because it is the coin with the most amount of data associated with it. We believe that we could transfer learning in order to use this trained model to then train on any coin of our choosing.

REFERENCES

[1] Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures https://arxiv.org/abs/1808.08946

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding https://arxiv.org/abs/1810.04805

[3] Time2Vec: Learning a Vector Representation of Time https://arxiv.org/abs/1907.05321

[4] Attention Is All You Need https://arxiv.org/abs/1706.03762

[5] A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial Institutions https://www.sciencedirect.com/science/article/pii/S2214212620307535