Forecasting When To Buy Bitcoin Using ResNet with Squeeze Excitation Along With ARIMA and GARCH

Yang Nguyen, Ryan R. Thackston, Fatemeh Tale, Hien V. Nguyen

Take-Home Messages

- Forecast modeling using ResNet with Squeeze Excitation alongside ARIMA and GARCH Statisticla Methods
- ResNet with Squeeze Excitation was able to correctly forecast model price fluctuations 78% of the time.
- This is target towards the financial industry and will help decreasing volatility in the market while giving a profit for users in day trading
- The significance is using ResNet with Squeeze Excitation Machine Learning Method alongside ARIMA and GARCH statistical methods.

Forecasting When To Buy Bitcoin Using ResNet with Squeeze Excitation Along With ARIMA and GARCH

Yang Nguyen, Ryan R. Thackston, Fatemeh Tale, Hien V. Nguyen

Abstract The vast majority of deep neural networks that are trained utilize common optimizer and pooling methods like stochastic gradient descent algorithm, MaxPooling and Average pooling. Over time there have been methods to improve model training performance through various pooling and optimizers functions that aren't commonly used. In this paper, we propose utilizing various optimization algorithm and various pooling methods like Ranger, Blurpool, and GlobalWeight Pooling to associate in the training process with a residual neural network that uses a squeeze-excitation network to forecast a prediction on buying, selling, or waiting Bitcoin. Most of the approaches towards analyzing Bitcoin Prices were using exponential smoothing with Long short term memory (LSTM) combined with Natural Language Processing (NLP) to predict price. Instead of trying to predict prices, why can't we decide on investing or not? Our goal is to utilize the ARIMA (Auto-Regressive Integrated Moving Average) and GARCH (Generalized Auto-Regressive Conditional Heteroskedasticity) time series forecast models to assist in making a decision and process the time series with labels into a Residual Neural Network (ResNet).

***Keywords* — Resnet, CNN, Squeeze Excitation, Blurpool, GlobalWeightPooling, FitOneCycle, Lookahead, ARIMA, GARCH, Bitcoin, Forecast Modeling,**

I. INTRODUCTION

Time series forecasting in business can involve a range of different challenges such as predicting sales for a company, exact prices on stocks, predicting supply chain loads, predicting long-term trends, etc. Many forecasting methods used involve statistical models, classical machine learning models, and even deep learning models. A great introduction to time series forecasting methods include [1] in which various statistical and classical machine learning methods are evaluated and compared when forecasting for the M3 competition dataset. At the time, simpler statistical methods still widely outperformed machine learning methods. To evaluate the model's performance, this paper uses symmetric mean absolute percentage error (sMAPE), model fitting (MF), and computational complexity (CC). The forecasting models and evaluation methods are primarily concerned with matching a forecasted price.

In **The M4 competition: 100,000 time series and 61 forecasting methods** [2] the top performing models tended to be combinations of statistical and machine learning methods. Evaluation metrics are sMAPE, mean absolute scaled error (MASE), and overall weighted average (OWA). Again, these evaluation metrics emphasize forecasting an exact value. Forecasting exact prices is a much more difficult problem to solve but forecasting when to buy, sell, or wait has many more correct answers so we decided to test our model against classified labels. Because combinations of statistical and machine learning methods tended to do better in the M4 competition, we decided that our model should be similar. We wanted to measure when the data had relatively higher variance in certain periods and make a decision whether to buy or sell during that period. Other papers were also useful in learning about different time series forecasting models used in research and industry [3],[4],[5],[6].

II. METHOD

A. Data

Bitcoin-US Dollar Tether market data was gathered from the Bitfinex API. Metadata was gathered in 1-minute intervals and included the date, open price, close price, highest price, lowest price, and volume of Bitcoin exchanged during each interval. Market data was gathered from June 1, 2021 at 6:00 AM UTC to November 30, 2021 at 10:00 PM UTC. The hardware that we computed it on was using a CPU processor Intel Core i7-11800H Processor 8cores /16 threads 2.30GHz Turbo 4.6GHz 24MB Cache and the GPU NVIDIA GeForce RTX 3080 16GB GDDR6. The packages that we used for this experiment were python is 3.9.7, Keras version is 2.4.3, TensorFlow 2.5.0, Cudnn 8.2.1, and Cudatoolkit 11.3.1.

ARIMA

*A*RIMA models are used in time series forecasting to understand the data or predict future points within the data [7]. We created a fitted ARIMA model using the $\log_{10}$ of the "open price". From this open price we were able to measure the p lag order, d degree of differencing, and q size of the moving average window. ARIMA forecasts and forecast errors were then calculated.

GARCH

The GARCH statistical models measure the volatility (a measure of relative variance) of specific windows of time series data where the variance error is assumed to be serially auto-correlated [8]. Residual errors measured from ARIMA were added to the GARCH method as weights to measure the volatility of the market.

Labeling

Price data was labeled buy, sell, or wait based on the median GARCH conditional volatility as a threshold and the ARIMA forecast error. If the GARCH conditional volatility was below the median conditional volatility of all the data, then that datapoint in time was labeled "wait". Else if the GARCH conditional volatility was equal to or higher than the median, then the market was considered volatile at that time and an action needed to take place. The ARIMA forecast error was then measured for that datapoint. If the ARIMA forecast error was negative then the forecast price was lower than the actual value in which case we would want to buy in a volatile market so we label the data point "buy". Else if the ARIMA forecast error was positive, the forecasted price was higher than the actual price so we label that data point "sell".

B. *Model*

Overall Model

For our model, we devised a customized Convolutional Neural Network (CNN) that would utilize residual blocks that contain squeeze-excitation networks based on various pooling methods, activation function, and dimensionality of the output channels. Our model contains 8 convolutional layers where each of the following layers would be followed by a batch normalization and swish activation function. This model can be flexible in terms of its output channel dimension by saying dim = x, where x can be any number of choices. We decided to choose dim = 32 for the output channels that coexist with each convolutional layer. The first few convolutional layers will have an output channel of 32. As it progresses through each residual block, the output channel for each residual block would be dim = 32 * 2^n, where n is the current residual block starting at n=0. That indicates that we only have three residual blocks contained in the overall mode where each residual block output channel is 32, 64 and 128 respectively. Residual connections are used to improve gradient propagation throughout the network. We chose our parameter where the length and batch are 256. Figure 1 shows the flowchart for the overall model.

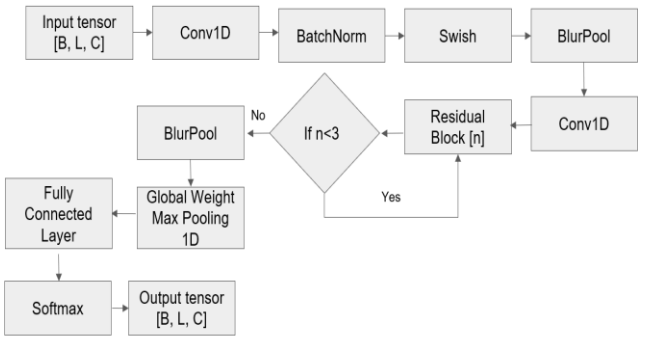

Figure 1: The flowchart of the overall model used in this study.

Residual Block

The skip connection in the residual block is made so that the training process of the model would be faster because when calculating the gradient for back propagation, it can decide to either go back to the input or go back to the previous layer. Identity shortcut connections simply perform the identity mapping and their outputs are added to the output of the stacked layer. Identity shortcut adds neither extra parameters nor significant extra computation. The advantage of this skip connection is that if any layers hurt the performance of the model, it will be skipped by regularization. The skip connection would prevent vanishing and exploding gradients from occurring during the training process of deep neural networks [9]. Figure 2 illustrates the flowchart related to the structure of the residual block.

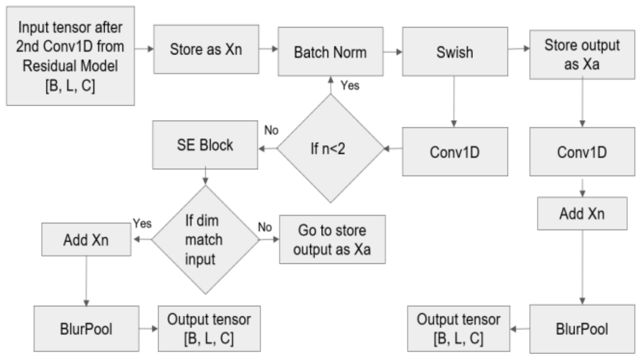

Figure 2: The flow chart of the residual block used in this study.

Squeeze-Excitation Block

The Squeeze-Excitation network weights each channel adaptively by converting the output of a weighted layer into a vector of features. The weights through the vector of features get scaled with the output of the weighted layer through element-wise multiplication. This element-wise multiplication through the weighted layer indicates which features are imposed as most important [10]. Figure 3 shows the structure of the Squeeze-Excitation block used in this study.

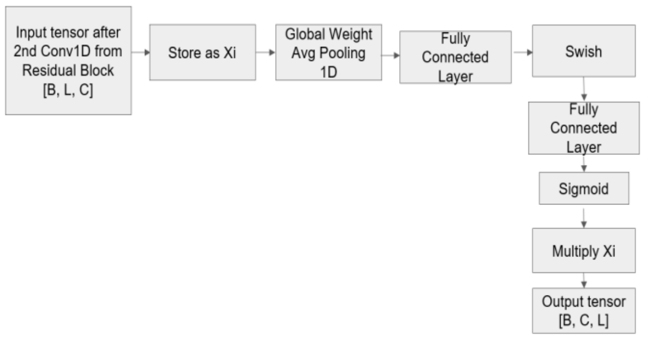

Figure 3: The flowchart of the Squeeze-Excitation block used in this study.

The Squeeze-Excitation block is given an input convolutional block and the current number of channels the block has. The convolutional block is then squeezed for each channel to a single numerical value using average pooling. A fully connected layer followed by a swish function adds smoothness for nonlinearity and the output channel is reduced by a certain ratio . A second fully connected layer followed by a sigmoid activation function gives each channel a smooth gating function. The Convolutional block is weighted for each feature map based on the results of the squeeze-excitation block. Adding a squeeze-excitation block adds barely any extra computation to a Residual network and the output is similar to a residual network that contains a significant amount of weighted layers.

C. Pooling

BlurPool

Most CNN utilizes downsampling methods like max-pooling, stride convolution and average pooling. However, they ignore a crucial theorem in signal processing called the Nyquist sampling theorem [11]. Because of this ignorance, this results in CNNs becoming shift variants and aliasing can occur during training because a small shift in the input can drastically affect the shape of the output. An over-aggressive filtering with downsampling can result in a heavy loss of information which reduces performance. What Blurpool proposes is that we can convolve the signal with a low pass filter so that the shifts in the inputs don't impact the output relatively and the feature layer gets impacted equally, making feature extraction generally stable. Making the model shift-invariant and shift-equivariance [12]. The BlurPool method chooses a blur kernel based on the kernel size. Each filter, depending on the kernel size is a discrete convolution of one another. For example, if the kernel size is equal to 2, then the filter is [1, 1]. If the kernel size is 3, then the filter is a convolution of [1, 1] with [1, 1] making the filter being [1, 2, 1]. If the kernel size is 5, then the filter is a convolution of the between the two filters when kernel size was 3. The output for filter when the kernel size is 5 is [1, 4, 6, 4, 1].

GlobalWeightPooling

Through CNNs, the convolution operation analyzes the data in a slight shifting window manner. Global weighting methods are used to bridge the localization of data points and the classification in fully connected layers. Picking the nearest neighbor from the average or maximum of all weights during the fully connected layer in the CNN helps emphasize the weights for the features. Global Weight pooling allows the network to learn what features are strongly emphasized when you have a standard field of view. Global Weight pooling can control the reduction in variance in the input domain since the point of interest is always at the center and the neighbor next to it. Global weight pooling can learn this relationship, especially through the feature axis [13].

D. Optimizer

Stochastic Gradient Descent

Stochastic Gradient descent uses a learning rate of 0.0001 and a momentum of 0.9. Stochastic gradient descent is applied with our learning rate scheduler that uses the OneCycle policy.

Rectified Adam

Adaptive learning rate optimizers struggle to generalize in the very beginning of training and suffer high variances. What rectified Adam does compared to most adaptive learning rate optimizers is that it trains a model with a low initial learning rate and it will get tuned with momentum for the first few batches.

Lookahead

Lookahead maintains a set of fast weights and slow weights, which get synced with fast weights for every k update through another optimizer [14]. We can take a step based on an inner optimizer like ADAM or SGD, and we will continue this for k steps. We can consider slow weight as the save point and fast weights as K steps into the future. We linearly interpolate in the parameters space along the direction of the fast weights and go to the specific direction of the fast weights by some factor alpha. The fast weight that the slow weights approaches becomes the new slow weight and then you continue to take K more steps with your inner optimizer and continue to interpolate in parameter space. In the end, we combined Lookahead with Rectified Adam to create an optimizer called Ranger [14].

E. Learning Rate Scheduler

FitOneCycle

FitOneCycle was used for making an LR scheduler that incorporated momentum, batch size, and weight decay to tune the learning rate during the training process. This would yield the model training process to converge faster because it will obtain the optimal learning rate resulting in a robust model. The way momentum plays a role in the learning rate is that the range of momentum stabilizes the training process instead of getting a learning rate too big or too small. Using a small momentum value when the learning rate increases provides the training process to allow a higher increment of learning rates. The idea is that we start with a high momentum value and decrease momentum as the learning rate increases. A larger value in momentum will speed up the training process but can cause the model to diverge. Batch size impacts how much data is input into a model at a time. The bigger the batch size, the less regularization occurs during training and more overfitting. This implies that having a bigger batch size should use a larger learning rate. Weight Decay implies a balanced form of regularization to obtain good training results. The weight decay depends on how deep the CNN is. A shallow network would require more regularization which suggests higher weight decay [15]. We chose a learning rate range between 0.0001 and 0.001, momentum range between 0.85 and 0.95, a batch size of 128 and weight decay of 0.

F. Layer

Swish

The Swish activation function is the combination of the Sigmoid activation function and the input data point. Swish is a smooth continuous function, unlike ReLU which is a piecewise linear function. Swish allows a small number of negative weights to be propagated through while RELU thresholds all negative weights to be zero [16]. There could be some relative pattern that can be analyzed from those negative values compared to ReLU which turn anything negative to 0. Swish is non-monotonic and an unbounded saturation function that can contain negative samples which makes it superior to ReLU. The trainable parameters allow better tuning to the activation function to maximize information propagation and push for smoother gradients. Because Swish is non-monotonic, it improves gradient flow which assists in finding the most optimal learning rate. The smoothness of Swish plays a huge role in optimization and generalization by reducing the sensitivity to initialization and learning rates, resulting in the model converging to a minimum loss [16]. Since Swish also prevents the idea of vanishing gradients and exploding gradients, it is always going to be superior to ReLU.

Batch Normalization

Batch normalization normalizes a layer's input by taking the z-score of the input. It ensures that the input distribution of every neuron will be the same as the model converges. Batch normalization improves the convergence of deep neural networks because it solves the internal covariate shift and vanishing gradient issue. This results in improving the learning rate and reducing overfitting. Batch normalization also reduces the opportunity of any small changes getting amplified after each interaction through many neurons by normalizing the input data. Batch normalization mitigates domain shifts because it puts each neuron activation into a gaussian noise distribution of zero means and unit variance.

III. EXPERIMENT

For experimenting with our model, we decided to have 2 categories for model training and have 3 different pooling methods based on Blurpool. The categories for this experiment are using the Ranger optimizer vs using fitonecycle with stochastic gradient descent. Each category utilizes blurpool, Average Blurpool and Maxblurpool. What we learned is that average Blurpool performs significantly better than Blurpool and Maxblurpool and that FitOneCycle helps the model converge faster with training time compared to the Ranger optimizer. The comparison between them was that FitOneCycle training time was 20 seconds while Ranger was trained in 70 seconds. The Ranger optimizer however performs better than Stochastic gradient descent even though stochastic gradient descent uses a FitOneCycle. We trained all these models with a batch size of 256 and the length of the sequence of 256

Model Category	Blurpool	Optimizer	Batch Size	Epoch	End Epoch	Time (s)
FitOne	Regular	SGD	256	50	16	17
FitOne	Average	SGD	256	50	26	17
FitOne	Max	SGD	256	50	42	18
Ranger	Regular	lookahead	256	50	26	68
Ranger	Average	lookahead	256	50	20	70
Ranger	Max	lookahead	256	50	17	62

Model Category	Blurpool	Train Loss	Train Acc (%)	Train CCE	Valid Loss	Valid Acc (%)	Valid CCE
FitOne	Regular	0.6992	64.71	0.7039	0.6142	77.11	0.5483
FitOne	Average	0.6397	66.55	0.6661	0.5801	77.11	0.5329
FitOne	Max	0.6060	67.69	0.6083	0.6164	74.65	0.6250
Ranger	Regular	0.5232	71.74	0.5755	0.6096	71.92	0.5960
Ranger	Average	0.5526	69.49	0.6083	0.5416	75.59	0.0568
Ranger	Max	0.5607	69.38	0.6169	0.5993	74.72	0.5829

IV. CONCLUSION

There were initially slight variations in the results when we tested the forecast model using our own labeling method and the Sharpe Ratio but after running a few training cycles and adding the best weights to re-train the model, the results became similar.

FitOneCycle appears to be good for generalizing a learning rate faster in order to improve convergence of models. If this was applicable to any optimizer besides stochastic gradient descent then models would improve in all aspects of training and performance. Using Lookahead to predict the best weights for an optimizer to approach was a good idea since it will improve the learning rate and loss values. GlobalWeight Pooling is best used to grasp the feature towards each sequence from a distinct perspective per batch to see correlation between each sequence. Blurpool was best to make sure that the model wouldn't lose information and make the training process more stable. In the future, we hope to get FitOneCycle versatile with other optimizers and see how well FitOneCycle with other optimizers would correlate with distinct pooling methods that would improve model training.

V. FUTURE WORK

We believe that if we were able to combine FitOneCycle with the Ranger optimizer then we are going to be able to obtain the best optimal learning rate for the model and assist in converging the training process with half the time. The flaw in stochastic gradient descent is that when deciding on a learning rate that is too small, it can make it painfully converge and having a learning rate too large can cause the training of the model to diverge. Even though this problem is somewhat solved with FitOneCycle, it still struggles to converge compared to other optimizers.

If we could apply a transformer to be used for Natural Language Processing so that we could incorporate social media into account while training the model, then we could encode each critical word that can impact on the prices to dictate the decision of buy, sell or wait. We can then determine a multi-categorical classification problem where the decision can be determined by the forecast values with a keyword that impacts the decision to buy, sell or wait.

If there is also a potential way to view the signal as an image, we could start with pretrained weights from ImageNet instead of just randomized weights. Having weights that have been pre-trained for various images should shorten the training process of the model so we would run with less epochs. The data for images can use a variety of indicators if convolution for signal data can be applied in 2D. This way, there would be more features that can be extracted from an image compared to analyzing a sequence that could improve the accuracy of making a decision.

Utilizing boosting methods to take current data to be used for data prices. However, this method is probably not good for deep neural networks because it would suffer from overfitting as boosting methods are best for shallow neural networks. During the training process, anything that was incorrectly predicted gets loaded back into the input of the model like a feedback loop and reconfigures the weights until training accuracy is 100%. What we are trying to consider here is that we could use this boosting method to not train the model but configure the weights in real time so that the current price value would be processed into the model to make the next prediction.

Another suggestion is using Sharpe ratios as an alternative approach to our original labeling scheme. The Sharpe Ratio is the ARIMA Forecast minus the ARIMA forecast error divided by the GARCH conditional volatility [18]. If the

Sharpe Ratio is above 1, you label the data point as “buy”. Else if the Sharpe Ratio is below 1, you label the data point as “sell”.

REFERENCES

[1]S. Makridakis, E. Spiliotis, V. Assimakopoulos, “Statistical and Machine Learning forecasting methods: Concerns and Ways Forward”, PLOS One, 2018.

[2]S. Makridakis, “The M4 competition: 100,000 time series and 61 forecasting methods”, PLOS One, 2019.

[3]H. Hewamalage, C. Bergmeir, K. Bandara “Recurrent Neural Networks for Time Series Forecasting: Current status and future directions” *International Journal of Forecasting*, 2020.

[4]S. Smyl “A Hybrid Method of Exponential Smoothing and Recurrent Neural Networks for Time Series Forecasting”, *International Journal of Forecasting*, 2019.

[5]E. Spiliotis, V. Assimakopoulos, S. Makridakis “Generalizing the Theta method for automatic forecasting” *European Journal of Operational Research*, 2020.

[6]D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, “DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks”, Amazon Research Germany, 2020.

[7]S. McNally, J. Roche, S. Caton, 2018. Predicting the Price of Bitcoin Using Machine Learning. Proc. - 26th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. PDP 2018 339–343.

[8]L. Horv, P. Kokoszka, 2003. GARCH processes: structure and estimation. Bernoulli 9, 201–227.

[9]K. He, X. Zhang, S. Ren, J. Sun., 2016. Deep residual learning for image recognition. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016–December, 770–778.

[10]J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 2011-2023, 1 Aug. 2020.

[11] P. Colarusso, L.H. Kidder, I.W. Levin, E. Neil Lewis, 1999. Raman and Infrared Microspectroscopy, in: Lindon, J.C.B.T.-E. of S. and S. (Ed.), . Elsevier, Oxford, pp. 1945–1954.

[12]R. Zhang, “Making Convolutional Networks Shift-Invariant Again” presented at the 36th International Conference on Machine Learning, Long Breach, California, Jun 2019.

[13]S. Qiu, “Global Weighted Average Pooling Bridges Pixel-level Localization and Image-level Classification”, arXiv, Sep 2018.

[14]M. Zhang, J. Lucus, G. Hinton, and J. Ha, “Lookahead Optimizer: k steps forward, 1 step back” ” presented at 33rd Conference on Neural Information Processing Systems, Dec 2019.

[15]L. Smith, “A Disciplined Approach to Neural Network Hyper-Parameters: Part 1 - Learning Rate, Batch Size, Momentum, and Weight Decay” ” US Naval Research Laboratory Technical Report, April 2018.

[16] H. Yong, J. Huang, X. Hua, L.Zhang, “Gradient Centralization: A New Optimization Technique for Deep Neural Networks”, arXiv, Sep 2018.

[17]P. Ramachandran, B. Zoph, Q. Le, "Searching for Activation Function," arXiv Google Brain, 27 Oct. 2017.

[18]Lo, Andrew W “The Statistics of Sharpe Ratios” in *Financial Analysts Journal*, January 02, 2019.