Extracting Generalizable Information from BitCoin Price Charts

Rishi Puri: puririshi98@berkeley.edu

May 2019

1 Introduction

It has long been assumed that numerical prediction of market prices given prior price data is essentially impossible. Despite this there are a plethora of highly successful investors who are able to combine information from media available to them such as financial reports and charts, without pouring over lists of numbers and using some mathematical model to predict. Taking this as inspiration we establish a method of using state of art deep computer vision techniques to extract features from the patterns apparent in BitCoin price charts to achieve accuracy of up to 87% on held out data when predicting whether a chart indicates a future rise or fall in price.

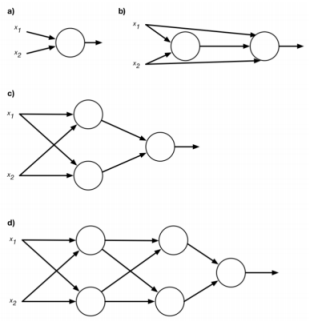
2 Literature Review

This study is building upon prior work from https://github.com/philipperemy/deep-learning-bitcoin where using a training set of only 10,000 images they were able to achieve 70% validation accuracy using AlexNet. This paper intends to improve upon these results using a larger training set and using a resNet-18 model to mitigate vanishing gradients thanks to the residual connections.

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	2	512×2 fully connections
softmax	2	

Figure 1: we modified the final layer to have 2 outputs

We expand this study further by analyzing the results under the framework of the generalization/memorization tradeoff. In order to do this we establish the concept of memory equivalent capacity for a Neural network, which is defined by these 4 rules:



1. The output of a single perceptron yields maximally one bit of information.
2. The capacity of a single perceptron is the number of its parameters (weights and bias) in bits.
3. The total capacity C_{tot} of M perceptrons in parallel is $C_{\text{tot}} = \sum_{i=1}^{M} C_i$ where C_i is the capacity of each neuron.
4. For perceptrons in series (e.g., in subsequent layers), the capacity of a subsequent layer cannot be larger than the output of the previous layer.

a) 3bits, b) RESNET: 3bits+4bits=7bits, c) 2*3bits+3bits=9bits
d) 2*3+max(2*3,2+2)+max(3,2+1)=6+4+3=13bits

Figure 2: These calculations are made assuming a learned bias for each perceptron

[1]

For an interactive demo on how to calculate the capacity of any network visit http://tfmeter.icsi.berkeley.edu

Its counterpart, the memory equivalent capacity of a dataset, has two bounds:

1. Maximum memory equivalent capacity which is simply the size in bits of the lookup table for the data set. (upper bound)

2. The estimated memory equivalent capacity is found by building a static-parameter(weights set to identity; biases are learned) machine learner to memorize the training data. This is, the memory-equivalent capacity can be minimally logarithmic of the size of the static-parameter machine learner.(lower bound)

The true capacity of the dataset will be somewhere between these upper and lower bounds depending on how random it is. Ideally we want to make a nueral network of capacity equal to the dataset's capacity in order to generalize.

The concept of memory equivalent capacity is essential for generalization which is the concept of handling different objects by a common property. For optimal generalization we follow Occams Razor: among equally accurate models, choose the one that requires the lowest memory-equivalent capacity. As an example the series 2,4,6,8 can be generalized by +2. When the maximum capacity of a dataset is its true capacity this implies that the data is entirely random and generalization is impossible. In order to gauge if there is generalizable information present in a data set we use the idea of generalization progression.

This is performed by estimating the capacity, as described above, needed to memorize 10%, 20%, . . . , 100% of the training table. If the capacity does not stabilize at the higher percentages either there is not enough training data or the data is too random to generalize.

Once we have a trained model we can evaluate its generalization using the formula

$$generalization\ ratio = \frac{Number\ of\ Correctly\ Classified\ Instances}{Capacity\ of\ Machine\ Learner\ Used} \tag{1}$$

A model with generalization ratio < 1 requires more data or is trained sub-optimally, while a model with generalization ratio of 1 is simply memorizing. A model with generalization significantly higher than 1 is truly generalizing while a model with generalization ratio slightly above 1 is somewhere in between full memory and generalization which we define as associative memory,also known as content-addressable memory.[2] This occurs when the learner has insufficient capacity to fully memorize and thus uses the majority of its memory capacity to memorize a large portion of the dataset and then uses the remaining capacity to implictly learn an associative function. This associative function works to match un-memorized examples to their closest(using a learned distance metric) memorized counterpart.

3 Experimental Setup

3.1 Data Set

Using the CSV of 18 million price points, the 550,000 image train and 50,000 image validation set were created by randomly sampling slices of size 40 from the CSV and passing them to matplotlib-finance. The classes are balanced(49.99:50.01). These charts are pre-processed by cropping out the graph axes(which contain numerical values) leaving only a visual pattern of prices. This random sampling method introduces the inherent flaw that the train, validation and test sets may have minor overlaps. However due to the immensity of data to sample from, the probability that the data sets contain identical examples is very low.

3.2 Optimizer

All networks were trained using an SGD optimizer with an initial lr of .001 and batch size 32, with an evaluation in between every epoch to save the weights that perform best on the validation set during training. We used a plateau LR scheduler which decreased the LR by a factor of .3 every time the validation accuracy plateaued for 1 epoch in the case of the ResNet and a factor of .9 for a plateau of longer than 4 epochs for all other networks.

4 Experiments and Results

4.1 ResNet Experiment

The dataset was used as input to train a resNet-18 model. We utilized the optimization settings described in section 3.2 to train our network. Training proceeded particularly well, converging to 99% training accuracy and 87% validation accuracy. Using the rules described in Figure 2, one can calculate that the memory equivalent capacity of the resnet-18 model is 201730 bits. With 550000 number of correctly classified instances, this leads to a generalization ratio of 2.72 according to equation (1).

4.2 Ablation Studies

The pretrained ResNet-18 model is used to embed the data set from images of dimension 224x224x3 to vectors of dimension 512. Through ablation studies, this embedded version of the dataset can be utilized for further analysis studying the generalization properties of our learned feature extractor.

4.2.1 Architecture Search

The max(upper bound) and estimated(lower bound) memory equivalent capacity of the training set was computed in addition to the generalization progression. We created a fully connected net that has the same memory equivalent capacity as the estimate for the training data. For unsatisfactory accuracy results, the size of the network was increased repeatedly until adequate accuracy was achieved. This led to an architecture search across the following networks:512-18-2,512-40-2,512-40-40-2,512-80-40-2,512-160-40-2,512-400-40-2,512-4000-40-2.

Once embedded into 512 features the data set had an estimated memory equivalent capacity of 9216 bits and a maximum of 90074388 bits according to the rules established in section 2. Using the generalization progression tool, we found that the memory equivalent capacity of the data set caps off at just 70% of the data, see Figure 3. This informs us that there should be enough information present in the data to generalize.

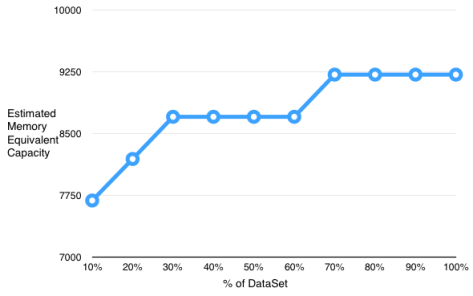


Figure 3: Generalization Progression

Even though there should be enough information, upon attempting to train a fully connected neural net on a dataset with approximately 9216 bits of capacity, we saw that this lower bound estimate for capacity was insufficient to achieve high training accuracy for such a random problem as Bitcoin price prediction. Training a neural network with a similar capacity of 9234 bits, the 512-18-2 network (512 features and 1 bias multiplied by 18 hidden neurons gives 9234) resulted in a best training accuracy of 53%. This gives a generalization ratio of 31.6 with a validation accuracy of 70.8% and 291500 correct training classifications. In order to find the network of proper size for higher training accuracy we searched our predefined architecture space until satisfactory results were found. Eventually we landed on a network with layer dimensions, 512-4000-400-2 which was finally able to reach high training accuracy(any smaller model would never pass 60% training accuracy). This architecture was able to achieve 78.3% training accuracy resulting in 50.1% validation accuracy, which, with a memeqcap of 3653202 bits, gives a generalization ratio of .118. This architecture was highly over fitted. In the end the 512-18-2 network performed the best with 70.8% validation accuracy.

Architecture	Train Acc	Val Acc	Capacity	Generalization Ratio
ResNet-18	99.99%	87%	201730	2.72
512-18-2 Net	53%	70.8%	9234	31.6
512-4000-400-2 Net	78.3%	50.1%	3653202	0.118

Figure 4:

4.2.2 Feature Capacity Comparison

To determine if any small subset of the features was more pertinent to prediction, we enumerated all 512 choose 5 combinations of features and measured their estimated memory equivalent capacity to see if any combination is lower than the others. Such analysis is important because if any combination of features has a lower estimated memory equivalent capacity than the others, this implies that an easier classification task (since a smaller neural network is required) can be created by using that group of features over any other group. Unfortunately, no size 5 combination of the 512 features had a lower estimated memory equivalent capacity than the others.

4.3 Final Ensemble Test

Finally, we trained an ensemble of 15 randomly initialized networks of the best performing size(512-18-2) to maximize validation accuracy. We trained each ensemble member on bootstraps of 25% of the data following the same optimization scheme as before. These were then evaluated on a freshly generated 2000 image test set embedded by the resNet as a final test accuracy metric. The 512-18-2 x 15 ensemble achieved a final test set accuracy of 77.8%.

5 Analysis

The resNet-18 was far above the estimated memory equivalent capacity of the image data allowing it to get near perfect training accuracy. Despite having much more memory equivalent capacity than the data's estimated capacity, the network still performs very well on unseen data which shows that the network is still able to generalize. It is somewhere in between complete generalization and true memory, creating an associated memory. Due to the near randomness of the labels, a neural net sized at the estimated memory equivalent capacity of the embedded data was not sufficient to classify the data set with near perfect accuracy. By repeatedly doubling its size we reached a sufficient size to classify the training data, however the learner was not able to perform as well on unseen data.

Even with using the more "general" small model at estimated memory equivalent capacity, an ensemble still underperforms the associative memory of the resNet in terms of validation accuracy. Since the 500,000 image dataset likely contains most price patterns of bitcoin, the resNet must memorize as many as its memory equivalent capacity allows, and use a small portion of this capacity to create an implicit associative function. This function serves as a content based indexing into this memory where two visually similar price patterns(e.g. slightly higher/lower peaks at some points in an otherwise identical graph) that differ by a few perturbations in pixel space (which are very easy to compress away using convolution and pooling) will be mapped to the same label in memory. Once this associative memory is created it is actually quite general as most of the unseen data will take the form of an already memorized price pattern(due to the immensity of training data) with a slight perturbation, as the training set is so immense.

In contrast, The sweep of feature combinations confirmed that all 512 features are nearly equal in importance to classification. Since all features of the embedding are equally important, and the spatial information was killed by reshaping into a vector, it is a much harder problem to form an association between two seemingly arbitrary vectors with the same label. This is likely due to the fact that perturbations in this space are very high dimensional(with each dimension being equally important). Unlike this, certain regions in images are inherently a lot more pertinent to classification(something that convolutions and pooling are designed to deal with) than others. This implies that the high

dimensional perturbations across the image in this space can be modeled by fewer dimensions implying that generalizing the perturbations is possible. With this power a huge computer vision architecture can memorize most of the data and then learn an associative function between new examples and memorized ones. This is why over parametrized models excel at forming associative memory for image inputs, but fail for feature vectors, where a fully general model is necessary to do well on unseen data.

6 Reproduction

These experiments are documented in https://github.com/puririshi98/Bitcoin-And-Generalization and repeatable by following the instructions outlined in the readMe section. Although the random seeds of the networks parameter initializer were not saved, through trial and error training results tended to be very similar regardless across multiple trials with the same hyperparams. Due to this, these results are still easily reproducible by using the same process outlined in the GitHub. All models used in these experiments are available for use in the GitHub.

7 Discussion

It has long been assumed that no mathematical model on the price points of a security is able to accurately predict a future price change. Despite this there are a plethora of swing traders who make millions by littering the walls of their office with generalized, hand drawn, charts of various price patterns labeled with what they USUALLY indicate about future returns, created based on their past experiences and intellect. In doing this they are creating an associative memory in their mind where they have memorized the outcomes of the set of patterns on their wall, and have learned an associative function used to map new price patterns they see to their similar, memorized counterpart.

This real world example carries great importance in this study as it gives human learning intuition to the supervised learning task being explored. Through exploring memory equivalent capacities and generalization of the embedded data we have shown that although it is theoretically possible to generalize from vectorized features of bitcoin charts(as the generalization progression capped off), in practice, it is very difficult to train a neural network to get high accuracy at the estimated memory equivalent, due to constraints in gradient based training techniques and the loss of spatial information. Despite the low training accuracy, this model has a very high generalization ratio and achieves adequate accuracy on unseen data(since anything significantly higher than 50% allows for monetary gains).

We have found that the problem of accurate bitcoin chart prediction from images is best solved somewhere along the spectrum of complete generalization and full memorization, with the concept of an associative memory. This associa-

tive memory is far easier to form using resNet due to the spatial nature of price charts being perfectly tailored to convolutional approaches. While the accuracy results of the resNet are very impressive, the higher generalization ratio of the small fullNet implies that it may be more robust across the unexpected changes in bitcoin prices. Since its accuracy is still significantly above 50% on unseen data, this opens a potential for making money in the market with a general prediction algorithm, and, therefore,is still a success.

These results also hold importance outside the world of financial prediction. In general machine learning we attempt to not use a significantly larger model than necessary in order to avoid over fitting, which is shown when using the oversized fullNet on the featurized data caused complete failure to unseen data. Despite this, the current trends in computer vision involve using ever larger models(for example resNet-152, which is orders of magnitude larger than my resnet-18 is often used on imagenet which is not significantly larger than my dataset) but somehow they are still able to perform very accurately on unseen data, and significantly smaller models fail to do as well on training or validation sets. This research has helped uncover one possible reason why. The generalizable perfect golden rule differentiating a plane and a car may not exist or be too complex to find using gradient methods. However, by creating an associative memory with these overparametrized computer vision models, such as ResNet-152, on sufficient amounts of data(like imagenet) we can memorize almost every possible version of a class to which same label examples differ only slightly, and simply learn to associate similar unseen examples to those memorized. This is helps to explain how our oversized resNet18 succeeded on unseen data when trained on the immense stock chart dataset.

7.1 Future Directions

One particular issue we had was that a neural network struggled to achieve the same training accuracy in the embedded space as the ResNet achieved in pixel space. This is likely due to pitfalls in the training methods and in future work we intend to attempt a much wider array of training techniques to remedy this issue.

We will train sentiment analysis on bitcoin and crypto related hashtagged posts from twitter. We will embed data in sentiment networks feature space and go through a similar pipeline of testing generalization on the the featurized data and creating appropriate memory equivalent capacity sized fully connected ensembles to classify these features. We will also test the results of using the last layer of spatial resNet features before they are flattened into a vector(in order to preserve spatial information), and train a small convolutional network to classify.

Finally, we will test accuracy and generalization progression improvements of combining the sentiment features with the spatial resNet features, and ultimately use the predictor to write a bitcoin auto trading script.

8 Conclusion

In this study we have found that residual convolutional networks excel at the task of predicting the rise and fall of bitcoin using price charts. Through the use of the generalization progression tool we have found that the resNet used is indeed generalizing when embedding the input data into 512 features. Using these features we have created an ensemble of small learners that accurately predict(77.8%) on unseen data. Although these networks are considerably less accurate than the resNet their higher generalization ratio suggests that they may be more robust to the shifts in distribution that constantly occur in financial markets. These experiments have highlighted an interesting dichotomy between the ideas of associative memory and generalization which require further study as they are still fledgling concepts.

References

[1] G. Friedland. *Measuring Generalization in Machine Learning.*

[2] D. Mckay. *Information Theory, Inference, and Learning Algorithms.*