applied sciences

MDPI

Article

Wide-TSNet: A Novel Hybrid Approach for Bitcoin Price Movement Classification

Peter Tettey Yamak [1,*], Yujian Li [2], Ting Zhang [1] and Pius K. Gadosey [3]

[1] Faculty of Information Technology, Beijing University of Technology, Beijing 100021, China; zhangting@bjut.edu.cn
[2] School of Artificial Intelligence, Guilin University of Electronic Technology, Guilin 541004, China; liyujian@guet.edu.cn
[3] Computer Science Department, Lancaster University Ghana, Accra LA1 4YW, Ghana; p.gadosey@lancaster.edu.gh
* Correspondence: peteryamak@emails.bjut.edu.cn

Abstract: In this paper, we introduce Wide-TSNet, a novel hybrid approach for predicting Bitcoin prices using time-series data transformed into images. The method involves converting time-series data into Markov transition fields (MTFs), enhancing them using histogram equalization, and classifying them using Wide ResNets, a type of convolutional neural network (CNN). We propose a tripartite classification system to accurately represent Bitcoin price trends. In addition, we demonstrate the effectiveness of Wide-TSNet through various experiments, in which it achieves an Accuracy of approximately 94% and an F1 score of 90%. It is also shown that lightweight CNN models, such as SqueezeNet and EfficientNet, can be as effective as complex models under certain conditions. Furthermore, we investigate the efficacy of other image transformation methods, such as Gramian angular fields, in capturing the trends and volatility of Bitcoin prices and revealing patterns that are not visible in the raw data. Moreover, we assess the effect of image resolution on model performance, emphasizing the importance of this factor in image-based time-series classification. Our findings explore the intersection between finance, image processing, and deep learning, providing a robust methodology for financial time-series classification.

Keywords: time-series; Bitcoin; convolutional neural network; Wide ResNet; Markov transitional fields; histogram equalization

Citation: Yamak, P.T.; Li, Y.; Zhang, T.; Gadosey, P.K. Wide-TSNet: A Novel Hybrid Approach for Bitcoin Price Movement Classification. *Appl. Sci.* **2024**, *14*, 3797. https://doi.org/10.3390/app14093797

Academic Editors: Chihhsuan Wang and Douglas O'Shaughnessy

Received: 1 March 2024
Revised: 21 April 2024
Accepted: 26 April 2024
Published: 29 April 2024

1. Introduction

The representation of time-series data is crucial for several analytical tasks, including comparison analysis, clustering, and classification. Conventional illustration methods yield uniform representations derived from statistical data [1]. Moreover, analyzing time-series data on currency values such as bitcoin can provide insights for investment decisions. Time-series data are represented numerically and analyzed using statistical and machine-learning techniques. Numerous studies have recently focused on creating deep learning frameworks to tackle time-series problems. This trend is largely driven by advancements in deep learning technology and the evolution of graphics processing units. Within the cryptocurrency field, a considerable amount of research is being conducted on various machine-learning methods for the prediction of their prices and returns [2].

Predicting the future behavior of complex systems such as financial markets remains challenging. As the leading cryptocurrency [3], Bitcoin exhibits volatile price fluctuations, making accurate price predictions highly desirable for investors and analysts. With Bitcoin at the forefront, cryptocurrencies have transformed the financial scene, offering a decentralized alternative to the old monetary systems. However, the volatile nature of Bitcoin prices [4,5] poses a significant challenge for investors and market analysts. While technically a cryptocurrency, Bitcoin struggles to function as a traditional currency due to

its extreme price fluctuations [6]. These fluctuations make it impractical and expensive to use Bitcoin as a standard of value or a means of transaction. This is true for short-term periods (minutes) and longer durations (days, weeks, and months). However, Bitcoin can serve as a store of value, despite its volatility, when viewed over an extended timeframe. The high volatility negatively impacts Bitcoin's utility as a currency and its viability as an investment. However, even though Bitcoin has experienced significant fluctuations in its value compared to major currencies, it has the potential to develop into a reliable store of value, serving as an alternative to traditional value reserves such as gold. It is these issues related to fluctuations and high volatility that make predicting Bitcoin prices challenging.

Most of the existing literature in this context has focused on techniques such as long short-term memory (LSTM) and ARIMA [7]. While models like LSTM and BiLSTM are proficient in identifying time-related dependencies, they can encounter problems such as vanishing and exploding gradients, which can negatively impact their effectiveness [8]. A further challenge with many of these models is that their predictions span a temporal range varying from several hours to weeks, which is contingent upon the time step employed during model training. However, an imperative for accurate predictions within a more condensed time frame has emerged, which is dependent on the time step of the data under consideration. In order to leverage the superior classification capabilities of convolutional neural networks (CNNs), recent studies have converted time-series data into image format, effectively transforming the time-series classification (TSC) problem into an image classification task. This approach amplifies the unique areas of a sequence and establishes temporal correlations, leading to enhanced accuracy [9,10].

In this paper, we introduce a unique method, Wide-TSNet, for predicting Bitcoin prices using time-series data. The Wide-TSNet method employs a hybrid approach that combines image-based convolutional neural networks (CNNs) with a series of transformations and enhancements. In this approach, we first convert the Bitcoin price data into images. This time-series data approach is often used to forecast future trends and detect temporal patterns. This conversion aims to encapsulate the dynamic relationships between past price points in a visual format, potentially unveiling hidden patterns that may not be apparent in the raw data. The transformation process involves converting the time-series data into a Markov transition field (MTF) [11] image. The MTF image represents the probability of transitioning between different price states at different points in time, providing a unique perspective on the data's inherent patterns and temporal dependencies. Following the transformation, we enhance the MTF image using histogram equalization [12]. This technique improves the image's contrast, making the patterns within the data more discernible and, thus, more suitable for the subsequent classification process. Finally, we employ Wide ResNets [13], a CNN known for its performance in image classification tasks, to classify the enhanced MTF images. This classification serves as the basis for our Bitcoin price predictions.

The main contributions of this paper are as follows:

- We introduce Wide-TSNet, a novel image-based approach that leverages convolutional neural networks (CNNs) to predict Bitcoin prices. Wide-TSNet provides a powerful and accessible methodology for analyzing and forecasting complex financial data.
- We demonstrate the effectiveness of Wide-TSNet in predicting Bitcoin price movements. By integrating innovative data conversion techniques with an efficient CNN architecture, we contribute valuable insights for cryptocurrency price prediction. This will empower researchers, particularly those with limited computational resources, to further explore this challenging domain.
- We investigate the impact that varying the number of pixels in the generated images has on the classification results. This exploration provides insights into the use of computer vision approaches for time-series analysis.
- Unlike conventional approaches that dichotomize time-series data into 'increase' or 'decrease,' Wide-TSNet introduces a third class (i.e., 'stable'), providing a more nuanced representation of Bitcoin price trends. This tripartite classification forms the basis of our image-generation process.

The remainder of this article is structured as follows: Section 2 describes the related work, while Section 3 explains the Wide-TSNet architecture, including data preparation, image generation, as well as image classification. Furthermore, Section 4 gives details on the experiment setup for the research. Section 5 summarizes and discusses the results of the study, concluding the work in Section 6.

2. Related Work

The following section explores the existing literature on time-series classification. This body of work is vast and varied, encompassing a range of methodologies from time-warping [14–16] and feature-based methods [17–20] to ensemble-based methods [21,22], deep neural networks, and hybrid methods [23–25]. Each approach produces unique insights and challenges, contributing to the rich tapestry of research. Through examining these works, we aim to contextualize our research within this broader academic landscape, identifying where our work aligns with, diverges from, and builds upon these established methods. For this study, we will primarily emphasize research that employs deep learning models.

Deep neural networks (DNNs) have revolutionized the field of time-series classification. They leverage models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to automatically extract features from raw time-series data, capturing complex patterns and dependencies. In addition, the novel approach of using time-series images for classification has gained traction. This method transforms time-series data into image format, which can then be processed using image-based deep-learning techniques. The transformation process can involve various techniques, such as Gramian angular fields (GAFs), recurrence plots (RPs), and Markov transition fields (MTFs), each providing a unique visual representation of the time-series data. They can capture local features through their convolutional layers and global features with pooling layers. This combination allows CNNs to effectively handle the complexity and variability of time-series images. Using time-series images for classification combines the strengths of time-series analysis and image classification techniques. This approach enhances the feature extraction capabilities of DNNs and broadens the scope of methodologies that can be applied to time-series classification problems.

Two-dimensional (2D) GAF data together with one-dimensional (1D) time-series data have been used in a dual classification system for high-performance gas classification [26]. The gas sensor array (GSA) data set was used to evaluate the AlexNet model for 2D GAF data and an improved version of GasNet for 1D data. The modified GasNet model achieved a state-of-the-art Accuracy of 96.0% on time-series data, while AlexNet achieved an 81.3% test Accuracy in GAF classification. A GAF [27] has also been utilized to classify energy micro-moments and small contextual data points in time-series. The system, designed for edge computing efficiency, can classify up to 7 million GAF-converted data points with an approximate 90% Accuracy in less than 30 s, indicating potential for industrial adoption in edge Internet of Energy applications. Recurrence plots for the classification of time-series [28–30] have been used in various forms, all of which leverage deep neural networks for classification. The first paper proposed a novel method for time-series classification (TSC) using multi-scale signed recurrence plots (MS-RP) and fully convolutional networks (FCN). This method effectively handles the variability in the distinctive region scale, the length of sequences, and the tendency confusion problem. The second paper addressed the limitations of RP and TSC networks in terms of handling the scale and length variability of sequences by proposing a new method, MSRP-IFCN, which includes a multi-scale signed RP (MSRP) and the inception fully convolutional network (IFCN). The MSRP enhances the scale of images and represents long sequences, while the IFCN improves multi-scale feature extraction. The method had a superior performance on 85 UCR data sets, indicating its effectiveness. In the third paper, the authors transformed the time-series into a black-and-white RP image. A convolutional neural network was then used to classify the image. Xiaoting et al. [31] have proposed an innovative approach that leverages Markov transition fields (MTFs) and deep learning to classify vibration events and measure the vibration frequency in a φ-OTDR-based fiber-optic distributed vibration sensor. The method transforms normalized time-series data from a detected signal into an

MTF image, which is subsequently classified using a convolutional neural network (CNN) and a fully connected neural network. Notably, this cost-effective and efficient method outperformed conventional techniques, effectively handling both vibration events and single-frequency vibrations. This study provided a technical reference for applying deep learning to measure the vibration frequency and offers insights into the event classification of vibrations using image processing methods.

Deep learning models have found extensive application in handling financial time series datasets. Convolutional neural networks and recurrent neural networks stand out as the most prevalent deep neural network architectures employed for this purpose. Researchers have leveraged these models in various forms to address forecasting and classification tasks related to financial time series data. Their utilization has yielded valuable insights and contributed significantly to the field.

In response to the challenges posed by mode mixing in high-frequency financial time series data, a novel classification method based on low-frequency approximate representation is proposed [32]. The approach leverages complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN) to decompose time series into modal components and residual terms. By calculating permutation entropy and clustering it into two categories, the method integrates corresponding modal components and residual terms to extract low-frequency information. This adaptive extraction mitigates mode mixing issues, leading to improved classification. Distance matrices (computed using Euclidean distance or dynamic time warping) are then used for classification.

To address the speculative nature of Bitcoin, which complicates price forecasting, this research [33] proposes a robust forecasting framework that mitigates noise in Bitcoin time series and evaluates the predictive capabilities of three distinct types of predictors: fundamental indicators, technical indicators, and univariate lagged prices. The framework encompasses a three-step hybrid feature selection process to identify the most predictive variables, followed by the application of Hampel and Savitzky–Golay filters to remedy outliers and eliminate signal noise from the Bitcoin time series. Subsequently, multiple deep neural networks, fine-tuned using Bayesian optimization, are employed to forecast short-term prices for intervals ranging from the next day to seven days ahead. Among the benchmark models evaluated, the Deep Artificial Neural Network model, utilizing technical indicators as input data, outperforms alternative models. The presented findings demonstrate a notable level of accuracy, surpassing all existing models reported in prior literature.

This study [34] examines the impact of various augmentation methods on stock datasets using two state-of-the-art deep learning models. The findings demonstrate that the incorporation of augmentation methods leads to significant enhancements in financial performance when combined with a trading strategy. In the case of both relatively small and large datasets, the augmentation methods yielded a remarkable improvement in risk-adjusted return performance. These outcomes highlight the efficacy of augmentation methods in addressing the challenges associated with financial data and underscore their potential for enhancing predictive models in this domain.

This paper [35] introduces a novel tri-state labeling approach to classify the underlying patterns in price data, categorizing them as up, down, or no-action. The inclusion of a no-action state in this approach reduces the need for denoising the dataset as a preprocessing step. Additionally, the framework incorporates Bayesian optimization to select the optimal tuning values for hyperparameters. The price trend prediction module generates trading signals necessary for decision-making. The results demonstrate that the framework achieves an average annualized Sharpe ratio of approximately 2.823, indicating excellent cumulative returns and highlighting the effectiveness of the approach.

3. Wide-TSNet

In this chapter, we delve into the details of our proposed method, Wide-TSNet, a novel hybrid approach for time-series classification. Wide-TSNet was designed to predict Bitcoin prices from time-series data, leveraging the power of image-based convolutional neural

networks (CNNs) and a series of transformations and enhancements. We begin by discussing the data processing layer. Thereafter, we describe the image generation layer, including the process of converting time-series data into a Markov transition field (MTF) image. This unique transformation visually encapsulates the dynamic relationships between past price points.

The enhancement layer follows the image generation layer, which plays a crucial role in improving the quality of the input data. This layer employs histogram equalization (HE), a powerful image processing technique that enhances the contrast, thus making the patterns in the data more discernible. This is achieved by redistributing the pixel intensities of the image, such that they are uniformly distributed across the entire available intensity range. This uniform distribution of intensities can significantly improve the image's visual quality, revealing details that may have been obscured in the original data.

In our model, this enhancement layer pre-processes the input data, preparing it for the subsequent layers of the network. By improving the contrast and enhancing the details of the input data, the enhancement layer aids in extracting meaningful features, which is vital for the model's performance.

Following the enhancement layer, we have the classification layer. This layer utilizes the Wide ResNet framework—a CNN variant known for its performance in image classification tasks. Wide ResNet was designed to utilize a wide range of complex patterns and structures within the data, making it an excellent choice for our model.

The classification layer analyzes the enhanced data, identifies the key features, and classifies the data based on these features. Wide ResNet's strength lies in its ability to handle high-dimensional data and extract complex patterns, which is crucial for financial time-series prediction tasks.

The Wide-TSNet framework diagram shown in Figure 1 visually represents these layers and their sequence in the model. This diagram serves as a roadmap, guiding the flow of data through the various layers of the network from the initial input to the final output.

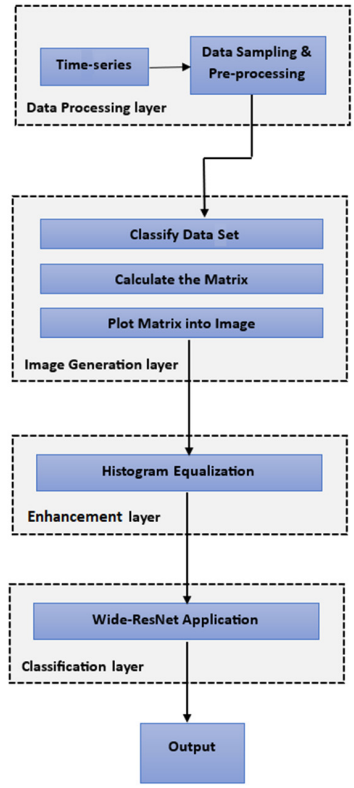


Figure 1. Wide-TSNet framework diagram.

3.1. Data Processing Layer

Data processing begins with the Bitcoin price data being loaded from a CSV file. The data are read into a Pandas DataFrame, in which the 'date' column is converted into the DateTime format and set as the index of the DataFrame. This step ensures that the data are chronologically ordered, which is essential for time-series analysis. The number of samples and timestamps for the training set is then defined. The number of samples refers to the total number of data points to be used for training, while the number of timestamps refers to the length of each time-series data point. In this case, we used 2554 samples, each containing 50 timestamps.

We also define the number of valid days and samples for our validation set. The valid days represent the number of days to be used for validation, and the valid samples represent the number of data points to be used for validation. We calculate the number of valid samples as one-fifth of the total number of samples and generate the start indices for our training and validation sets. These indices represent the starting points of our time-series data points within the original data frame. For the training set, the start indices are randomly generated in the range between 0 and the length of the DataFrame minus the number of timestamps and valid days. For the validation set, the start indices are randomly generated in the range between the length of the DataFrame minus the number of timestamps and valid days and the length of the DataFrame minus the number of timestamps.

We then create our training and validation sets with the generated start indices. For each start index, we extract a time-series data point from the 'close' column of the DataFrame, starting from the start index and ending at the start index plus the number of timestamps. These data points are then appended to the respective training or validation set.

Finally, we convert our training and validation sets into NumPy arrays. This step is necessary as Wide-TSNet, like most machine learning models, requires the input data to be numerical.

3.2. Image Generation Layer

In this layer, the processed data are passed through further operations, such as data set classification, Markov transition field (MTF) calculation, and image plotting. These steps are crucial for converting the processed data into a format that can be analyzed visually or passed through additional processing stages.

3.2.1. Data Classification

In our strategy, a function is defined to calculate the percentage change in price between two points in time for each day. This function categorizes the price change into 'INCREASE', 'DECREASE', or 'STABLE'. After extensive data exploration and analysis concerning these categories and the thresholds defining them, we determined the following:

- If the price change exceeds 5%, it is categorized as an 'INCREASE';
- If the price change is less than -5%, it is categorized as a 'DECREASE';
- Otherwise, it is categorized as 'STABLE'.

3.2.2. Markov Transition Field

The Markov transition field (MTF) [11] is defined as a Matrix $\boldsymbol{M}$.

$$M = \begin{bmatrix} w_{ij|x_1 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_1 \in q_i, x_n \in q_j} \\ \vdots & \ddots & \vdots \\ w_{ij|x_n \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_n \in q_i, x_n \in q_j} \end{bmatrix}. \quad (1)$$

A Q $\times$ Q Markov transition matrix (referred to as W) is constructed by dividing the data (magnitude) into Q quantile bins. These quantile bins correspond to the data

at timestamps i and j along the temporal axis, denoted as $\mathbf{q}_i$ and $\mathbf{q}_j$ (where $q \in [1, Q]$). The element M_{ij} in the Markov transition field (MTF) represents the transition probability from q_i to q_j. Essentially, we extend the transition probability matrix W—which captures transitions based on magnitude—into the MTF matrix by incorporating temporal positions. At each pixel, the probability of transitioning from the quantile at timestep i to the quantile at timestep j is assigned to M_{ij}. Consequently, the MTF matrix M effectively encodes multi-span transition probabilities.

Figure 2 visually represents the Markov transition fields (MTFs) at two sample times, presented in grayscale and RGB formats. Grayscale is a type of image in which the value of each pixel is a single sample, representing an amount of light only; therefore, only concentration information is reflected. Grayscale images are composed solely of shades of gray, with the contrast ranging from black at the weakest intensity to white at the strongest.

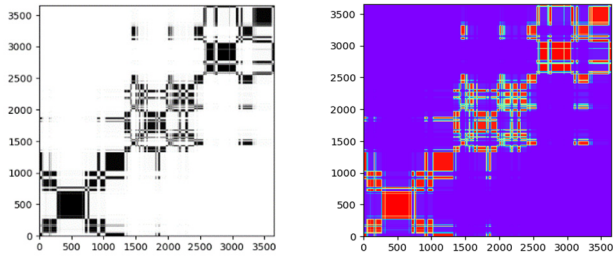


Figure 2. Sample Markov transition fields of the Bitcoin time-series data in grayscale and RGB.

On the other hand, RGB (red, green, blue) is the color model used in digital screens worldwide. Color in the RGB model is defined using three integer values ranging from 0 to 255 for red, green, and blue, where a value of 0 denotes dark and a value of 255 denotes bright [36]. Using both the grayscale and RGB formats in the representation of MTFs allows for a comprehensive understanding of the time-series data, as each format provides a unique perspective on the data's inherent patterns and temporal dependencies.

3.2.3. Plot Image

In the final phase of the image generation layer, the processed data are transformed into Markov transition fields (MTFs), which are subsequently plotted and stored as images. The decision to employ the MTF for image plotting was not arbitrary. We systematically explored alternative image generation methods, yet MTF consistently yielded superior results compared to other models. Consequently, we opted for MTF as our chosen approach. Additional findings from alternative image generation methods are documented in the results and discussion section.

This phase is crucial, as it translates the numerical data into a visual format that can be further analyzed using image-based machine learning techniques. Once the MTFs are calculated, they are iteratively plotted and stored as images. For each time-series data point in the training and validation set, the direction of movement (increase, decrease, or stable) is determined based on the closing prices at two consecutive timestamps. This direction forms part of the filename for the corresponding MTF image, providing a straightforward way to identify each image class.

The MTF image is then plotted using a specified colormap and origin, with the axes turned off to ensure that only the MTF image is visible. The plotted images are in Portable Network Graphic (PNG) format. The image is saved to a specified directory with a filename that includes the direction of movement and the index of the data point. The image is saved with transparency and a tight bounding box to ensure that only the MTF image is stored (i.e., without additional whitespace or annotations). Finally, the plotted image is closed to free up memory for subsequent iterations. This process is repeated for each time-series data point in the training set, resulting in a collection of MTF images for further analysis.

3.3. Enhancement Layer

The enhancement layer focuses on improving the quality of the generated images. One common technique employed at this stage is histogram equalization. This method improves image contrast by adjusting the intensity values in an image such that the resulting histogram closely aligns with a pre-defined target histogram. This step is crucial as it makes the patterns within the data more discernible and, thus, more suitable for the subsequent classification process.

Image enhancement is a crucial aspect in the realm of low-level image processing. Its primary objective is to augment the quality of images that exhibit low contrast. In other words, it amplifies the intensity disparity between the objects and the background. Numerous methodologies [37,38] have been devised for this purpose, which can be broadly categorized into local and global methods [39]. Histogram stretching is generally considered a global method as it operates on the entire image simultaneously, stretching the range of pixel intensity values across the whole image to span a desired range.

On the other hand, histogram equalization can be either local or global, depending on its application. When applied to the entire image, it is a global method; that is, it redistributes the pixel intensity values in the image such that they follow a uniform distribution. However, a variant called adaptive histogram equalization (or contrast-limited adaptive histogram equalization) [40] operates on small regions in the image, making it a local method. This can be more effective at enhancing contrast in images with varying lighting conditions. In our study, however, we utilized the histogram equalization method.

3.3.1. Histogram Equalization

Histogram equalization (HE) [41] is used in image processing to enhance an image's contrast and dynamic range. It redistributes the pixel intensities so that they are more uniformly distributed across the available range. This results in an image with improved visual quality and enhanced details, making it particularly useful for various computer vision and image analysis applications. Histogram stretching maintains the original shape of the histogram while enabling interactive enhancement. In contrast, histogram equalization alters the histogram's shape and does not support interactive image enhancement; instead, it produces a single result. The equalization mapping function, which maps the original intensity levels to a new one, is given by the following:

$$E(k) = \frac{C(k) - C_{min}}{N - 1} \times (L - 1), \tag{2}$$

where L stands for the number of possible intensity levels. The function scales the cumulative distribution function (CDF) values to cover the full intensity range (from 0 to L $-$ 1), and C_{min} is the minimum value of the CDF. The following formula is used to compute the CDF:

$$C(k) = \sum_{j=0}^{k} \frac{H(j)}{N}. \tag{3}$$

In particular, the CDF is computed by summing up the relative frequencies of intensity levels from 0 to k. N represents the total number of pixels in the image. When performing histogram equalization on a color image, the color intensity is uniformly adjusted while keeping the color unchanged. Figure 3 compares the original image to the histogram equalized image. The equalized image is created by applying the equalization mapping function to the original image:

$$I_{equalized}(x, y) = E(I(x, y)). \tag{4}$$

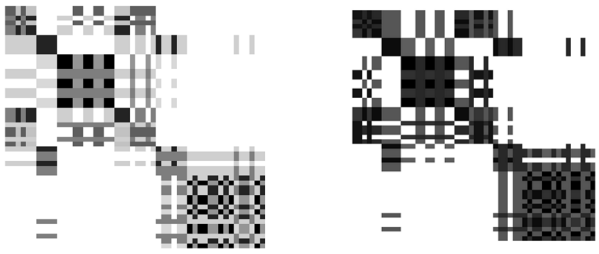


Figure 3. Comparison of original and histogram equalized image.

3.4. Classification Layer

The final layer of the Wide-TSNet framework is the classification layer. In this layer, Wide ResNets are employed to classify images (or patterns within them) based on features learned from previous layers. Wide ResNets are advanced machine learning algorithms known for their accuracy and efficiency in handling complex data sets. The output of this layer is the classification result, which provides the final Bitcoin price prediction.

Convolutional neural networks (CNNs) [42,43] revolutionized computer vision by automatically learning spatial hierarchies of features in tasks such as image and video classification. These networks consist of one or more convolutional layers, often followed by pooling, fully connected, and normalization layers. The convolutional layer performs a dot product between its weights and a small region in the input volume, extracting feature maps from the image. Subsequent pooling layers reduce the spatial size to control overfitting. The architecture of a simple CNN model is depicted in Figure 4.

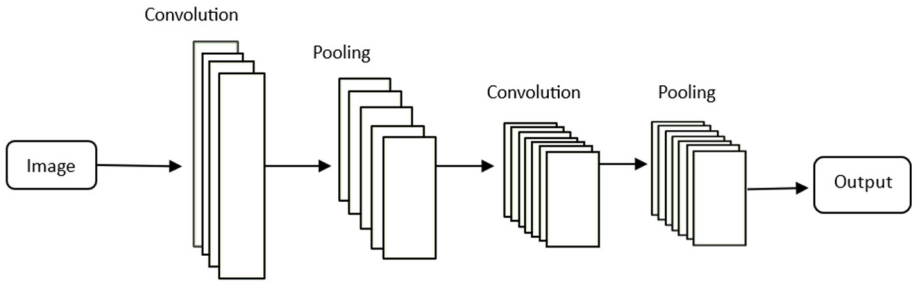


Figure 4. A simple convolutional neural network framework.

The CNN architecture is specifically tailored to exploit the inherent 2D structure of input images (or other 2D data, such as speech signals). This is accomplished through the use of local connections and tied weights, followed by a pooling process that yields translation-invariant features. An additional advantage of CNNs is their ease of training and reduced parameter count, compared to fully connected networks with an equivalent number of hidden units. Several CNN variants have been introduced, and in the next section, we outline those selected for this study.

Wide ResNets

A residual network (ResNet) is a type of artificial neural network. The key innovation of a ResNet is the introduction of "Skip connections" or "Shortcut connections," which allow the network to skip one or more layers. This approach helps to address the vanishing gradient problem, a common issue in training deep neural networks. ResNet has several variants, each with a different number of layers, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, among others. Other studies have further developed ResNet variants, such as ResNeXt, ResNeSt, Res2Net, and Res2NeXt5 [44–46]. As this study focuses on utilizing lightweight CNN models, we focused on ResNet-18. We also used Wide ResNets for comparison with lightweight models.

Wide residual networks (Wide ResNets) [13] are a ResNet variant with decreased depth and width. This novel architecture was proposed to tackle the problem of diminishing feature reuse in very deep residual networks, which makes these networks very slow to train. Wide ResNets also have several variants with different numbers of layers [47]. The naming convention for Wide ResNets is WRN-n-k, where 'n' refers to the total number of convolutional layers and 'k' refers to the widening factor. Wide ResNets have several advantages over standard ResNets, such as having fewer layers and being faster to train. For example, a simple 16-layer-deep Wide ResNets outperformed all previous deep residual networks in terms of accuracy and efficiency, including 1000-layer deep networks. Moreover, they achieved state-of-the-art CIFAR, SVHN, and COCO results, as well as significant improvements in the ImageNet data set.

4. Experimental Setup

4.1. Data

We chose the Bitcoin data set, as it is distinct from other cryptocurrency data sets. Bitcoin was created in 2008 and was the first cryptocurrency. It boasts the largest market share and has a large data set. Moreover, its high frequency and volatility [48] made it an ideal fit for our research.

The data used in this research were Bitcoin [49] data from 13 July 2013 to 22 October 2023. We selected this timeframe to address the issue of data set imbalance after splitting the data into different classes. Prior to 2013 (and even a few timesteps after), closing prices remained relatively stable. Consequently, we ended up with more STABLE images compared to INCREASE and DECREASE images. Figure 5 shows a plot of the time-series data.

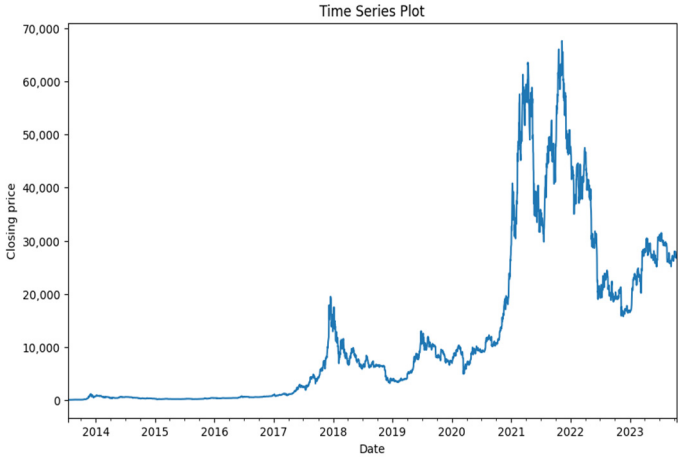


Figure 5. Plot of the closing price from the Bitcoin data set.

The data set contains 3745 data rows, each representing a specific time point in the data. We opted for a 70:30 ratio for training and testing. This resulted in 2554 days for training and 1095 days for testing. Our study utilized these historical Bitcoin price data, which were loaded from a CSV file. The 'date' column was converted into the DateTime format and set as the index of the DataFrame.

After generating and categorizing the images for each time-series data point (either 'INCREASE', 'DECREASE', or 'STABLE'), it was observed that the image data set was highly imbalanced, favoring the 'STABLE' category. A class weight was introduced in the classification layer to address this imbalance. This adjustment compelled the neural network to focus more on the under-represented categories.

4.2. Experimental Platform

The results were generated using a Python 3.10 Jupyter Notebook executed on Google Colab. The system employed an NVIDIA Intel® A100 GPU with 40 GB of HBM2e RAM,

from California, United States. Despite these resources, when the batch size was limited to 8, both the SqueezeNet and EfficientNet models could run the generated images on Google Colab on a Tesla T4 GPU, which has 16 GB of GDDR6 memory and 320 Tensor Cores.

4.3. Experimental Settings

We generated the images using the 'pyts' [50] open-source Python package. The MTF images were created using the 'MarkovTransitionField' function with an image size of 50. This parameter determines the resolution of the output image. We then transformed the training and testing data into images using the 'fit_transform' function. During this process, the strategy used to categorize the images was applied. To plot the image, we set the colormap ('cmap') to 'Greys' to provide a grayscale data representation; alternatively, it could be set to 'Rainbow' to generate an RGB representation of the data.

We then applied histogram equalization to the generated images. For this purpose, we used the 'cv2.equalizedHist' functions from the 'cv2' package. This function redistributes the pixel intensities of the image such that they become more uniformly distributed across the entire available intensity range, as described in Section 3.3.1. Finally, we saved the processed images to a specified output directory using the 'cv2.imwrite' function.

Finally, to enable CNN to classify the generated images, we used the FastAI library [51], a high-level library built on top of PyTorch. First, we created DataLoaders from the DataFrame using the FastAI library. The DataLoaders handle the loading and pre-processing of the images, including cropping the images to a uniform size, normalizing the pixel intensities using ImageNet statistics, and splitting the data into training and validation sets. The seed for the random split was set to 42 to ensure reproducibility. To address the class imbalance in the data set, we calculated class weights based on the inverse of class frequencies. These weights were then used when defining the 'CrossEntropyLoss' loss function used during the model's training. This approach ensures that the model pays more attention to under-represented classes.

The pre-trained Wide ResNet-50_2 was then employed for training on the images. Table 1 shows the used hyperparameters. The final layer of the model was replaced with a linear layer that matched the number of classes in our data set. The model was trained using the one-cycle policy for four epochs, after which the entire model was unfrozen to allow for fine-tuning. The learning rate was found using the learning rate finder, and the model was fine-tuned for an additional 40 epochs.

Table 1. Hyperparameters of Wide ResNet.

Parameter Name	Parameter Value
Model	Wide ResNet-50_2
Pretrained	True
Batch Size	32
Weight Decay	0.0001
Learning Rate	0.000132
Epochs	40

Finally, the performance of the model was evaluated using the Accuracy and F1 score. These metrics provide a comprehensive view of the model's performance, taking into account both the positive and negative classes. The Accuracy and F1 score were calculated using the model's predictions on the validation set and were printed for reference.

5. Results and Discussion

5.1. Evaluation Metrics

In this section, the quantitative and qualitative results and the evaluation metrics employed are presented and discussed. We chose the following as evaluation metrics:

The Accuracy is a commonly used metric in machine learning that measures the overall correctness of the model. It is the ratio of the number of correct predictions to the

total number of predictions, where the i^{th} prediction is the same as the actual value and N is the total number of predictions. The Accuracy is calculated using the following formula:

$$Accuracy = \frac{\sum_{i=1}^{n} Correct_i}{N}. \quad (5)$$

The F1 score is another important metric that considers both Precision (the proportion of true positive predictions overall positive predictions) and Recall (the proportion of true positive predictions overall actual positives). The F1 score is the harmonic mean of Precision and Recall, thus balancing the trade-off between these two metrics. It is particularly useful in scenarios where both false positives and false negatives are crucial. The F1 score is calculated as:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}, \quad (6)$$

where precision and recall are calculated as:

$$Precision = \frac{TP}{TP + FP}, \quad (7)$$

$$Recall = \frac{TP}{TP + FN}, \quad (8)$$

where;

- True Positive (TP) indicates a correct prediction;
- False Positive (FP) indicates an incorrect prediction (i.e., a negative instance is incorrectly classified as positive);
- True Negative (TN) indicates the correctly predicted negative instances;
- False Negative (FN) indicates a missed prediction (i.e., a positive instance is incorrectly classified as negative).

5.2. Experimental Results

5.2.1. Different Image Generation and Classifier Techniques

During the experimental phase, we conducted various tests to validate the effectiveness of Wide-TSNet in predicting Bitcoin time-series data. One of the key steps involved generating images with Gramian angular fields. Specifically, we experimented with two types of Gramian angular fields [11]: the Gramian angular summation field (GASF) and the Gramian angular difference field (GADF).

As part of the experiment, we also utilized two lightweight convolutional neural network (CNN) models: SqueezeNet [52] and EfficientNet [53]. These models were chosen for their efficiency and performance in image classification tasks. In addition to SqueezeNet and EfficientNet, we experimented with two other lightweight models: MobileNet and ResNet18. However, the results from these models are not included in our discussion. Despite training these models for several epochs, we observed that the validation loss was consistently lower than the training loss. This is typically indicative of underfitting—a scenario in which the model fails to capture the underlying pattern of the data. Therefore, we decided to focus on the models that demonstrated more promising results in our experimental setup.

Tables 1 and 2 present a comprehensive comparison of the performance metrics (i.e., the Accuracy and F1 score) of the three distinct convolutional neural network (CNN) models—Wide ResNets, SqueezeNet, and EfficientNet—when applied to the images derived from Bitcoin time-series data using the different image generation approaches. The first table focuses on RGB images, while the second table is dedicated to grayscale images.

Table 2 provides a comparative analysis of the performance of the Wide-TSNet model against other models enhanced with the Gramian angular summation field (GASF) and Markov transition field (MTF). The performance was evaluated based on the key metrics: Accuracy, F1 score, Precision, and Recall.

Table 2. The experimental results as compared to Wide-TSNet.

Model	Accuracy	F1 Score	Precision	Recall
Wide-TSNet	0.939614	0.900686	0.901232	0.900054
Wide ResNet + GASF	0.888235	0.776290	0.775972	0.776608
Wide ResNet + GADF	0.890196	0.778253	0.768930	0.787805
Wide ResNet + MTF	0.874510	0.710430	0.711149	0.709712
SqueezeNet + GASF	0.848403	0.735559	0.732773	0.738366
SqueezeNet + GADF	0.821536	0.737268	0.734545	0.740011
SqueezeNet + MTF	0.803628	0.639979	0.640627	0.639332
EfficientNet + GASF	0.875887	0.764666	0.751254	0.778566
EfficientNet + GADF	0.875688	0.765998	0.765684	0.766312
EfficientNet + MTF	0.860341	0.698523	0.699230	0.697817

The Wide-TSNet model outperformed all other models, achieving an Accuracy of approximately 0.939614, Precision at 0.901232, Recall at 0.900054, and an F1 score of approximately 0.900686. These metrics indicate a high level of Precision and Recall, suggesting that the Wide-TSNet model is highly effective in accurately classifying or predicting outcomes while minimizing false positives and negatives. The combinations of Wide ResNet with GASF and GADF also exhibited commendable performance, with Accuracies above 0.87 and F1 scores exceeding 0.71. This underscores the effectiveness of integrating these enhancement techniques with established models to obtain improved performance metrics.

The results obtained with the SqueezeNet and EfficientNet combinations demonstrated that, while they are efficient, there was a noticeable drop in both the Accuracy and F1 scores when compared with the Wide ResNet combinations. This could be attributed to the architectural differences between these models. These results indicate the superiority of Wide-TSNet, in terms of Accuracy, F1 score, Precision, and Recall; however, it also highlights the potential for improving the model's performance through integrating enhancement techniques such as GASF or MTF. This offers insights for future research directions, with the aim of optimizing machine learning models for increased accuracy and efficiency.

In addition to the performance metrics, the computational efficiency of a model also plays a crucial role in its practical applicability. Table 3 shows the computational and time performance of the CNN models employed. While demonstrating superior Accuracy and F1 score, the models that employed Wide ResNet—including Wide-TSNet—required a minimum of 15.36894 gigabytes (GiB) of memory for a batch size of 32 and 13 min and 36 s for a 40-epoch run. This indicates a higher computational demand when compared to the other models.

Table 3. Computational and time performance of CNN models.

Model	Memory (GiB)	Training Time (40 Epochs)
Wide ResNet	15.36894	13 min 36 s
SqueezeNet	13.10000	10 min
EfficientNet	14.92400	10 min 48 s

The SqueezeNet models, on the other hand, required a minimum memory of 13.1 GiB and took an average of 10 min for a 40-epoch run. Despite their lower Accuracy and F1 score, this reduced computational requirement makes them a more efficient choice in scenarios where computational resources are limited.

Finally, the EfficientNet models required a minimum of 14.924 GiB and took an average of 10 min and 48 s for a 40-epoch run. While they exhibited a slightly higher computational requirement than SqueezeNet, they still balanced performance and efficiency.

5.2.2. Grayscale vs. RGB

In the dynamic landscape of time-series image classification, the choice between RGB (red, green, blue) and grayscale color spaces significantly influences model performance and interpretability. In this section, we outline the experiment aimed at scrutinizing the impact of color representation on classification outcomes. The results were compared for the Wide-TSNet model, as shown in Table 4.

Table 4. Optimization test of the Wide-TSNet model.

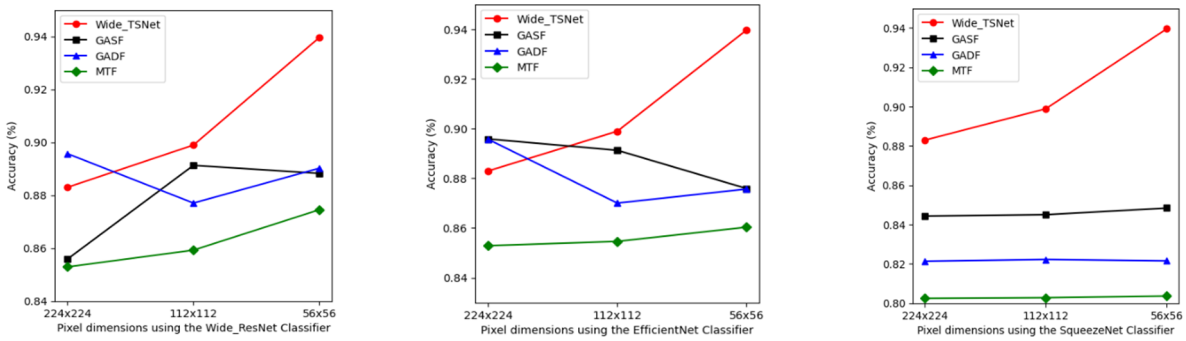
Model	Accuracy	F1 Score	Precision	Recall
Wide-TSNet	0.939614	0.900686	0.901232	0.900054
Wide ResNet + MTF	0.874510	0.710430	0.711149	0.709712
Wide ResNet + MTF (RGB)	0.862267	0.707301	0.708016	0.706587

Wide ResNet + MTF: This variant, in which the generated MTF image is not enhanced by histogram equalization, exhibited declines in both performance metrics, with an Accuracy of approximately 0.874510 and an F1 score of 0.710430. This suggests that the lack of image enhancement negatively impacts the model's performance.

Wide ResNet + MTF (RGB): This variant, in which the MTF image is in RGB, exhibited slightly lower metrics than the non-RGB version, with an Accuracy of approximately 0.862267 and an F1 score of 0.707301. This suggests that converting the MTF image to RGB does not necessarily improve the model's classification abilities or the balance between Precision and Recall.

5.2.3. Pixel Dimension Analysis

The image resolution of the generated images plays a pivotal role in the performance of these models. To assess the impact of different resolution settings on the final classification performance of Wide-TSNet, we examined how the image size affected the Accuracy. We compared Wide-TSNet with the various classifiers and image generation methods, as depicted in Figure 6. We chose different pixel dimensions (224 × 224, 112 × 112, and 56 × 56), and used the resulting images in the selected classification models.



(**a**) (**b**) (**c**)

Figure 6. Effect of image size on the classification accuracy of the various classifiers, as compared to Wide-TSNet: (**a**) Wide ResNet; (**b**) EfficientNet; and (**c**) SqueezeNet.

It appears that the Wide-TSNet model performed best with an image size of 56 × 56. It also provided better performance than the other models at most resolutions, which could be attributed to the model's ability to extract more features from higher-resolution images. This provides a richer data set for the model to analyze and interpret, improving its performance. The GASF image generation method demonstrated consistent performance across varying resolutions. This robustness is a significant advantage in applications where the image resolution varies. Additionally, the MTF model showed an improvement with increasing pixel dimensions, suggesting that this model performs better at higher

resolutions. These insights could be instrumental in tailoring pre-processing strategies to enhance classification efficacy while mitigating computational overheads.

On the other hand, the EfficientNet and SqueezeNet models exhibited an inverse relationship between image size and classification accuracy, suggesting that these models may experience overfitting at higher resolutions. Overfitting occurs when a model captures noise or irrelevant features in the data, which can compromise its performance. This highlights the need for a balanced approach that considers the model's specific architecture and application domain when choosing the image resolution.

5.3. Discussion

Our findings reveal that Wide-TSNet achieved an impressive accuracy of 0.94 on the Bitcoin dataset. The result significantly outperforms other models, as shown in Table 5. For the sake of uniformity, we selected research studies that utilized the same dataset and evaluation metric. This approach ensures a fair comparison of the performance across various models. However, relying solely on a handful of evaluation metrics limits the models.

Table 5. A comparison of related research results.

Paper	Best Method	Data Set	Accuracy
Wide-TSNet	Wide-TSNet	Bitcoin	0.94
Sin et al. [54]	MLT + ANN	Bitcoin	0.63
Tanwar et al. [55]	LSTM + GRU	Bitcoin	0.55
McNally et al. [56]	LSTM	Bitcoin	0.53
Critien et al. [57]	RF + MLP	Bitcoin	0.55
Hitam et al. [58]	Support Vector Machine (SVM)	Bitcoin	0.90

The substantial accuracy gain demonstrates the effectiveness of our approach in capturing intricate patterns with the volatile Bitcoin price data. This allows for deep feature extraction, enabling accurate predictions.

However, it is essential to recognize that superior accuracy comes at a computational cost. Researchers must carefully weigh this computational demand against the accuracy benefits. In scenarios where computational resources are abundant, Wide-TSNet emerges as a compelling choice.

The model that employed the Support Vector Machine achieved the second-highest accuracy of 0.90. This is possibly due to the versatility and effectiveness of SVM in high-dimensional spaces. SVM uses kernel functions to transform the data into higher-dimensional spaces. This flexibility allows SVM to capture non-linear relationships in the Bitcoin dataset.

While our model primarily utilizes Bitcoin prices, it is important to acknowledge that in real-world scenarios, numerous factors influence Bitcoin prices. These factors need to be considered in prediction models. Currently, our model requires high computational resources, and introducing further advanced feature extraction methods would increase complexity, necessitating even higher computational resources.

Furthermore, the research lacks a thorough understanding of how the model performs outside of a controlled environment. Evaluating its effectiveness in real-world applications is crucial. The model's versatility and robustness have not been thoroughly tested across different economic scenarios, potentially limiting its applicability to specific markets or financial instruments.

6. Conclusions

The paper presented Wide-TSNet, a novel hybrid approach for Bitcoin price prediction using time-series data and convolutional neural networks (CNNs). The study introduced a tripartite classification system, comprising 'increase', 'decrease', and 'stable' classes, in order to accurately represent Bitcoin price trends. Image-based techniques are employed to transform time-series data into Markov transition fields (MTFs), which are enhanced

through histogram equalization for better pattern discernibility. The results of this study demonstrate the effectiveness of Wide-TSNet in predicting Bitcoin price movements.

The research also suggests that lightweight CNN models, such as SqueezeNet and EfficientNet, present comparable performance to more complex models in certain configurations.

The Wide-TSNet model showcases a promising research direction relating to financial time-series prediction, with particular application in the volatile cryptocurrency market. Through leveraging image-based CNNs, the model can achieve high accuracy and F1 scores, indicating its potential as a reliable tool for investors and analysts. However, the results of this study also revealed the importance of image resolution settings and the impact of image enhancement techniques on model performance.

In future research, we hope to explore the following areas:

- Further, lightweight CNN architectures and their configurations will be investigated to enhance model performance while maintaining computational efficiency.
- We intend to develop advanced methods for feature extraction from time-series images to improve the model's ability to identify subtle trends and patterns.
- We aim to investigate the feasibility of employing various test-to-train data splits, along with additional evaluation metrics. These endeavors would facilitate a more comprehensive and nuanced analysis.
- The presented model will be implemented in a real-time trading environment to assess its practicality and responsiveness to live market conditions. It is crucial to further explore the practical implementation of the model in real-world applications. This will help understand the model's performance and effectiveness outside of a controlled environment.
- The model's application will be extended to other financial instruments and markets in order to validate its versatility and robustness across different economic scenarios. In addition, insights from the finance, economics, and computer science fields will be combined to refine the model's predictive capabilities and understand the underlying factors influencing price fluctuations.

By addressing these suggestions, future research can build upon the foundation laid by Wide-TSNet, potentially leading to more sophisticated and accurate predictive models for financial time-series data. The ultimate goal is to provide a tool that not only forecasts prices but also offers insights into the mechanics of market movements, thus contributing to a more stable and informed financial ecosystem.

Author Contributions: Conceptualization, P.T.Y.; methodology, P.T.Y.; software, P.T.Y.; validation, P.T.Y. and T.Z.; formal analysis, P.T.Y.; investigation, P.T.Y.; resources, P.T.Y.; data curation, P.T.Y.; writing—original draft preparation, P.T.Y.; writing—review and editing, P.T.Y., T.Z. and P.K.G.; visualization, P.T.Y.; supervision, Y.L.; project administration, P.T.Y.; funding acquisition, P.T.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhou, Y.; Jiang, R.; Qin, H.; Hu, H. Representation and analysis of time-series data via deep embedding and visual exploration. *J. Vis.* **2023**, *26*, 593–610. [CrossRef]
2. Snega, S.; Nivedha, B. Bitcoin Price Prediction Using ML. *SSRN* **2022**. [CrossRef]
3. Martynov, O. Sustainability Analysis of Cryptocurrencies Based on Projected Return on Investment and Environmental Impact. Master's Thesis, Harvard University, Cambridge, MA, USA, 2020.
4. Giudici, G.; Milne, A.; Vinogradov, D. Cryptocurrencies: Market analysis and perspectives. *J. Ind. Bus. Econ.* **2020**, *47*, 1–18. [CrossRef]

5. Fang, F.; Ventre, C.; Basios, M.; Kanthan, L.; Martinez-Rego, D.; Wu, F.; Li, L. Cryptocurrency trading: A comprehensive survey. *Financ. Innov.* **2022**, *8*, 13. [CrossRef]
6. Baur, D.G.; Dimpfl, T. The volatility of Bitcoin and its role as a medium of exchange and a store of value. *Empir. Econ.* **2021**, *61*, 2663–2683. [CrossRef] [PubMed]
7. Si, Y. Using ARIMA Model to Analyze and Predict Bitcoin Price. *BCP Bus. Manag.* **2022**, *34*, 1210–1216. [CrossRef]
8. Siami-Namini, S.; Tavakoli, N.; Siami Namin, A. The Performance of LSTM and BiLSTM in Forecasting Time Series. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3285–3292. [CrossRef]
9. Yang, C.-L.; Chen, Z.-X.; Yang, C.-Y. Sensor Classification Using Convolutional Neural Network by Encoding Multivariate Time Series as Two-Dimensional Colored Images. *Sensors* **2020**, *20*, 168. [CrossRef] [PubMed]
10. Qin, Z.; Zhang, Y.; Meng, S.; Qin, Z.; Choo, K.-R. Imaging and fusing time series for wearable sensor-based human activity recognition. *Inf. Fusion* **2020**, *53*, 80–87. [CrossRef]
11. Wang, Z.; Oates, T. Imaging Time-Series to Improve Classification and Imputation. In: Twenty-Fourth International Joint Conference on Artificial Intelligence. *arXiv* **2015**. [CrossRef]
12. Roy, S.; Bhalla, K.; Patel, R. Mathematical analysis of histogram equalization techniques for medical image enhancement: A tutorial from the perspective of data loss. *Multimed. Tools Appl.* **2023**, *83*, 14363–14392. [CrossRef]
13. Zagoruyko, S.; Komodakis, N. Wide residual networks. *arXiv* **2016**, arXiv:1605.07146 2016.
14. Jeong, Y.-S.; Jeong, M.K.; Omitaomu, O.A. Weighted Dynamic Time Warping for Time Series Classification. *Pattern Recognit.* **2011**, *44*, 2231–2240. [CrossRef]
15. Matsuo, S.; Wu, X.; Atarsaikhan, G.; Kimura, A.; Kashino, K.; Iwana, B.K.; Uchida, S. Deep attentive time warping. *Pattern Recognit.* **2023**, *136*, 109201. [CrossRef]
16. Herrmann, M.; Tan, C.W.; Webb, G.I. Parameterizing the cost function of dynamic time warping with application to time series classification. *Data Min. Knowl. Disc* **2023**, *37*, 2024–2045. [CrossRef]
17. Hills, J.; Lines, J.; Baranauskas, E.; Mapp, J.; Bagnall, A. Classification of time series by shapelet transformation. *Data Min. Knowl. Discov.* **2014**, *28*, 851–881. [CrossRef]
18. Grabocka, J.; Wistuba, M.; Schmidt-Theme, L. Fast classification of univariate and multivariate time series through shapelet discovery. *Knowl. Inf. Syst.* **2016**, *49*, 429–454. [CrossRef]
19. Renault, A.; Bondu, A.; Lemaire, V.; Gay, D. Automatic Feature Engineering for Time Series Classification: Evaluation and Discussion. In Proceedings of the 2023 International Joint Conference on Neural Networks (IJCNN), Gold Coast, Australia, 18–23 June 2023; pp. 1–10.
20. Wu, D.; Peng, F.; Cai, C.; Du, X. Time Series Classification Based on Adaptive Feature Adjustment and Multi-scale AGRes2Net. *Neural Process. Lett.* **2023**, *55*, 8441–8463. [CrossRef]
21. Fauvel, K.; Fromont, É.; Masson, V.; Faverdin, P.; Termier, A. XEM: An explainable-by-design ensemble method for multivariate time series classification. *Data Min. Knowl. Discov.* **2022**, *36*, 917–957. [CrossRef]
22. Bertsimas, D.; Boussioux, L. Ensemble Modeling for Time Series Forecasting: An Adaptive Robust Optimization Approach. *arXiv* **2023**, arXiv:2304.04308.
23. Hajirahimi, Z.; Khashei, M. Hybridization of hybrid structures for time series forecasting: A review. *Artif. Intell. Rev.* **2023**, *56*, 1201–1261. [CrossRef]
24. Elen, A.; Avuçlu, E. A hybrid machine learning model for classifying time series. *Neural Comput. Appl.* **2022**, *34*, 1219–1237. [CrossRef]
25. Abouelnaga, M.; Vitay, J.; Farahani, A. Multivariate Time Series Classification: A Deep Learning Approach. *arXiv* **2023**, arXiv:2307.02253.
26. Jaleel, M.; Kucukler, O.; Alsalemi, A.; Amira, A.; Malekmohamadi, H.; Diao, K. Analyzing Gas Data Using Deep Learning and 2-D Gramian Angular Fields. *IEEE Sens. J.* **2023**, *23*, 6109–6116. [CrossRef]
27. Alsalemi, A.; Amira, A.; Malekmohamadi, H.; Diao, K. Lightweight Gramian Angular Field classification for edge internet of energy applications. *Clust. Comput.* **2023**, *26*, 1375–1387. [CrossRef]
28. Zhang, Y.; Hou, Y.; Zhou, S.; Ouyang, K. Encoding Time Series as Multi-Scale Signed Recurrence Plots for Classification Using Fully Convolutional Networks. *Sensors* **2020**, *20*, 3818. [CrossRef]
29. Zhang, Y.; Hou, Y.; OuYang, K.; Zhou, S. Multi-scale signed recurrence plot based time series classification using inception architectural networks. *Pattern Recognit.* **2022**, *123*, 108385. [CrossRef]
30. Kirichenko, L.; Zinchenko, P. Time Series Classification Based on Visualization of Recurrence Plots. In *Tools and Methods of Program Analysis, Proceedings of the International Conference on Tools and Methods for Program. Analysis. TMPA 2019, Tbilisi, Georgia, 7–9 November 2019*; Springer: Cham, Switzerland, 2021; pp. 101–108.
31. Zhao, X.; Sun, H.; Lin, B.; Zhao, H.; Niu, Y.; Zhong, X.; Wang, Y.; Zhao, Y.; Meng, F.; Ding, J.; et al. Markov transition fields and deep learning-based event-classification and vibration-frequency measurement for φ-OTDR. *IEEE Sens. J.* **2021**, *22*, 3348–3357. [CrossRef]
32. Liu, B.; Cheng, H. Financial time series classification method based on low-frequency approximate representation. *Eng. Rep.* **2023**, *1*, e12739. [CrossRef]

33. Tripathi, B.; Sharma, R.K. Modeling Bitcoin Prices using Signal Processing Methods, Bayesian Optimization, and Deep Neural Networks. *Comput. Econ.* **2023**, *62*, 1919–1945. [CrossRef]
34. Fons, E.; Dawson, P.; Zeng, X.; Keane, J.; Iosifidis, A. "Evaluating Data Augmentation for Financial Time Series Classification." Quantitative Finance. *arXiv* **2020**. [CrossRef]
35. Dezhkam, A.; Manzuri, M.T.; Aghapour, A.; Karimi, A.; Rabiee, A.; Shalmani, S.M. A Bayesian-based classification framework for financial time series trend prediction. *J. Supercomput.* **2023**, *79*, 4622–4659. [CrossRef]
36. Garcia, G.R.; Michau, G.; Ducoffe, M.; Gupta, J.S.; Fink, O. Temporal signals to images: Monitoring the condition of industrial assets with deep learning image processing algorithms. *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* **2022**, *236*, 617–627. [CrossRef]
37. Kaur, H.; Sohi, N. A Study for Application of Histogram in Image Enhancement. *Int. J. Eng. Sci. (IJES)* **2017**, *6*, 59–63. [CrossRef]
38. Qi, Y.; Yang, Z.; Sun, W.; Lou, M.; Lian, J.; Zhao, W.; Deng, X.; Ma, Y. A Comprehensive Overview of Image Enhancement Techniques. *Arch. Comput. Methods Eng.* **2022**, *29*, 583–607. [CrossRef]
39. Cheng, H.D.; Shi, X.J. A Simple and Effective Histogram Equalization approach to Image Enhancement. *Digit. Signal Process.* **2004**, *14*, 158–170. [CrossRef]
40. Wu, X.; Kawanishi, T.; Kashino, K. Reflectance-Guided, Contrast-Accumulated Histogram Equalization. *arXiv* **2022**, arXiv:2209.06405v1.
41. Oliver, W.R. Histogram Stretching Or Histogram Equalization In Image Processing. *Microsc. Today* **1998**, *6*, 20–24. [CrossRef]
42. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaria, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 53. [CrossRef] [PubMed]
43. O'Shea, K.; Nash, R. An Introduction to Convolutional Neural Networks. *arXiv* **2015**, arXiv:1511.08458.
44. Widodo, S.P.; Rachmawati, N. Exploration of Resnet Variants in High Spatial Resolution Domain Adaptation: From air-to-space imagery. *Proc. Int. Conf. Data Sci. Off. Stat.* **2023**, *2023*, 38–46. [CrossRef]
45. Bello, I.; Fedus, W.; Du, X.; Cubuk, E.D.; Srinivas, A.; Lin, T.-Y.; Shlens, J.; Zoph, B. Revisiting resnets: Improved training and scaling strategies. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22614–22627. [CrossRef]
46. Thakur, A.; Chauhan, H.; Gupta, N. Efficient ResNets: Residual Network Design. *arXiv* **2023**, arXiv:2306.12100.
47. Chen, L.-C.; Wang, H.; Qiao, S. Scaling wide residual networks for panoptic segmentation. *arXiv* **2020**, arXiv:2011.11675.
48. Tang, Y.; Arias-Calluari, K.; Harré, M.S.; Alonso-Marroquin, F. Stylized Facts of High-Frequency Bitcoin Time Series. *arXiv* **2024**, arXiv:2402.11930.
49. CoinMarketCap. Bitcoin Price History Dataset. 2023. Available online: https://coinmarketcap.com/currencies/bitcoin (accessed on 24 October 2023).
50. Faouzi, J.; Janati, H. Pyts: A Python Package for Time Series Classification. *J. Mach. Learn. Res.* **2020**, *21*, 1–6.
51. Howard, J.; Gugger, S. Fastai: A layered API for deep learning. *Information* **2020**, *11*, 108.
52. Iandola, F.N.; Han, S.; Moskewicsz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
53. Tan, M.; Le, Q.V. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
54. Sin, E.; Wang, L. Bitcoin price prediction using ensembles of neural networks. In Proceedings of the 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 29–31 July 2017; pp. 666–671.
55. Tanwar, S.; Patel, N.P.; Patel, S.N.; Patel, J.R.; Sharma, G.; Davidson, I.E. Deep learning-based cryptocurrency price prediction scheme with inter-dependent relations. *IEEE Access* **2021**, *9*, 138633–138646. [CrossRef]
56. McNally, S.; Roche, J.; Caton, J. Predicting the price of bitcoin using machine learning. In Proceedings of the 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP) 2018, Cambridge, UK, 21–23 March 2018; pp. 339–343.
57. Critien, J.V.; Gatt, A.; Ellul, J. Bitcoin price change and trend prediction through twitter sentiment and data volume. *Financ. Innov.* **2022**, *8*, 45. [CrossRef]
58. Hitam, N.A.; Ismail, A.R. An Optimized Support Vector Machine (SVM) based on Particle Swarm Optimization (PSO) for cryptocurrency forecasting. *Proc. Comput. Sci.* **2019**, *163*, 427–433. [CrossRef]