A TOPOLOGICAL ANALYSIS OF CONVOLUTIONAL NEURAL NETWORK FILTERS

ADRIAN LOPEZ, DANNY GENDLER, JOHN PETERSON, JOHN HARER

Abstract. Convolutional neural networks, mainly used for image recognition, have been proposed as a powerful tool for time series classification. To understand how CNNs are able to perform time series classification, we use topological data analysis methods on data sets consisting of the filter weights of CNNs trained on financial data. Using methods established in previous research, we conduct several experiments on these point clouds, obtaining persistence diagrams and comparing them for different volatility of underlying data, different layers of a CNN, and for different types of deep neural networks. We analyze persistent homology using persistence diagrams, Wasserstein distance, and the Mapper algorithm.

Introduction

During the last two decades, Time Series Classification (TSC) has been considered one of the most challenging problems in data mining. With the increase of temporal data availability, hundreds of TSC algorithms have been proposed since 2015. Recently, Deep Neural Networks (DNNs) and more specifically, Convolutional Neural Networks (CNNs) have shown promise in the field of TSC with neural architectures such as ResNet, Fully Convolutional Network (FCN), and multi-layer perceptron (MLP) showing promising performance.[4]

Many studies have examined the visual cortex of animals and found that their neural networks are separated into layers responsible for picking up features, which increase in complexity as you move into deeper layers. This idea inspired the creation of deep neural networks.[9] One of the first attempts at mimicking this was the creation of the multi-layer perceptron algorithm, which consists of fully connected linear computational units learns based off of the minimization of a loss function by using the optimization algorithm known as stochastic gradient descent.

The convolutional neural network (CNN) is another deep learning architecture which makes use of convolutional layers. These layers slide spatial filters (small areas) across regions of the pixels of an image, with dimensions $a \cdot b \cdot c$ (where a and b represent the width and length of the image in pixels and c represents the number of pixel values). These spatial filters are arrays of numbers, or weights, which have width w and height h. Filters allow us to identify certain features of images. If the filters are identifying vertical lines in an image, for example, the weights of the filter will be high along the columns of the filter. Sliding a filter across the pixels of an inputted image outputs an activation map or feature map, which allows us to recognize where in an image a certain feature is occurring. More than one filter may be used in a convolutional layer, allowing for the detection of several types of features. The different convolutional layers of a CNN detect different features. While the first convolutional layer might search for low-level features such as edges and lines, which will be shown in its activation map, each activation map that follows it represents

Key words and phrases. Topological Data Analysis, Persistent Homology, Deep Neural Networks, Explainable AI, Statistical Learning Theory.

increasingly complex features, such as circles, spirals, and even faces. Finally, CNNs input the activation map outputted from the final convolutional layer into a fully-connected layer. This layer determines what complex or high-level features have the highest correlation with certain classes of image types by assigning probabilities to each class. Before training begins, the weights of the filters are randomized. CNNs then constantly adjust the values of the filter weights through the training process - over time, the weights are adjusted to better capture the essence of small regions of an image, and learn basic global structures.[2] As training ensues, these global structures also change.[7]

The success of CNNs in image recognition has inspired research into the use of CNNs for time-series classification. By viewing multivariate time series as images in space-time, we can utilize CNNs to discover and learn the useful features of these space-time data sets.[4] This can be incredibly useful for the identification of short-term patterns, which is useful in the prediction of time-series data such as financial recordings and physiological readings. In traditional image data, pixels that are near each other are more correlated. Because time-series data is temporally correlated as well as spatially correlated, we are able to treat the data as an image by using a sliding-window approach.[6]

In this paper, we plan to provide a topological analysis of data sets created using the filters of CNNs trained on time series data. By doing so, we can better understand how the global structures learned by the weights of the CNNs change, and also develop an intuition for how to interpret this in the context of time-series classification. This could lead to insight into the methodology behind the learning process of CNNs. It will also allow us to better understand whether any topological features are correlated with improved performance of certain CNNs and even other DNNs. We aim to do so through the following three experiments. We first plan to study how the topological features and structures of CNN filters respond to being trained on datasets with varying volatility. We also aim to test how the topological structure of these filters changes throughout different layers, and whether there are any significant topological features present in all of the layers. Since CNN filters learn more complex features as you move from layer to layer, understanding their topological structure could give us better insight into how they learn these more complex features. Finally, we hope to identify whether there are any differences in the topological structure of the filters of different DNNs, which would explain why they learn in different ways or achieve superior performance compared to other DNNs.

Background

Finance

Time series analysis on financial data, especially high-dimensional time series data, is difficult due to the occurrence of both short- and long-term events which are hard to detect and/or predict. Complex financial systems often experience these events, or anomalies, due to changes in broad market sentiment, such as market crashes like the subprime crisis in 2007-2008 or the dot-com crash in the early 2000s, as well as due to idiosyncratic or asset-specific shifts or risks, such as regulatory issues, individual company bankruptcies, or supply gluts and shortages (in the case of commodities). The efficient market hypothesis claims that markets efficiently reflect all past information which is publicly available in current-day prices, since the flow of information is unimpeded. It follows that any information that can be predicted or detected should be reflected in the current price of a security – and thus stock prices a day from today can only change as a result of new information that comes between the present and a day from today.[1] Since the occurrence of most financial anomalies is difficult or impossible to predict, stock prices (and the prices of other securities) are similarly difficult to predict given only public information. While the key tenets of the efficient market hypothesis still largely hold true, it has received criticism in the last few decades on the basis that asset prices are at least partially predictable and that markets can sometimes be irrational. Indeed, evidence has shown the existence of short-term pricing irregularities and predictable patterns in asset price and return data. It was this notion that led us to choose to train and test our CNNs on financial data. Since there is at least some degree of predictability for return data, a predictive model could identify these patterns and irregularities, although past efforts to do so have proved difficult.[8] Thus, we use financial data as a proxy for a complex time-series system on which we can train CNNs. The specific data we use consists of daily closing price, return, and volume data for 11 publicly traded securities with varying volatility, for a period of 4 years between January 1st, 2015 and January 1st, 2019. In order to avoid studying assets which all belong to the same asset class (and which would thus be likely to display significant correlation of return data), we chose securities from a number of different asset classes, including commodities, equities, credit, and alternative investments. These securities are classified by their volatility, which we will calculate as the annual standard deviation calculated on percentage returns, as displayed in the formula below:

$$\textbf{Annual Volatility Formula} = \sqrt{252} \times \sqrt{variance}$$

A summary of the security data is presented in **Table 1** below. Note that the only security for which weekend price data is available is Bitcoin, resulting in an increased amount of data points.

Table 1

Security	*Ticker*	*Data Points*	*Annualized Volatility*
Dow Jones US Real Estate Index	*DJUSRE*	1,042	14.55
S&P Listed Private Equity Index	*SPLPEQTY*	1,042	12.78
WTI Crude Oil	*CL1 : COM*	1,042	39.26
Corn	*C1 : COM*	1,042	21.61
Dow Jones Industrial Average	*INDU*	1,042	13.98
Nikkei 225 *Index*	*NKY*	1,042	20.54
Financial Times Stock Exchange 100 *Index*	*UKX*	1,042	13.08
iShares 7 − 10 *Year Treasury Bond ETF*	*IEF*	1,042	5.22
iShares iBoxx& High Yield Corporate Bond ETF	*HYG*	1,042	6.35
Bitcoin	*XBTUSD*	1,462	44.67
ZKB Gold ETF (USD)	*ZGLDUS.SW*	1,042	12.78

Time Series Classificiation

Time series data is an ordered set of data points ordered by chronological order. Time series classification is central to this experiment and is the process of training a model on time-series data that finds a function $f(\mathrm{x}|\theta^*)$ that maps input data to an output. The input data can be univariate, in which only variable corresponds to one output, or the input data can be multivariate in which multiple variables are associated to an output.

A dynamical system is a system that explains the behavior of a point through time. This system can be described through time series data by taking a sample observation of certain aspects of a point through time and recording them in chronological order. Many types of systems can be described as dynamical systems, including the motion of a pendulum, the evolution of wavefunction in many-particle systems, and the change of a stock price when measured with respect to time. Some systems such as the motion of a pendulum are deterministic, and given enough information, one can determine the future state of the system. Others are stochastic, such as the evolution of a wavefunction, where purely random events play a role in the evolution of the system.[10]

Deep Neural Networks

A deep neural network (DNN) is an artificial neural network with hidden layers where distinct computational procedures take place. The goal of DNNs is to find a function $\mathrm{y}=f(\mathrm{x}|\theta^*)$ that maps the input data to an output in an accurate way by optimizing the values of θ^* through backpropagation during training. To do so DNNs have hidden layers where backpropagation takes place which uses gradient descent to find the values of θ^* that come closest to the true θ. It is thought that each hidden layer learns and focuses on distinct different aspects of the input data (such as in images one layer may learn edges, another may learn faces, etc.) and combines their relationship in usually a non-linear fashion to approximate θ. Many different DNN architectures exists such as convolutional neural networks, recurrent neural networks, and multi-layer perceptrons.

Multi-layer Perceptron(MLP)

An MLP is a deep neural network consisting of many neurons organized into distinct layers, with 500 neurons in each layer and each neuron in a layer connected to a neuron in the layer that precedes it (otherwise known as fully-connected layers). It has three distinct components: an input layer, hidden layers, and an output layer. Except for the input layer, each one of these components uses a non-linear activation function, ReLU, to introduce non-linearity into neuron outputs.[4]

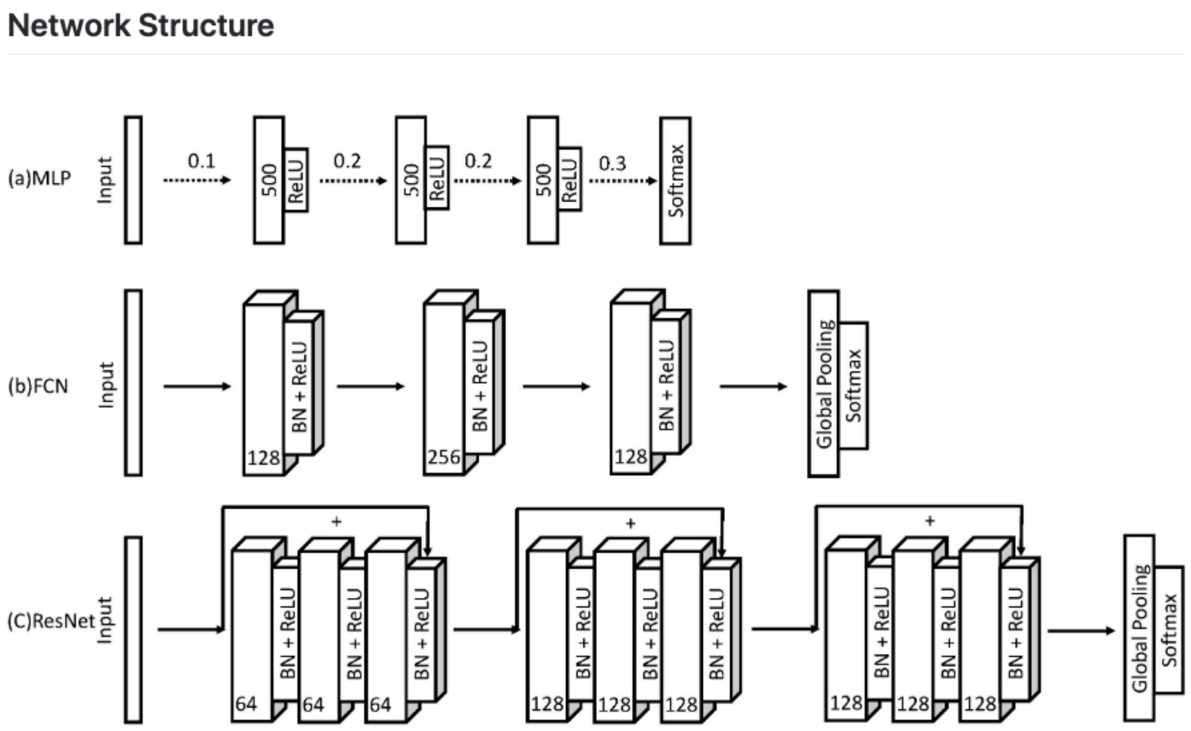
Fully Convolutional Network(FCN)

An FCN is a convolutional network composed of 3 convolutional blocks, with each block performing three tasks: convolution, batch normalization, and non-linear activation using ReLU. Fully convolutional networks replace the final fully-connected layer with a pooling layer which decreases the amount of parameters and allows the network to better understand which sections of the input time series are most critical for classification.[4]

Residual Network (ResNet)

ResNet is a very deep convolutional network with many convolutional layers, a pooling layer, and a classifier. It features shortcut connections between convolutional layers in each of its three residual blocks, which allows for more direct gradient flow.[4]

We illustrate the structure of these networks below:[4]



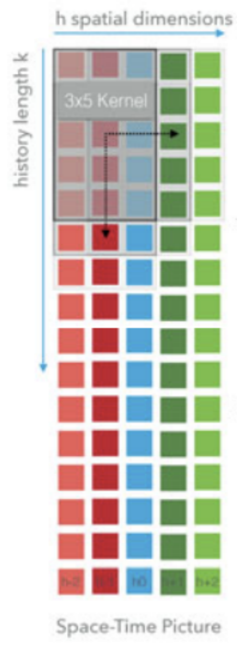
Persistent Homology

Topological data analysis, or TDA, is a field which allows one to study the shape and structure of data by identifying topological features such as connected components and loops. TDA makes use of the concept of topological concept of homology, which allows us to characterize and count these topological features. It does so through persistent homology, which is an incredibly useful tool for understanding the topology of point clouds. By computing filtrations of simplicial complices, we can identify birth and death times of n-dimensional features, which can be plotted on persistence diagrams.

Method

Data Collection Pipeline

To train the deep neural networks, we took the approach specified by [5] and [6] on our multivariate financial time-series dataset. We provide an image of our "space-time picture" approach below. We essentially used a sliding window approach, since the stride is 1 in the CNNs we employed, in order to input subsequences of data. We also had 4 spatial dimensions: price, return percentage, volume, and "up or down" classification.



80% of the financial dataset was used for training, 10% was used for testing and 10% for validation. We then normalized the financial dataset and gave it the task to predict a "0" if the price of the asset went down the next day, and "1" if the price of the asset went up the next day, essentially predicting the value of the 4th "pixel" in the next row after the kernel. We used Adam as the optimizer and built the following deep learning models:

ResNet

We used the following layers in ResNet:

- Convolutional Layer with 8 × 8 filters with stride 1, followed by batch normalization and ReLU.
- Another Convolutional Layer with 5 × 5 filters with stride 1, followed by batch normalization and ReLU.
- Convolutional Layer with and 1 × 1 filters with stride 1, followed by batch normalization and ReLU.
- Another Convolutional Layer with 3 × 3 filters with stride 1, followed by batch normalization and ReLU.
- Convolutional Layer with 8 × 8 filters with stride 1, followed by batch normalization and ReLU.
- Another Convolutional Layer with 5 × 5 filters with stride 1, followed by batch normalization and ReLU.
- Convolutional Layer with 1 × 1 filters with stride 1, followed by batch normalization and ReLU.

- Another Convolutional Layer with 3×3 filters with stride 1, followed by batch normalization and ReLU.
- Convolutional Layer with 8×8 filters with stride 1, followed by batch normalization and ReLU.
- Another Convolutional Layer with 5×5 filters with stride 1, followed by batch normalization and ReLU.
- Another Convolutional Layer with 3×3 filters with stride 1, followed by batch normalization and ReLU
- Global Average Pooling
- Dense Layer

We trained the ResNet for 10,000 epochs using the weights after 2,000 epochs to compare between FCN, MLP, and ResNet. We used the ResNet weights after 10,000 epochs to compare between the volatility of the assets.

Fully Convolutional Network

The trained FCN model consisted of the following layers:

- Convolutional Layer with 8×8 filters (weights) with stride 1, followed by batch normalization and ReLU.
- Another Convolutional Layer with 5×5 filters with stride 1, followed by batch normalization and ReLU.
- Convolutional Layer with 3×3 filters with stride 1, followed by batch normalization and ReLU.
- Global Average Pooling
- Dense Layer

We trained the FCN for 2000 epochs.

Multi-Layer Perceptron

The trained MLP model consisted of the following layers:

- Dropout layer followed with a dense layer and ReLU.
- Dropout layer followed with a dense layer and ReLU.
- Dropout layer followed with a dense layer and ReLU.
- Dropout layer followed with a dense layer and ReLU.
- Dropout layer followed by a dense layer.

We trained the MLP for 2000 epochs.

Topological Data Analysis

To perform topological analysis on the CNN filters, we follow the basic procedure outlined by Gabrielsson and Carlsson in [2], modifying it as needed for the purposes of our different experiments. We first extract the filters for each convolutional layer in the model. We proceed by selecting an individual layer and obtaining an array of the weights for the filters of the given layer. The filter weights are next vectorized to create a point cloud of m points $\in \mathbb{R}^{128}$, with m the number of filters in our selected layer. Next, we mean-center and standardize the variance of the weight vectors. We then create a point cloud in $\mathbb{R}^2$ and a visualizable simplicial complex using the KeplerMapper algorithm. We first project the point cloud onto its 2 principal components using PCA 1 and 2 as our reference maps or lenses. We select kmapper.Cover as our covering algorithm, which allows us to group our data points into bins. We then use DBSCAN as our clustering algorithm, which creates sets of clusters based on how close data points are to one another (using the Euclidean distance metric). Finally, we visualize the resulting simplicial complex so we can interpret the topological features of the projected data, with edges drawn between two clusters if they share at least one data point. We also analyze the 0D and 1D persistence of the point cloud outputted from Mapper by creating persistence diagrams of the projected data. This is achieved through filtration of the Vietoris-Rips complex, as previously mentioned.

Results

Volatility Analysis

Figures 1-11 show persistence diagrams for the filters of the last convolutional layer of the ResNet CNN, trained on the time-series data for the 11 aforementioned assets. We sort the assets by asset class and label the persistence diagrams by the underlying asset studied.

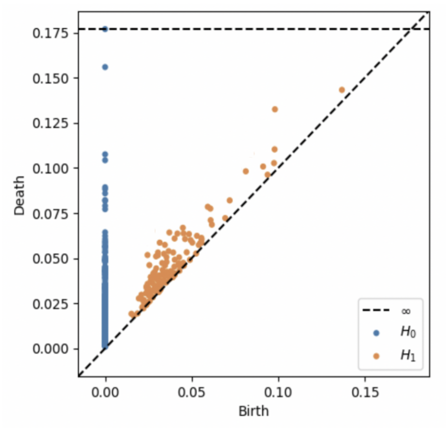


Figure 1. Corn

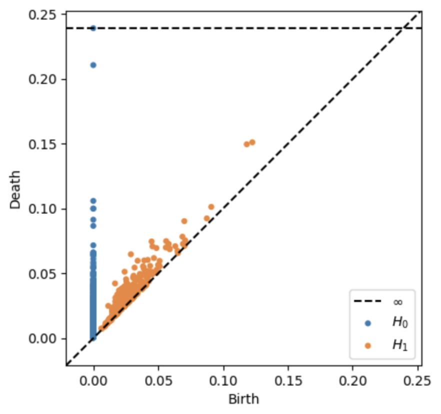


Figure 2. Gold

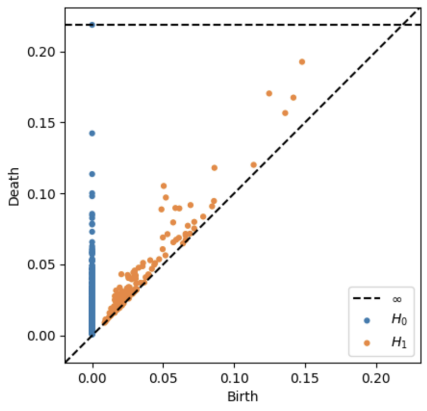


Figure 3. Oil

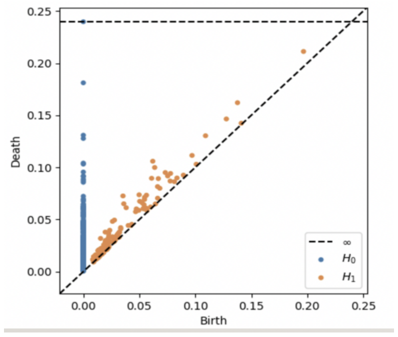


Figure 4. Dow Jones

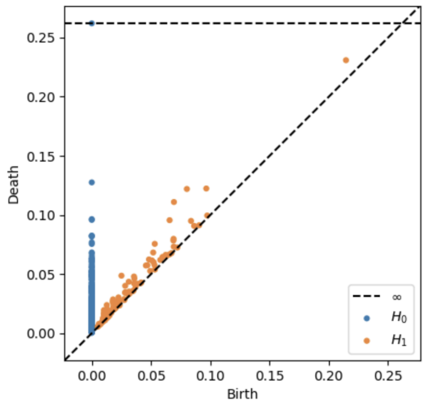


Figure 5. FTSE

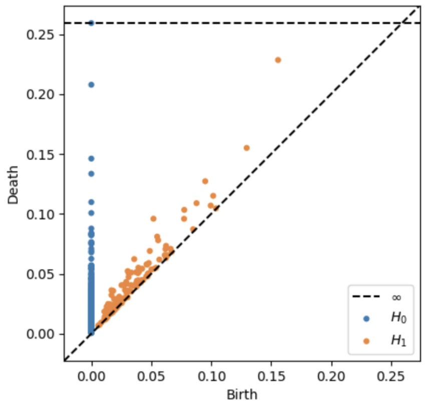


FIGURE 6. Nikkei

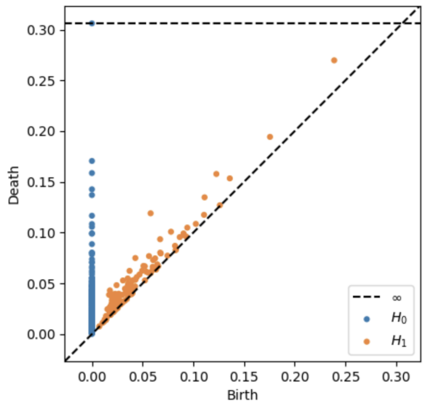


FIGURE 7. Treasury Bond

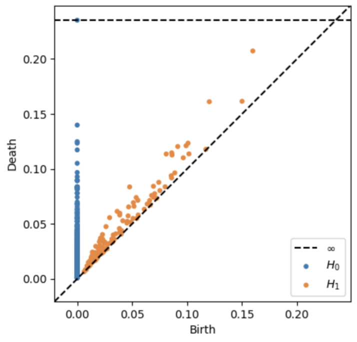


FIGURE 8. High Yield

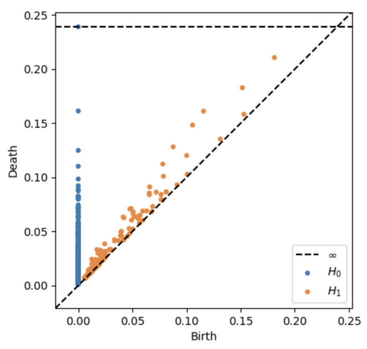


Figure 9. Private Equity

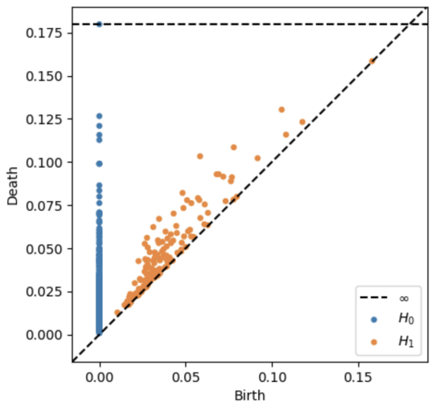


Figure 10. Real Estate

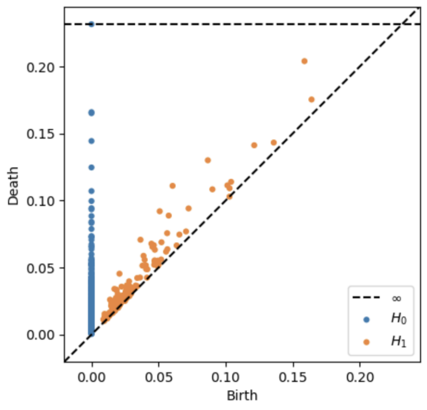


Figure 11. Bitcoin

While it was difficult to identify any relationship between volatility and persistent homology (we specifically focused on 1-dimensional homology) when viewing the persistence diagrams as a whole, we were able to notice a relationship when grouping the assets into different asset classes. For the commodities asset class (Figures 1-3), we found that oil, which had the greatest volatility out of the 3 commodities studied, had more 1-dimensional loops with higher persistence than the other 2 assets. In addition, gold, which was the least volatile asset, had the simplest topological structure, with only 1 noticeable 1-dimensional loop. For the equities asset class (Figures 4-6), this relationship largely held true. Within the equities asset class, the Nikkei persistence diagram (Figure 6) displayed a significant amount of 1-dimensional loops with high persistence, while the less volatile Dow Jones and FTSE (Figures 4 and 5) diagrams displayed a smaller amount of 1-dimensional loops with high persistence. In comparing figures 7 and 8, we also noticed that the more volatile high yield ETF's persistence diagram displayed more 1-dimensional loops with high persistence as compared to that of the less volatile Treasury bond ETF. The results we found from the alternative class were harder to interpret, as the alternatives asset class largely encompasses all assets which can't be grouped into the other 3, and so comparing between the persistent diagrams for the 3 yielded inconclusive results.

Although our sample size is limited, there does seem to be a positive correlation between volatility and degree of 1-dimensional persistent homology. It has previously been found that neural networks with better generalizing abilities to other data sets learn simpler and stronger topological structures.[2] Thus, it may be that networks trained on more volatile data with greater anomalies, such as oil or Bitcoin price data, will have more complex topological structures (as the generalizing abilities of these networks will be weaker) than networks trained on less volatile data, such as real estate or Treasury bond price data. This complex topological structure will result in the existence of more 1-dimensional persistent homology, which we have identified in our experiment.

Layer Analysis

Previous research has found that simple topological structures may be found in data sets of convolutional neural network filters as early as the first layer. To test whether this held true for convolutional neural networks trained on time-series data, we performed the aforementioned experiment and studied how the homology of neural network filter weights changes throughout the different layers of the network. We conducted this analysis for all 11 assets – for the sake of keeping this paper short, we will present results for one representative asset, Corn, and generalize our findings for the rest of the assets. We will also display results for only the first, second, and final convolutional layers. First, we display the Mapper visualization for the filters of the first convolutional layer, as well as the corresponding persistence diagram (Figure 12 and 13). We then display the Mapper visualization and corresponding persistence diagram for the second convolutional layer (Figure 14 and 15), and for the final convolutional layer (Figure 16 and 17).

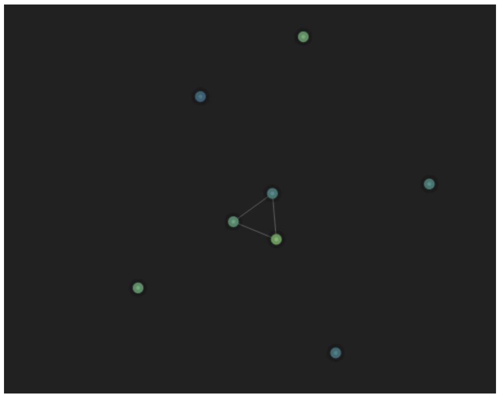


FIGURE 12. Point cloud of first convolutional layer filter weights

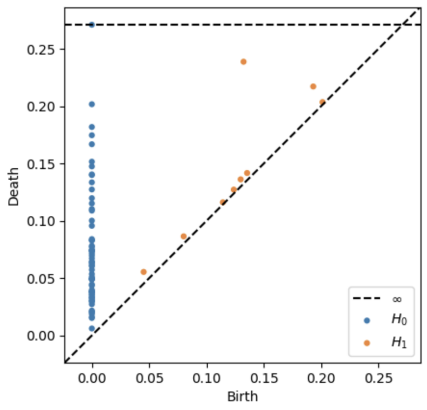


FIGURE 13. Corresponding persistence diagram of first convolutional layer filter weights

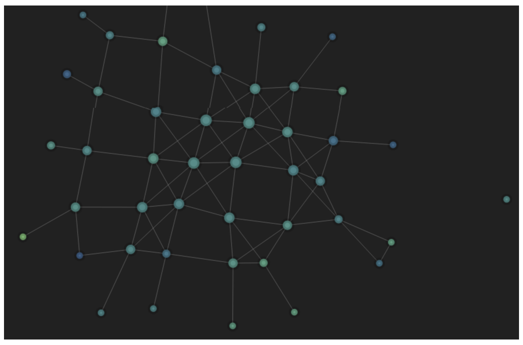


FIGURE 14. Point cloud of second convolutional layer filter weights

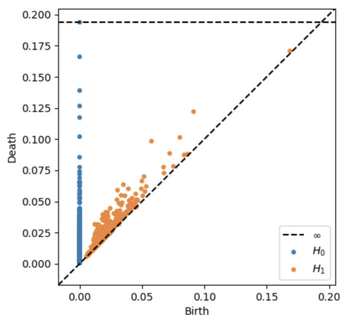


Figure 15. Corresponding persistence diagram of second convolutional layer filter weights

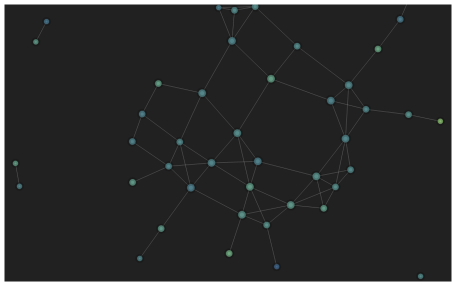


Figure 16. Point cloud of final convolutional layer filter weights

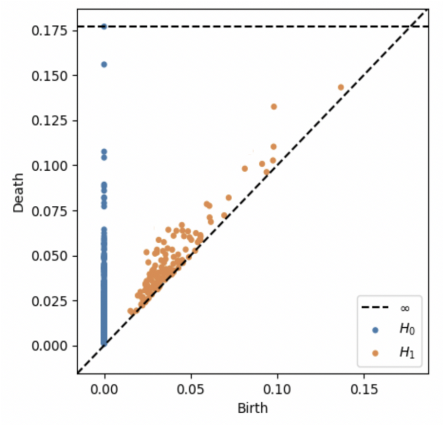


Figure 17. Corresponding persistence diagram of final convolutional layer filter weights

As expected, there is much more 1-dimensional homology as we move from layer 1 to later layers, as the number of filters in each convolutional layer must be greater than that of the previous layer. An interesting result we found, however, is that the circular feature we found in the point cloud of filters from layer 1 largely seemed to be present in many of the subsequent point clouds obtained from future layers. This is verified by the persistence diagrams for the first and final convolutional layers, which show that each point cloud has a 1-homology which has a lifetime significantly greater than that of the other 1-homologies. We found this same result for nearly all of the assets we studied. In [2], Gabriellson and Carlsson found that the primary circle was the dominant structure in most of the layers of the network model they studied, namely VGG16. Our results demonstrate the finding of a circular feature as well (although further investigation is needed to determine whether this is in fact the primary circle), indicating that the manner in which the weights are organized for different layers when CNNs are used for time-series classification purposes is likely similar to the manner in which weights are organized when CNNs are used for image recognition.

Analysis of Different Models

To gain insight into how the weights of different types of neural networks learn, we performed the aforementioned experiment on 2 additional networks – an FCN and an MLP. We present Mapper visualizations (to better understand both geometric and topological features of the point clouds) and persistence diagrams obtained from the filters of the final layer of the ResNet model, FCN, and the MLP, for 2 representative assets: corn and gold. We do this for the sake of brevity, as we found similar results for all 11 assets.

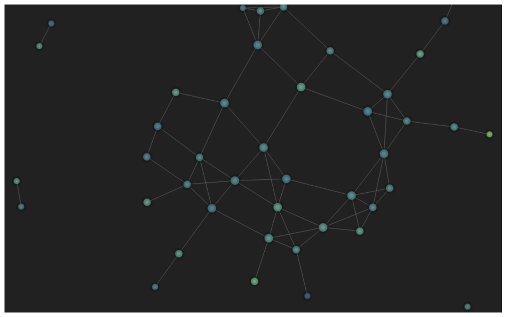


Figure 18. Mapper visualization for final convolutional layer filter weights obtained from the ResNet model for corn

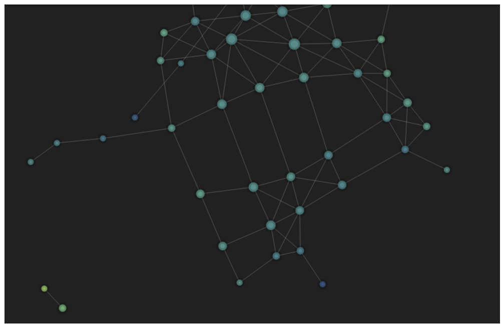


Figure 19. Mapper visualization for final convolutional layer filter weights obtained from the FCN model for corn

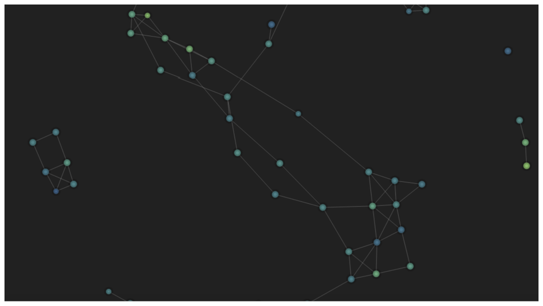


Figure 20. Mapper visualization for final convolutional layer filter weights obtained from the MLP model for corn

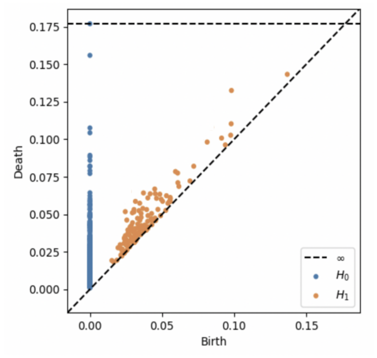


Figure 21. Corresponding persistence diagram for final convolutional layer filter weights obtained from the ResNet model for corn

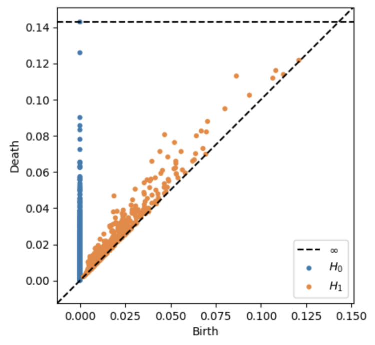


Figure 22. Corresponding persistence diagram for final convolutional layer filter weights obtained from the FCN model for corn

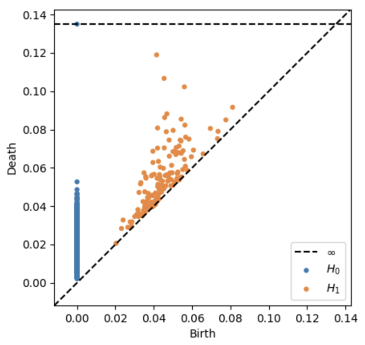


FIGURE 23. Corresponding persistence diagram for final convolutional layer filter weights obtained from the MLP model for corn

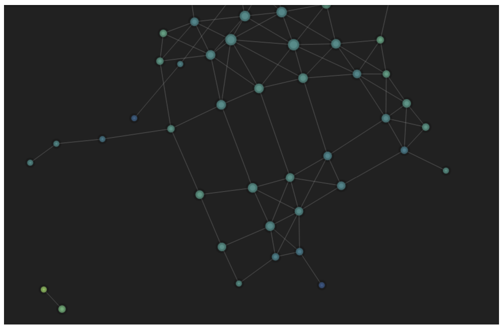


FIGURE 24. Mapper visualization for final convolutional layer filter weights obtained from the ResNet model for gold

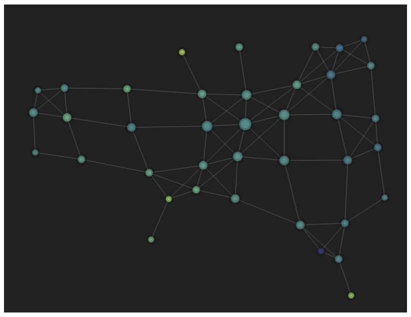


FIGURE 25. Mapper visualization for final convolutional layer filter weights obtained from the FCN model for gold

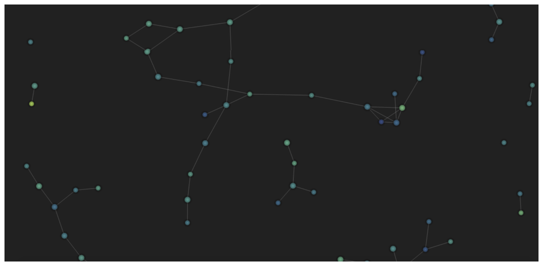


Figure 26. Mapper visualization for final convolutional layer filter weights obtained from the MLP model for gold

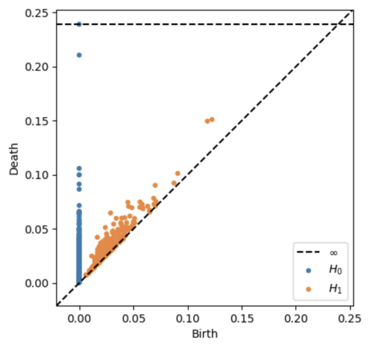


Figure 27. Corresponding persistence diagram for final convolutional layer filter weights obtained from the ResNet model for gold

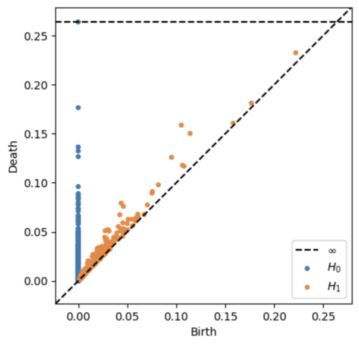


Figure 28. Corresponding persistence diagram for final convolutional layer filter weights obtained from the FCN model for gold

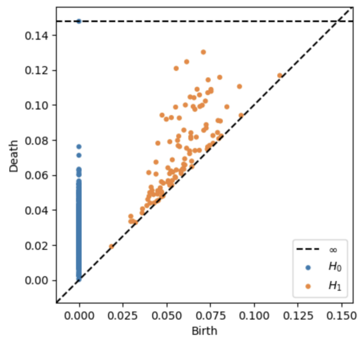


FIGURE 29. Corresponding persistence diagram for final convolutional layer filter weights obtained from the MLP model for gold

Although a cursory glance at the sets of persistence diagrams and Mapper visualizations reveals that the point clouds of filter weights obtained from the FCN and ResNet (which are both convolutional neural networks) models appear to be much more similar to one another than to the point cloud obtained from the MLP, we chose to examine this relationship further by computing the Wasserstein distance between persistence diagrams of 2 different networks, for each asset. The Wasserstein distance allows us to quantify the optimal cost of matching persistence diagrams, giving insight into differences between the topological features of different point cloud. We present the results of this analysis in Table 1, with the Wasserstein distance rounded to the nearest hundredth. We also provide, in Table 2, the maximum accuracy of each network model in predicting time-series data, as a way to evaluate the performance of each network.

	MLP vs ResNet	*ResNet vs FCN*	*FCN vs Volatility*
Treasury Bond	0.74	0.50	0.96
High Yield Bond	0.82	0.48	1.08
Private Equity	0.65	0.47	1.02
Gold	0.69	0.49	1.00
Dow Jones Industrial Average	0.55	0.56	1.02
Nikkei	0.66	0.55	1.04
Real Estate	0.73	0.51	0.86
Corn	1.03	0.59	0.91
Oil	0.50	0.47	0.83
Bitcoin	0.86	0.58	1.11
Mean Value	0.71	0.51	0.98

TABLE 2. Wasserstein distances between persistence diagrams obtained from filter weight point clouds of different networks, ordered by asset.

	FCN	*ResNet*	*MLP*
Treasury Bond	0.51	0.54	0.51
High Yield Bond	0.49	0.53	0.51
Private Equity	0.51	0.55	0.49
Gold	0.51	0.53	0.50
Dow Jones	0.53	0.50	0.52
Nikkei	0.54	0.55	0.53
Real Estate	0.60	0.61	0.58
Corn	0.51	0.57	0.52
Oil	0.56	0.58	0.51
Bitcoin	0.52	0.54	0.48
Mean Value	0.524	0.545	0.514

Table 3. Maximum prediction accuracy for each neural network model studied, ordered by asset.

[4] found that the filters in FCN and ResNet when used for time-classification tasks are very similar. It also found that the FCN and ResNet outperformed other networks, including MLP, when evaluated on classification performance. The results we display in Tables 1 and 2 and Figures 15 and 16 seem to provide support for these claims. Although we use a different measure, accuracy, to evaluate the performance of the 3 networks we studied, we also found that FCN and ResNet outperformed MLP on the basis of accuracy. In addition, the mean Wasserstein distance between persistence diagrams of point clouds obtained from ResNet and FCN was much smaller than the mean Wasserstein distances between ResNet and MLP and FCN and MLP, indicating that, as expected, the topology of the filters of these 2 networks is similar. In addition, by examining the persistence diagrams and Mapper visualizations for the 3 networks, we find that the topological structure of ResNet and FCN was much more simple than the topological structure of MLP. As previously mentioned, a simple topological structure of weights has been hypothesized to allow for better generalization ability and thus better performance, which may explain why ResNet and FCN achieve superior performance for time-series classification purposes. In addition, [3] explained that the MLP model does not preserve temporal information, causing each time-series element to be considered independently. This may explain the complexity of the topological structure of the corresponding filter point clouds.

The paper also found that ResNet had higher accuracy on different time-series datasets with varying characteristics than FCN did. This was believed to be a result of ResNet's ability to "filter-out non-distinctive regions of the time series." This ability was attributed to the fact that ResNet utilizes residual connections between convolutional blocks, which allow it to learn to skip unnecessary convolutions on account of its shortcut connections. We believe that this may be related to the fact that the weights we obtained from ResNet were topologically simpler than the weights obtained from FCN, and were more closely arranged in a circular pattern than the FCN. In essence, skipping unnecessary convolutions may result in a simpler topological structure. To test this assumption, however, we would need to perform our experiments on more assets and with a larger sample size.

Discussion

We have demonstrated that there is a noticeable correlation between the volatility of the time-series data being studied (within sub-groups where the data type – in our case, asset class, is similar) and the corresponding amount of 1-dimensional homology, which we associate with simplicity and compactness of the topological structure of the weights of ResNet. We hope in the future to potentially quantify this simplicity by finding total persistence (or lifetime) for a certain amount of 1D loops in our persistence diagrams. In addition, we would like to perform our experiment on a greater number of asset classes and assets within each class to see whether the relationship holds. In addition, we may experiment with training a CNN on a greater amount of multi-asset data (essentially, inputting several space-time pictures, instead of a single one as we did in our experiments) to "diversify away" idiosyncratic anomalies. This likely would result in greater generalization abilities, and we hypothesize that it would result in a simpler topological weight structure.

We have also found the existence of a persistent 1D loop in our point cloud of weights obtained from many of the layers of ResNet – as posited in [2], this is likely a result of the weights organizing to learn a simple global structure, which allows for greater generalization capacity. The fact that the weights organize themselves in similar structures when used for time-series classification and image classification purposes is an interesting finding, and leads us to believe that the "space-time picture" approach to classifying multivariate data sets could be very promising in classifying time-series data with CNNs.

Finally, we found that the topological structure of the weight spaces for different DNNs can vary significantly, with the greatest similarities occurring between ResNet and FCN. Although we expected this result, we would like to better understand the relationship between topological structure and network performance. ResNet and FCN, whose weights were arranged in a relatively simple and compact structure, both outperformed MLP, whose weights seemed to be organized in a complex and seemingly random fashion – we hope to run this same experiment with a greater variety of DNNs to identify what other factors may be explained by the corresponding topological structure of each DNN's weights.

While the range of experiments we can perform is vast (as each parameter is a separate experiment in itself), we believe that one of the most interesting experiments we can perform going forward would be comparing the topological structure of CNN weights trained on time-series data to the structure of the neural network architecture of the brain when subjects are asked to identify and extrapolate small-scale patterns found in time-series data to the future. Since CNN filters function by learning the essence of small patches of images, the analog to time-series data is that CNNs learn the essence of small patterns in the data. Thus, understanding whether the same process occurs in the human brain and whether the synaptic weights of dendritic spines learn similar global structures would be a goal of future research.

Acknowledgments

We thank Professor Harer for his help throughout the course of the semester.

Bibliography

[1] Malkiel, B. (2003). The Efficient Market Hypothesis and Its Critics. The Journal of Economic Perspectives, 17(1), 59-82. Retrieved from www.jstor.org/stable/3216840

[2] Gabrielsson, R. B., & Carlsson, G. (2019). Exposition and interpretation of the topology of neural networks. ArXiv:1810.03234 [Cs]. Retrieved from http://arxiv.org/abs/1810.03234

[3] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, Pierre-Alain Muller (2019). Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, Springer, 33 (4), pp.917-963. ff10.1007/s10618-019-00619-1ff. Ffhal-02365025f

[4] Wang, Z., Yan, W., & Oates, T. (2016). Time series classification from scratch with deep neural networks: A strong baseline. ArXiv:1611.06455 [Cs, Stat]. Retrieved from http://arxiv.org/abs/1611.06455

[5] Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J. L. (2016). Exploiting Multi-Channels Deep Convolutional Neural Networks for Multivariate Time Series Classification. Retrieved from https://bigdata.ustc.edu.cn/paper_pdf/2016/YiZheng-FCS2016.pdf

[6] ESANN. (2017). 25th european symposium on artificial neural networks, computational intelligence and machine learning: Esann 2017: bruges, belgium, april 26, 27, 28, 2017: proceedings (M. Verleysen, Ed.). Louvain-la-Neuve, Belgique: Ciaco.

[7] Deshpande, A. (n.d.). A beginner's guide to understanding convolutional neural networks. Retrieved December 11, 2019, from https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[8] Siripurapu, A. Convolutional Networks for Stock Trading. Retrieved from https://pdfs.semanticscholar.org/86af/cb8f6

[9] Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology, 160(1), 106–154. doi:10.1113/jphysiol.1962.sp006837

[10] Katok, A.; Hasselblatt, B. (1995). Introduction to the Modern Theory of Dynamical Systems. Cambridge: Cambridge University Press. ISBN 978-0-521-34187-5.