Expert Systems With Applications 298 (2026) 129555

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A spectral relevance analysis approach to pattern recognition of financial time series

Christine Distler [a], Yarema Okhrin [b,*], Jonathan Pfahler [c]

[a] Institute for Employment Research (IAB), Nuremberg, Germany
[b] Chair of Statistics and Data Science, Faculty of Business and Economics, University of Augsburg, Germany
[c] Allianz SE, Munich, Germany

ARTICLE INFO

Keywords:
Pattern recognition
Cryptocurrency
GAF
CNN
Finance
Bitcoin

ABSTRACT

Understanding patterns in financial time series is crucial for improving prediction accuracy in algorithmic trading and risk management. This paper presents a novel AI-based computer vision approach for classifying financial time series. Historical price sequences are transformed into Gramian Angular Difference Field (GADF) images and fed into a convolutional neural network (CNN) for pattern recognition. To interpret the CNN's decision-making process, we apply Spectral Relevance Analysis (SpRAy), enabling the identification of distinct clusters based on relevance maps. Clustering the images according to their relevance profiles reveals groups with significantly higher predictive performance compared to the full dataset. The corresponding relevance patterns highlight favorable price movement structures and are identified via the associated clusters.

1. Introduction

Technical analysis and fundamental analysis are two widely recognized approaches in trading. Both methods are employed by active traders in financial markets, although they contradict the assumptions of the efficient market hypothesis. In essence, both approaches assume that a publicly traded asset is temporarily mispriced - either undervalued or overvalued. Traders use these techniques to estimate the "fair" value of an asset and capitalize on the discrepancy. When an asset is perceived to be undervalued, traders may take a long position, anticipating price increases. Conversely, when an asset appears overvalued, they might take a short position, expecting the price to fall.

While fundamental analysis has demonstrated its effectiveness in various scenarios, the validity of technical analysis remains a subject of debate. Only a slight majority of studies have shown that technical analysis methods lead to consistent outperformance. For instance, Gerritsen (2016) argues that most commonly used technical trading rules fail to produce excess returns. Furthermore, Hoffmann and Shefrin (2014) highlights that excessive trading, driven by high turnover rates, diminishes returns. However, in contrast, Dourra and Siy (2002) suggests that technical trading rules can indeed generate extraordinary returns - but only when applied by traders with advanced mathematical expertise.

In technical analysis, traders typically rely on pre-defined indicators or patterns to predict price movements. However, the rapid advancements in machine learning have enabled the use of algorithms to identify the most profitable patterns. For example, Miller et al. (2019) apply smoothing splines to high-frequency *Bitcoin* (BTC) price data and demonstrated that regression splines can effectively identify profitable technical analysis patterns and trading strategies for BTC time series data. Similarly, Bose et al. (2021) propose a hybrid model combining multivariate adaptive regression splines (MARS) with a deep neural network (DNN) to predict stock closing prices, achieving an impressive 92 % accuracy in predictions. Another hybrid model is propsed by Ku et al. (2023), who present a flexible LSTM-based model for stock forecasting that integrates domain knowledge from investors into the feature selection process. Their model allows investors to specify preferred indicators, resulting in significantly better predictive accuracy and risk-adjusted returns. Their findings suggest that informed indicator selection, combined with LSTM's ability to model temporal dependencies, leads to more robust and actionable predictions in financial markets. As is typical in time series prediction, features are fed into the models as numeric sequences. Alternatively, data can be transformed into images, as demonstrated by Li et al. (2020). Modern image analysis techniques, particularly computer vision models (Chai et al., 2021), yield superior performance, not only in image recognition and classification, but also in financial applications. For instance, Jiang et al. (2023) uses *convolutional neural networks* (CNNs) to predict stock-market trends based on price charts, including open-high-low-close bars, moving average lines, and trading volume bars. This approach outperforms traditional price trend models. Chen et al. (2016) developed the mean average mapping

* Corresponding author.
E-mail addresses: christine.distler@iab.de (C. Distler), yarema.okhrin@uni-a.de (Y. Okhrin), jonathan.pfahler@allianz.de (J. Pfahler).

https://doi.org/10.1016/j.eswa.2025.129555
Received 8 May 2025; Received in revised form 10 August 2025; Accepted 28 August 2025
Available online 30 August 2025

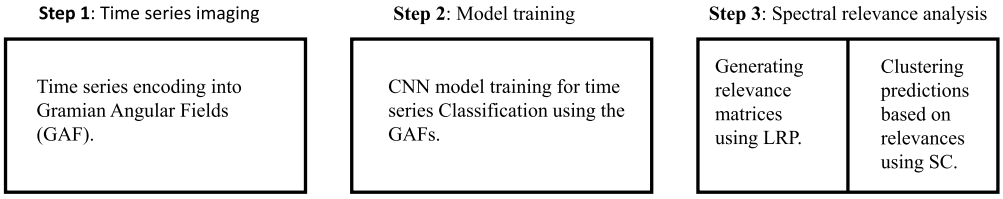


Fig. 1. Three-step framework of the proposed method.

method to convert financial time series data into 2-dimensional plots, preserving all relevant information from the time series. These plots are then used as features to train a CNN, which predicts future price trends. In trading simulations, the CNN demonstrates robust results and strong predictive performance. Cohen et al. (2020) employ candlestick charts as inputs to train CNNs for price prediction, yielding similarly strong results. They also argue that treating financial time series classification as a computer vision problem can be particularly useful for technical analysis. Finally, Barra et al. (2020) use an ensemble of CNNs to predict future trend of the S&P 500 index. They input *Gramian Angular Fields* (GAF), which preserve time series information, and compare their approach against a buy-and-hold strategy, random guessing, and a 1-D CNN that uses numerical data. Their results indicate that the ensemble model outperforms the benchmark strategies, providing robust predictive results.

The aforementioned studies indicate that representing time series data as images can offer advantages over purely numerical formats. According to Jiang et al. (2023), this is primarily because images serve as the default input for CNNs, which are particularly well-suited for pattern recognition in visual data. The authors argue that CNNs can leverage their intrinsic capabilities, such as automatic feature extraction, to identify complex structures more effectively. This is especially relevant in financial markets, where underlying patterns are often intricate, making it beneficial to extract features directly from raw data rather than relying on manual feature engineering. Beyond financial forecasting, hybrid deep learning models have also shown strong performance in other domains: He et al. (2024) propose a hybrid model for epileptic seizure detection based on EEG signals, combining wavelet-based signal decomposition, CNNs, and an attention-based transformed LSTM. Their results demonstrate that such integrated models outperform conventional baselines, underscoring the potential of advanced deep learning pipelines for time series classification. Similarly, Yang et al. (2025) present a novel method for detecting malicious HTTP requests by combining an autoencoder with a transductive LSTM and a GAN architecture. Their approach achieves state-of-the-art performance on multiple benchmark datasets, highlighting the adaptability and robustness of LSTM-based hybrid models beyond the financial and medical domains.

In this paper, we propose an approach that utilizes a CNN to automatically extract relevance patterns for time series forecasting and demonstrate how to identify the patterns with the highest predictive performance. Our approach follows a three-step procedure, as illustrated in Fig. 1.

In the second step, we use the GADF matrices as input data for a ResNet50 CNN to predict the direction of BTC price movements. While the trained models achieve reasonable forecasting performace, they do not exhibit extraordinary predictive accuracy. We hypothesize that certain subsamples within the dataset yield higher predictive power compared to the overall performance across the full dataset. In the final step, we identify these high-performing subsamples by clustering the GADF matrix observations based on the importance of individual values for prediction. To achieve this, we apply spectral relevance analysis (SpRAy). Specifically, We compute relevance scores for each value in the input data using the layer-wise relevance propagation (LRP) method, a widely used approach eXplainable Artificial Intelligence. The resulting relevance score matrices for each GADF matrix are then clustered using spectral clustering (SC) (Anders et al., 2022; Lapuschkin et al., 2019; von Luxburg, 2007). This relevance-based clustering approach allows us to identify clusters that outperform the overall CNN model in predictive accuracy and to investigate the underlying pattern in the relevance matrices associated with superior predictive performance. In doing so, we detect specific patterns within the relevance matrices that correspond to enhanced forecastability.

Our study contributes to the literature on time series classification and pattern recognition in two key ways:

- Forecastability of BTC price movements using computer vision models: We demonstrate that BTC price movements can be forecasted using CNNs when the lagged price movements sequences are encoded as GADF matrices. We validate this approach on minute-level high-frequency time series data across five different forecasting horizons.
- Relevance-based clustering to identify predictive patterns: We introduce a novel methodology for identifying patterns associated with excess returns using SpRAy. By clustering observations based on the relevance scores of GADF matrix values, we uncover distinct decision patterns within the CNN's relevance matrices. Several identified clusters exhibit higher accuracy scores than the overall CNN model, demonstrating the potential of relevance-based clustering for financial time series prediction.

To highlight the novelty of our approach, Table 1 provides a schematic comparison with related studies that combine image-encoded time series data with CNN models and, in some cases, XAI techniques. To the best of our knowledge, no prior study has combined image-encoding, CNNs, LRP, and subsequent clustering of relevance matrices to identify highly forecastable patterns.

The table shows that several studies have explored individual components of our approach - particularly the combination of image encoding and CNNs has received considerable attention in financial forecasting research. However, only very few studies incorporate explainability techniques such as LRP, and none to our knowledge combine all four elements: image-encoded time series, CNN architectures, LRP-based relevance extraction, and clustering of relevance maps. Our method thus represents a novel contribution that builds upon existing components but introduces a new pipeline for identifying and exploiting high-predictive structures in financial time series data.

The remainder of this paper is organized as follows: Section 2 outlines the research design and empirical approach, providing an overview of image encoding, CNNs, LRP, and SC. Section 3 presents the experimental study and Section 4 discusses the key empirical findings. Finally, Section 5 concludes the paper and highlights avenues for future research.

2. Methodology

We investigate the presence of recurring patterns in high-frequency financial time series with the goal of identifying structures that exhibit robust predictive power for the direction of future price movements. Specifically, we aim to determine whether certain patterns are particularly reliable in forecasting the direction of asset price changes.

As a necessary preprocessing step, we construct partially overlapping subsequences from the original time series, and accordingly define binary target variables.

Table 1
Methodological comparison with related GAF-CNN-LRP studies.

Study	Focus and Method	Methodological overlap
Jiang et al. (2023)	Use CNNs to predict stock market trends based on image-encoded price charts. Focuses on outperforming traditional trend models using visual inputs.	**
Barra et al. (2020)	Employ an ensemble of CNNs on GAF to forecast the S&P 500 index. Benchmarked against buy-and-hold, random guessing, and 1D CNNs on numerical input.	**
Chen et al. (2016)	Propose the Mean Average Mapping method to convert financial time series into 2D plots for CNN-based trend prediction. Demonstrate robust simulation performance.	**
Xu and Lin (2023)	Develop a quantum-enhanced federated learning model using quantum GAFs as input features for CNN-based stock forecasting.	**
Singh and Singh (2024)	Compare the performance of CNN and CNN-LSTM hybrids in predicting Indian stock prices, focusing on model architecture rather than data encoding or explainability.	*
Kumar et al. (2025)	Integrate XAI (SHAP) into LSTM-based and Prophet forecasting models. Emphasize interpretability in classical time series contexts.	**
Carta et al. (2022)	Apply XAI-based feature selection techniques for next-day stock return prediction. Combine explainability with classical indicators rather than deep visual models.	**
Huang et al. (2023)	Propose a hybrid model combining fuzzy C-means clustering and CNNs to forecast corporate financial performance. Use entropy-based feature weighting and correlation filtering.	**
Arratia and Sepúlveda (2020)	Convert financial time series into image representations to leverage CNNs' pattern recognition for forecasting. Emphasize the visual structure of time-dependent data.	**
This study	Introduce a GADF-based CNN model for BTC forecasting, enhanced by spectral clustering on relevance maps. Identify high-performing prediction patterns through cluster analysis.	****

Note: Stars indicate methodological overlap with our proposed approach. One star (*) denotes that a study employs one of the core components used in our method (i.e., image-encoding, CNN architecture, XAI methods, or clustering). Two stars (**) indicate the use of two components, and so on.

Let the time series be given by $X = \{x_1, x_2, \dots, x_S\}$, where S denotes the number of real-valued observations. This series is segmented into overlapping subsequences $X^{(i)}$, each of length m, such that:

$$X^{(i)} = \{x_{i-(m-1)}, x_{i-(m-2)}, \dots, x_i\}, \tag{1}$$

with $i = m, \dots, (S-h)$, and h representing the prediction horizon. For each sequence $X^{(i)}$, we define a binary target variable $c_h^{(i)}$ indicating whether the average price over the next h minutes exceeds the current price x_i:

$$c_h^{(i)} = \begin{cases} 1 & \text{if } \bar{X}_h^{(i+1)} > x_i, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where the average is given by $\bar{X}_h^{(i+1)} = \frac{1}{h}\sum_{j=1}^{h} x_{i+j}$. This formulation yields a total of $M = (S-h) - m$ labeled subsequences per prediction horizon h. Due to the binary nature of the target variable, the forcasting task is framed as a binary classification problem.

The classification problem is set as a multi-step procedure. First, each price sequence is transformed into a GADF matrices, enabling a 3-dimensional representation of temporal dynamics. Second, the resulting GADF matrices serve as inputs for training a ResNet-50 CNN model, which is tasked with predicting the binary target variables. Third, to enhance interpretability, the decision-making process of the deep neural network is analyzed through the generation of relevance maps, which are subsequently clustered using SpRAy. The following sections provide a detailed description of each step.

2.1. Image encoding

In this section, we describe the process of encoding time series sequences as GADF matrices. This encoding framework was introduced by Wang and Oates (2015a), on which the following explanations are based. For simplicity, we omit the sequence index i and the index for the time horizon h.

First, each generic time series sequence $X = (x_1, x_2, \dots, x_m)'$ is rescaled in the interval $[-1, 1]$. The scaled observations are denoted as $\tilde{x}_j$ with $j = 1, \dots, m$:

$$\tilde{x}_j = \frac{(x_j - \max(X)) + (x_j - \min(X))}{\max(X) - \min(X)}. \tag{3}$$

By scaling the time series sequences, we can utilize the inverse cosine function to represent $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_m)'$ in polar coordinates. The rescaled observations are encoded as the angular cosine ϕ_j, while the timestamp $t_j \in \{1, \dots, m\}$ serves as the radius r_j. The polar transformation is defined as follows, where m regulates the span of the polar coordinate system:

$$\begin{cases} \phi_j = \arccos(\tilde{x}_j) \\ r_j = \frac{t_j}{m}. \end{cases} \tag{4}$$

The observation at timestamp t_1 is positioned closest to the pole, with subsequent timestamps moving outward. The arccosine function maps values to the interval $[0, \pi]$, ensuring a bijective transformation. A key advantage of this mapping is the preservation of absolute temporal relationships. In Cartesian coordinates, the area between two timestamps t_s and t_j depends on $|x_{t_s} - x_{t_j}|$, whereas in polar coordinates, the sector area is determined by the absolute values of the timestamps t_s and t_j. The area in Cartesian coordinates is given by $A_{s,j} = \int_s^j f(x)dx$, ensuring that if $f(x)$ on $[s, s+z]$ is identical to $f(x)$ on $[j, j+z]$, then $A_{s,s+z} = A_{j,j+z}$. However, in polar coordinates, the corresponding area is defined as $A'_{s,j} = \int_{\phi_s}^{\phi_j} r^2 d\phi$, meaning that the area is independent on the time interval $|s-j|$ and $A'_{s,s+z} \neq A'_{j,j+z}$.

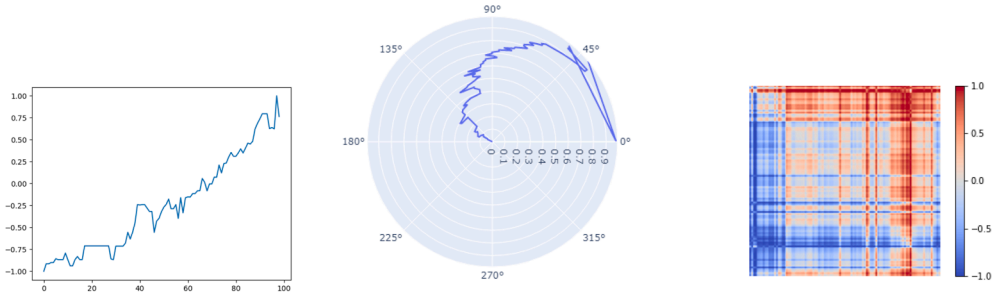
To complete the encoding process, we define the GADF matrix. Let $\mathbf{I}$ denote the unit vector of length m, i.e. $(1, 1, \dots, 1)'$. The GADF matrix G is computed as follows:

$$\begin{aligned} G &= \{G_{js}\}_{j,s=1,\dots,m} = \{\sin(\phi_j - \phi_s)\}_{j,s=1,\dots,m} \\ &= \left(\mathbf{I} - \tilde{X} \circ \tilde{X}\right)^{\circ 1/2} \cdot \tilde{X}' - \tilde{X} \cdot \left(\mathbf{I} - \tilde{X} \circ \tilde{X}\right)^{\circ 1/2\,\prime}, \end{aligned} \tag{5}$$

where $\circ$ denotes the Hadamard product and $\circ 1/2$ the Hadamard root, i.e. the element-wise square root. The elements of the resulting $m \times m$ matrix G lie within the interval $[-1, 1]$. The GADF matrix provides two key advantages:

i) Temporal dependency is preserved. The first value of the time series sequence x_1 corresponds to the top-left pixel G_{11}, and subsequent values are mapped progressively toward the bottom-right corner G_{mm}.
ii) Reconstruction of the rescaled time series sequence is possible. The diagonal elements G_{jj} contain the rescaled observations, enabling full reconstruction (Wang & Oates, 2015a,b).

The preservation of temporal structure is particularly relevant for our empirical approach. Fig. 2 illustrates the GADF generation process and visualizes the transformation using a conventional heatmap. The GADF matrices serves as input data for the classification task in the subsequent step.



(a) Price sequence (normalized). (b) Transformation to polar coordinates. (c) Heatmap of the GADF matrix.

Fig. 2. Illustration of the GADF encoding process from normalized BTC prices.

2.2. Classification

The encoded time series $\{G^{(i)}\}$ is used as input data for predicting our target variables $c_h^{(i)}$ using CNNs. In this subsection, we use the sequence index i, but omit the time horizon index h for simplicity.

CNNs are designed to process data with a grid-like topology, such as images, and are well-suited for capturing both local patterns and temporal dependencies through learnable filters. They reduce image dimensions while preserving essential features, resulting in a lower number of parameters compared to traditional feed-forward neural networks. CNNs are also inherently robust to shifts and transformations in the input data, making them highly effective for pattern recognition tasks. A basic CNN architecture typically consists of four main components:

i) *Input layer*. This layer contains the input data represented as a matrix. For RGB images, the input is a tensor with three channels, each containing values for a respective color (red, green, or blue). A grayscale image, on the other hand, is a two-dimensional matrix, representing pixel intensity. In our case, we use GADF matrices as inputs, which are treated like grayscale images.

ii) *Convolutional layers*. These layers use kernel functions, or filters, to extract local features and create feature maps. Multiple kernels can be applied to detect different patterns, as the choice of kernel determines which features are extracted. A node in the convolutional layer is computed as a spatial convolution between neighboring pixels of the input layer and a set of kernel weights. Let $d \in \mathbb{N}$ and $d \leq m$. For a filter of size $2d \times 2d$, the matrix of filter weights is defined as

$$\mathbf{\Omega}^{(1)} = \begin{pmatrix} \omega^{(1)}_{-d\ -d} & \omega^{(1)}_{-d\ -(d-1)} & \cdots & \omega^{(1)}_{-d\ d} \\ \omega^{(1)}_{-(d-1)\ -d} & \omega^{(1)}_{-(d-1)\ -(d-1)} & \cdots & \omega^{(1)}_{-(d-1)\ d} \\ \cdots & \cdots & \ddots & \vdots \\ \omega^{(1)}_{d\ -d} & \omega^{(1)}_{d\ -(d-1)} & \cdots & \omega^{(1)}_{d\ d} \end{pmatrix}_{(2d+1)\times(2d+1).}$$

Then the convolution operation is performed as follows:

$$z^{(1)}_{ij} = \mathbf{\Omega}^{(1)} * G = \sum_{s=-d}^{d} \sum_{s'=-d}^{d} \omega^{(1)}_{ss'} G_{(i+d)+s,\ (j+d)+s'}$$

where $i, j = 1, \ldots, m'$ with $m' = m - 2d$.

iii) *Pooling layers*. After each convolutional layer, a pooling layer reduces dimensionality, increases spatial invariance, and eliminates redundant pixels. Pooling is performed by partitioning a feature map into small regions and replacing all elements in each region with a single value. Three common pooling methods are average pooling, max pooling, and Euclidean pooling. The maxpooling operation is defined as:

$$G^{(*)}_{ij} = \max_{s,s'=1,\ldots,m^*} (G_{(i-1)m^*+s,(j-1)m^*+s'}),$$
$$i = 1, \ldots, \frac{m}{m^*},\ j = 1, \ldots, \frac{m}{m^*},$$

where m^* is the pooling region size.

iv) *Fully connected layers and output*. Extracted features are passed into fully connected layers, leading to the final classification output L. The number of neurons in the output layer, d_L, corresponds to the number of classes. Here, the probability of an image belonging to a specific class is determined by applying the softmax activation function. It maps the outputs of the last layer to multinomial probabilities by using the normalized exponent of the input values. This guarantees that $\hat{y}^{(i)} \in [0, 1]$ and $\hat{y}^{(i)} + (1 - \hat{y}^{(i)}) = 1$, where $\hat{y}^{(i)}$ denotes the predicted probability for class 1 and $(1 - \hat{y}^{(i)})$ denotes the probability for class 0 in our case of binary classification. The discrete binary class prediction $\hat{c}^{(i)}$ is obtained as follows:

$$\hat{c}^{(i)} = \begin{cases} 1 & \text{if } 1 - \hat{y}^{(i)} < \hat{y}^{(i)}, \\ 0 & \text{else.} \end{cases} \tag{6}$$

Instead of building a CNN architecture from scratch, we use the well-established ResNet-50 model (He et al., 2015), which is widely used for image classification. Compared to conventional CNNs, ResNet-50 introduces skip connections, intensive batch normalization, and identity mappings. Identity mappings add the original input to the output of operations within the residual model:

$$H(x) = F(x) + x, \tag{7}$$

where $F(x)$ is the transformation performed by convolutional layers, and x is the input to the residual block. These skip connections mitigate the vanishing gradient problem in deep neural networks, improving training stability. For further details on residual networks, see He et al. (2015).

2.3. Pattern recognition

Convolutional Neural Networks are often considered black-box models due to their complex and non-transparent decision processes (Molnar, 2025). To interpret their predictions, various techniques from the field of Explainable AI (XAI) have been developed.

One widely used method for interpreting neural networks at the local level is LRP (Bach et al., 2015). LRP decomposes the prediction of a CNN by assigning relevance scores to input features, indicating their contribution to the model's decision. This allows for a detailed, instance-specific understanding of how input values influence the output.

2.3.1. Local explanations

LRP is commonly used to generate relevance maps for input images, identifying which pixels contribute most to a model's prediction (Bach et al., 2015). In our setting, we apply LRP to GADF matrices and aim to quantify the relevance of each value in the matrix for the classification of the corresponding sequence.

To simplify notation, we omit the sequence index i and forecasting horizon h in the following. Let $G \in [-1, 1]^{m \times m}$ denote the GADF matrix, and let $\hat{y}_c$ be the model's predicted probability for class c. The goal of

LRP is to compute a relevance score $R_g(G)$ for each element G_g such that the total relevance equals the prediction score:

$$\forall\ G : \hat{y}_c = \sum_g R_g(G). \tag{8}$$

The relevance propagation starts at the output layer L, where is initialized as the neuron activations: $R_p^{(L)} = a_p^{(L)}$ with $p = 1, \dots, d_L$. Relevance is then propagated backward layer by layer, redistributing it across neurons in earlier layers. Highly activated neurons and strong positive weights lead to greater relevance attribution in the lower layers.

Different propagation rules are applied depending on the network layer:

- LRP-0 is used in upper layers and redistributes relevance proportionally to the weighted activations:

$$R_p^{(l)} = \sum_{q^*=1}^{d_{l+1}} \frac{a_p^{(l)} w_{pq^*}}{\sum_{p^*=1}^{d_l} a_{p^*}^{(l)} w_{p^*q^*}} R_{q^*}^{(l+1)}, \tag{9}$$

  with p and q denoting two neurons at the consecutive layers l and $l+1$. The activation of the neuron p is denoted as $a_p^{(l)}$ and the weight connecting the two neurons is as w_{pq}.
- LRP-ϵ adds a small stabilizing term ϵ to the denominator to reduce the influence of weakly activated neurons and noise. LRP-ϵ is used for middle layers, which show more disentangled representations.
- LRP-γ, used in lower layers, emphasizes positive contributions by modifying weights:

$$R_p^{(l)} = \sum_{q^*=1}^{d_{l+1}} \frac{a_p^{(l)} (w_{pq^*} + \gamma w_{pq^*}^+)}{\sum_{p^*=1}^{d_l} a_{p^*}^{(l)} (w_{p^*q^*} + \gamma w_{p^*q^*}^+)} R_{q^*}^{(l+1)}, \tag{10}$$

  where $w_{pq^*}^+$ indicates that only positive weights are considered. Larger γ values strengthen the focus on features with a positive influence on the prediction.

Following these rules, we generate a heatmap for each GADF matrix that visualizes the relevance distribution across its values. These local explanations are then grouped using SC to identify common relevance patterns across the entire dataset. LRP is particularly well-suited to our setting for two reasons. First, it provides fine-grained, instance-specific relevance scores, which are essential for analyzing individual GADF matrices derived from high-frequency time series. Second, unlike perturbation-based methods, LRP does not require repeated model evaluations with modified inputs. This makes it computationally efficient and well adapted to large-scale, high-resolution image-like representations such as GADFs.

2.3.2. Clustering

As outlined above, our objective is to identify clusters of CNN decision behavior that exhibit higher predictive performance compared to the model's performance on the full dataset. To this end, we first compute local explanations for each GADF matrix using LRP, resulting in relevance maps that indicate the contribution of each input feature to the classification outcome.

The next step is to cluster these relevance maps such that local explanations within a cluster are highly similar, while those between clusters differ substantially. For this purpose, we adopt the SpRAy framework proposed by Lapuschkin et al. (2019), which combines LRP with SC. In SpRAy, LRP is first used to generate instance-specific relevance maps, which are then grouped via SC to uncover distinct patterns in the model's decision-making process. SC is particularly well suited for this task due to the high dimensionality of the relevance maps ($m \times m$ matrices).

SC interprets clustering as a graph partitioning problem: the goal is to partition a *similarity graph* such that intra-cluster edges (within subgraphs) have high weights, while inter-cluster edges (between subgraphs) have low weights (von Luxburg, 2007). The SC procedure consists of three main stages:

i) *Preprocessing.* A similarity graph $\mathcal{G} = (V, E)$ is constructed, where the vertex set $V = \{v_1, \dots, v_M\}$ represents the M relevance maps, each corresponding to a vectorized relevance matrix derived from a GADF matrix via LRP. An undirected edge exists between two vertices v_i and v_l if they are mutual k-nearest neighbors, meaning that v_l is among the k-nearest neighbors of v_i and vice versa. The resulting graph is referred to as a *mutual k-nearest neighbor graph* (von Luxburg, 2007). The construction of the similarity matrix is a crucial component of SC. Following the recommendation of von Luxburg (2007), we set the number of neighbors as $k = \log(M)$. To determine neighborhood relationships, pairwise similarities are computed using a Gaussian (RBF) similarity function:

$$s_{il} = \exp\Big(- \frac{||v_i - v_l||^2}{2\sigma^2} \Big), \tag{11}$$

   where the parameter σ controls the width of the neighborhood and how quickly similarity decays with increasing distance. The optimal value of σ is selected via grid search. Based on these pairwise similarities, we construct the weighted adjacency matrix $A = (s_{il})_{i,l=1,\dots,M}$.

ii) *Spectral representation.* In the second step, we compute the normalized graph Laplacian:

$$L^* = D^{-1/2}(D - A)D^{-1/2}, \tag{12}$$

   where D is the degree matrix with diagonal $d_i = \sum_{l=1}^M s_{il}$.

   Next, we compute the first e eigenvectors of L^*, corresponding to the e smallest eigenvalues, where e denotes the desired number of clusters. These eigenvectors capture the key structure of the similarity graph in a reduced-dimensional space. The e eigenvectors are combined column-wise into a matrix $U \in \mathbb{R}^{M \times e}$. To obtain a normalized spectral embedding, each row of U is scaled to unit norm, resulting in a new set of data points $z_1, \dots, z_M$ in $\mathbb{R}^e$, which serve as the input for the final clustering step.

iii) *Clustering.* In the final step, we apply the k-means algorithm to the set of points $z_1, \dots, z_M$ in the reduced eigenspace to partition the data into e clusters (Jia et al., 2014; Ng et al., 2001; von Luxburg, 2007).

The SC process is described in Algorithm 1, below. A critical aspect of SC is selecting the number of clusters, e. We employ the eigengap heuristic to determine e, which involves identifying a significant gap between the first e eigenvalues, $\lambda_1, \dots, \lambda_e$, and the remaining eigenvalues, $\lambda_{e+1}, \dots, \lambda_M$. The first e eigenvalues should be relatively small, while the subsequent eigenvalues should show a noticeable increase (Mohar, 1991).

Algorithm 1 Normalized spectral clustering.

Input: Weighted adjacency matrix $A \in \mathbb{R}^{M \times M}$, number e of subgraphs.

1: Compute the diagonal degree matrix $D \in \mathbb{R}^{M \times M}$ with diagonal $d_i = \sum_{l=1}^M s_{il}$.
2: Compute the normalized graph Laplacian $L^* = D^{-1/2}(D - A)D^{-1/2}$.
3: Form the matrix $U \in \mathbb{R}^{M \times e}$ containing the first e eigenvectors $u_1, \dots u_e$ of L^* as columns.
4: Renormalize each row of U to norm 1 by $t_{il} = u_{il}/(\sum_e u_{ie}^2)^{1/2}$ and form matrix $T \in \mathbb{R}^{M \times e}$ with elements t_{il}.
5: For $i = 1, \dots, M$, let $z_i \in \mathbb{R}^e$ be the vector corresponding to the i-th row of T.
6: Cluster the points z_i with the k-means algorithm into clusters $C_1, \dots, C_e$.

Output: Clusters $C_1, \dots, C_e$.

3. Experimental study

Our empirical analysis is based on the time series of BTC closing prices. The dataset consists of minute-level data from January 1st, 2018, to April 1st, 2022, containing a total of $S = 2,235,994$ numerical observations. The data are segmented into 2,235,895 sequences,

each consisting of $m = 100$ consecutive minute-level prices. Selecting an appropriate sequence length m involved balancing two competing goals: capturing sufficient temporal context for meaningful pattern recognition, and maintaining computational efficiency. A longer sequence allows the model to detect more complex and extended price structures. However, the size of both the GADF matrices and the LRP matrices grows quadratically with m, significantly increasing memory and processing demands. We chose $m = 100$ as a compromise between expressiveness and efficiency. This length surpasses common human-interpretable time intervals, such as one hour or 90 minutes, and thus enables the model to capture mid-term patterns that might elude shorter windows. At the same time, the resulting 100×100 GADF matrices (10,000 pixels) remain computationally manageable, allowing for efficient training and relevance analysis without exceeding memory or storage limitations.

For classification, we define five binary target variables $c_h^{(i)}$ with different forecasting horizons $h \in 1, 10, 30, 60, 100$. The target variable represents the direction of BTC price changes by comparing the most recent price with the future price, averaged over the next 10, 30, 60, or 100 minutes. Each sequence is encoded as a GADF matrix.

The dataset is split into training and test set as follows:

- Training set: The first 2,135,895 GADF matrices.
- Test set: The last 100,000 GADF matrices.

Since time sequences are constructed using a one-minute shift, they overlap, making conventional cross-validation techniques inappropriate. Instead, we adopt a chronological split, preserving the natural order of the data to avoid information leakage from the future into the past. The test set - comprising the most recent 100,000 sequences - thus serves as a realistic proxy for evaluating model performance in a forward-looking setting. This number offers a substantial and representative sample size for robust evaluation, while also ensuring that training and test data are sufficiently separated in time to minimize autocorrelation effects.

The CNN model is trained by minimizing the generalization error, optimizing both the weights in the fully connected layers and the kernel in the convolutional layers. We use the cross-entropy loss function $J(\cdot)$ for binary classification:

$$J(\mathbf{c}_h, \hat{\mathbf{y}}_h) = -\sum_{i=m}^{M} c_h^{(i)} \cdot \log(\hat{y}_{\mathrm{h}}^{(\mathrm{i})}) + (1 - c_{\mathrm{h}}^{(\mathrm{i})}) \cdot \log(1 - \hat{y}_{\mathrm{h}}^{(\mathrm{i})}).$$

The weights of the five ResNet-50 models, trained for different time horizons h, are initialized using PyTorch's default random weight initialization and updated via stochastic gradient descent. For more information on the kernel sizes of the convolutional layers and other architectural parameters of the ResNet-50 models see He et al. (2015). The training parameters are as follows: We use a learning rate of 0.001, a batch size of 64 and a momentum of 0.9. Each model is trained for a maximum of 10 epochs, but the minimum out-of-sample loss is always reached within the first five epochs. The predictive performance of each model is evaluated using in-sample (IS) and out-of-sample (OS) accuracy scores, which measures the fraction of correctly classified instances.

For further investigation, we select only the model with the best OS accuracy, as training is computationally expensive. For the same reasons, we use a subset of the data for SC. Specifically, we randomly select 50,000 GADFs from the IS dataset and generate relevance maps for these images. We then apply SC to group the relevance maps, using a mutual k-nearest neighbor graph with $k = \lceil \log(50,000) \rceil = 5$. The determination of the optimal number of clusters is described in Section 4.2. We aim to identify *substantial* clusters in terms of both size and performance. We define a cluster as substantial in size if it contains between 100 and 10,000 images. A cluster is substantial in performance if its accuracy exceeds the IS accuracy of the overall model.

To assess the robustness and generalizability of our proposed approach, we conduct four complementary robustness checks.

- First, we benchmark our method against two simpler predictive baselines. The first is a previous movement repetition model, which naïvely assumes that the next price movement will mirror the most recent one. The second is a logistic regression model. These benchmarks provide a reference point for evaluating the added value of our deep learning-based pattern recognition framework.
- Second, we examine the sensitivity of our results to the choice of time series encoding. In our main experiments, we employ GADF to transform temporal sequences into two-dimensional representations. As a robustness test, we replace GADF with Markov Transition Fields (MTF) and apply the full pipeline to assess whether our results are dependent on the specific encoding method.
- Third, we assess the importance of the explanation-based preprocessing step by omitting the LRP. Specifically, we apply SC directly to the raw CNN output vectors, thereby removing the relevance-based transformation. This allows us to evaluate whether the clustering performance depends on the interpretability mechanism.
- Finally, we test the cross-asset generalizability of our approach. While our primary experiments focus on BTC, we assess whether a model trained on BTC can be applied to FX market to predict the direction of price changes for the same out-of-sample period. These assets differ in market structure and volatility, making them suitable test cases for evaluating the robustness of the learned representations across financial time series with varying characteristics.

The results of these robustness checks are discussed in Section 4.3.

4. Results

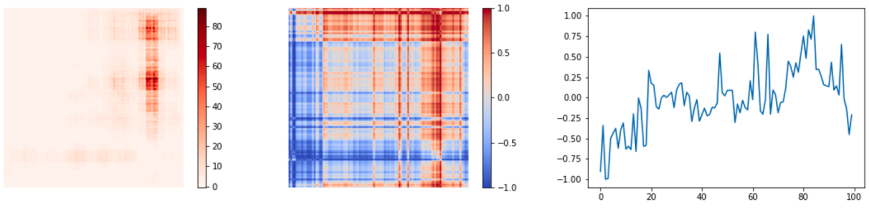
4.1. Results: Model training

We summarize the classification results in Table 2, which contains the performance measures of the trained CNNs. For each time horizon $h = 1, 10, 30, 60, 100$ the IS and the OS accuracy score, the *no-information rate* (NIR), and the p-value for the binomial tests with $H_A : Accuracy > NIR$ of the best epoch is shown. We define the best epoch as the one with the highest OS accuracy score. Predicting the price change direction in respect to the average price of the following 60 consecutive

Table 2
Forecast horizon-wise CNN accuracy and significance relative to NIR.

h	Best Epoch	IS Accuracy	IS NIR	IS p-Value	OS Accuracy	OS NIR	OS p-Value
1 min	5	0.511	0.5146	1	0.5026	0.5037	0.7517
10 min	4	0.5524	0.5463	<0.0001***	0.5047	0.5047	0.5038
30 min	2	0.5525	0.5468	<0.0001***	0.5108	0.5108	0.5013
60 min	2	0.5489	0.5445	<0.0001***	**0.5148**	0.5148	0.5013
100 min	1	0.6297	0.5425	<0.0001***	0.5	0.5167	1

Note: Five CNN models are trained on GADF matrices of past lagged price sequences to predict price change directions. The column h lists the forecasting horizon of each model. *Best Epoch* denotes the epoch that yields the best out-of-sample (OS) accuracy. The highest OS accuracy is marked in bold. Binomial tests assess whether model accuracy significantly exceeds the NIR (H_A: Accuracy > NIR); significance is indicated by p-values, with thresholds at $p < .1$ (*), $p < .05$ (**), and $p < .01$ (***).



(a) Relevance map (b) GADF (c) BTC price (normalized)

Fig. 3. Aligned visualization of input data, GADF encoding, and relevance attribution.

minutes leads to the best OS accuracy of 51.48 %, therefore it is further analyzed.

It is interesting to see, that neither the model predicting the longest time horizon of 100 minutes, nor the one predicting the shortest time horizon of 1 minute, yields the best OS accuracy, but the one predicting the 60-minute time horizon. One could imagine the effect of the information of the past prices becoming visible immediately or just over a longer time. The OS accuracy of 51.48 % is admittedly modest and does not surpass the NIR, but it still performs slightly better than random guessing. This is worth mentioning, since the standard assumption of the efficient market hypothesis is, that all public information is priced in. This way, past prices should not contain information, that can be used successfully for the prediction of future prices. By being better than random guessing, the model must have extracted some information from the past prices, that were encoded in the GADF matrices. In terms of IS prediction, all models, except for the one-minute-horizon model, predict well, which can be drawn from the statistically significant difference between IS accuracy and IS NIR. However, the critical criterion for the model evaluation is the OS accuracy. Overall, non of the models would be considered particularly good in terms of predictive performance, since the difference between OS accuracy and OS NIR is never statistically significant. However, we did not expect the model to perform much better than random guessing when predicting the complete data set, respectively the training data or test data sets. As explained before, we assume there to be patterns, which the CNN learns, for which it predicts well. Since these patterns are not present all the time, we do not expect the model to predict well for the whole data set, but only when the learned patterns that are present in the data.

4.2. Results: Spectral relevance analysis

Rather, we are interested in clustering the data set to find clusters, that yield an extraordinarily good predictive performance of the model. Since we base the clustering process on the relevance matrices obtained from applying LRP, the clusters contain observations with similar decision behavior of the CNN. In other words, the cluster formations are build based on what the CNN model *thinks* is crucial, when making a prediction for that very observation.

To better understand this process, an example for a relevance map with its associated GADF and time series sequence, is given in Fig. 3. The shading of the relevance map (a) reflects the relevance of each pixel for the decision process of the CNN. The darker the pixel is coloured, the more it contributes to the CNN's predictive decision for the associated observation. The colour gradient of the GADF map (b) runs from dark blue to dark red, with red pixels indicating high values and blue pixels low values. In the chart of the normalized BTC prices (c), the time stamps are plotted on the x-axis and the normalized prices on the y-axis. From the example in Fig. 3, it can be concluded that the most relevant part of the GADF map for the prediction is in the right corner of the image, as this is the area with the darkest pixels in the relevance map. For the illustrated observation, this is also the part in the GADF that has particularly high pixel values. Unfortunately, it is not possible to further conclude from this insight, what parts of the BTC price sequence are important.

In the process of applying SpRAy to find the aforementioned clusters, we create relevance maps using LRP for a random sample of 50,000 observations in form of GADFs. In the following step, the SC analysis is applied to the relevance maps. To determine the optimal number of clusters, we use the eigengap heuristic, which shows a gap between $\lambda_{1,298}$ and $\lambda_{1,299}$. Thus, we conclude that the optimal number of clusters in the set of relevance maps is 1298 cluster. Analyzing the clusters, we find that there is one cluster containing 8129 images (16.26 %). Furthermore, 24.11 % of the clusters contain less than 10 images. We are interested in clusters which contain similar images and which are large enough to allow conclusions about their performances. In our data set,

Table 3
Predictive performance within substantial relevance-based clusters.

Cluster	# of images	NIR	Accuracy	p-Value	Precision	Recall	F1-Score
1	122	0.5164	**0.5984**	0.0422**	0.6190	0.4407	0.5149
2	197	0.5381	**0.5838**	0.1120	0.5714	0.3956	0.4675
3	121	0.5702	0.5620	0.6100	0.4872	0.3654	0.4176
4	181	0.5304	**0.5746**	0.1319	0.5690	0.3882	0.4615
5	128	0.6016	0.5625	0.8397	0.4194	0.2549	0.3171
6	1201	0.5362	**0.5512**	0.1556	0.5230	0.3680	0.4320
7	136	0.6397	0.5735	0.9539	0.3953	0.3469	0.3696
8	171	0.5731	**0.5789**	0.4707	0.5098	0.3562	0.4194
9	139	0.6043	**0.6259**	0.3341	0.5349	0.4182	0.4694
10	109	0.5688	**0.6606**	0.0320**	0.6250	0.5319	0.5747
11	441	0.5873	0.5624	0.8669	0.4553	0.3077	0.3672
12	1245	0.5462	**0.5639**	0.1104	0.5302	0.3416	0.4155
13	147	0.5578	**0.5850**	0.2813	0.5455	0.3692	0.4404
14	109	0.5138	**0.5596**	0.1944	0.6667	0.2857	0.4000
15	179	0.5196	**0.5642**	0.1308	0.5741	0.3605	0.4429
16	315	0.6000	0.5587	0.9392	0.4253	0.2937	0.3474
17	115	0.5043	**0.5826**	0.0562*	0.6562	0.3621	0.4667
18	160	0.5438	**0.5750**	0.2379	0.5510	0.3699	0.4426
19	198	0.5505	**0.6061**	0.0663*	0.6000	0.3708	0.4583
20	266	0.5338	**0.5602**	0.2123	0.5393	0.3871	0.4507
21	120	0.6500	0.5833	0.9467	0.3947	0.3571	0.3750
22	152	0.5066	**0.5855**	0.0308**	0.6200	0.4133	0.4960
23	341	0.5279	**0.5630**	0.1060	0.5577	0.3602	0.4377
24	105	0.6381	0.6000	0.8200	0.4231	0.2895	0.3438
25	362	0.5497	**0.5525**	0.4796	0.5040	0.3865	0.4375
26	117	0.5812	**0.6496**	0.0790*	0.6333	0.3878	0.4810
27	108	0.5833	0.5648	0.7536	0.4688	0.3333	0.3896
28	122	0.5082	**0.5574**	0.1596	0.5714	0.4000	0.4706

Note: Clusters are created by applying SC to the relevance matrices of 50,000 randomly sampled GADF images. The column *Cluster* assigns an index to each substantial cluster. Each row represents a cluster with more than 100 images, qualifying it as substantial in size. A cluster is considered *substantial in performance* if its classification accuracy exceeds the in-sample (IS) accuracy benchmark of 0.5489. *# of images* lists the number of observations in each substantial cluster. The *Accuracy* column shows classification performance within each cluster, and values exceeding the corresponding No Information Rate (NIR) are highlighted in bold. p-values stem from one-sided binomial tests testing H_A: Accuracy > NIR. We follow standard thresholds for significance: $^*p < .1$, $^{**}p < .05$, $^{***}p < .01$. Performance metrics including *Precision*, *Recall*, and *F1-Score* are also reported to assess classification quality within each cluster.

we identify 28 substantial clusters, according to our prior definitions of a cluster being substantial in size and performance. As expected, most observations do not show important patterns. Instead, only 14.21 % of the images show patterns that are interesting for future price prediction. In Table 3 the number of images per cluster, the NIR, the accuracy and the p-value for the binomial test is shown for substantial clusters. Table 3 also reports precision, recall, and F1-score for each substantial cluster.

In total, 28 out of 1298 clusters exhibit accuracy scores above the IS accuracy of 0.5489. Of these, 20 clusters also surpass their respective NIR. In six cases, the difference between the observed accuracy and the NIR is statistically significant - three at the 10 % level and three at the 5 % level. The best-performing cluster achieves an accuracy of 66.06 % and includes 109 observations. Moreover, the table reports precision, recall, and F1-score for each substantial cluster. Overall, many clusters show solid precision values around 0.55-0.65, indicating that the model reliably assigns positive predictions with relatively few false positives. Although recall scores are generally lower (around 0.35-0.45), the F1-scores demonstrate a balanced performance across clusters, with several reaching values above 0.50. This suggests that the clustering approach enables the model to identify semantically coherent groups in which the classifier performs notably better than average - both in terms of correctness and consistency.

These results suggest that clustering observations based on their relevance matrices can identify subsets with markedly better predictive performance than the overall model. Each cluster can be interpreted as a group of observations that share similar relevance patterns - i.e., similar internal model reasoning. We thus identify specific relevance structures associated with higher predictive power than those found in the training or test set as a whole.

To illustrate this, Figs. 4 and 5 display the heatmaps of the relevance matrices, the corresponding GADF representations, and the normalized BTC price charts for observations in cluster 7 and cluster 14, respectively.

It is observable that within clusters the relevant pixels of the relevance maps of different observations occur in similar locations, making them look quite similar. Between clusters, the observations display very little similarity in terms of the relevance matrices. In the seventh cluster the relevant patterns appear mainly on the bottom-right side of the map and a large part of the area is very bright, symbolizing small values. In the 14th cluster the relevance maps are characterized by high values of the entire area. In both clusters, the patterns in the heatmaps of the relevance matrices are well recognizable. In terms of the GADF maps and the time series plots the observations display no obvious similarities within the clusters. By taking a closer look at the actual and the predicted values of the targets in the captions above the GADF maps of each observation in both figures, it can be observed that the (predicted) BTC price directions vary within the clusters. The good performance within the clusters is therefore not due to the model simply predicting constant upward or downward movements, which then also happen to be present in the clusters. The situation is similar in the remaining clusters. Concluding from this, the high predictive power within the clusters compared to the overall model speaks for our relevance-based clustering approach.

4.3. Results: Robustness checks

To assess the robustness and generalizability of our proposed framework, we perform four complementary analyses.

First, we compare our model to two benchmark approaches: (1) an autoregressive logistic regression model and (2) a naive movement repetition model. For comparability, we use a forecasting horizon of $h = 60$ across all models. The autoregressive logistic regression predicts - based on the past 100 values - whether the price will increase or decrease

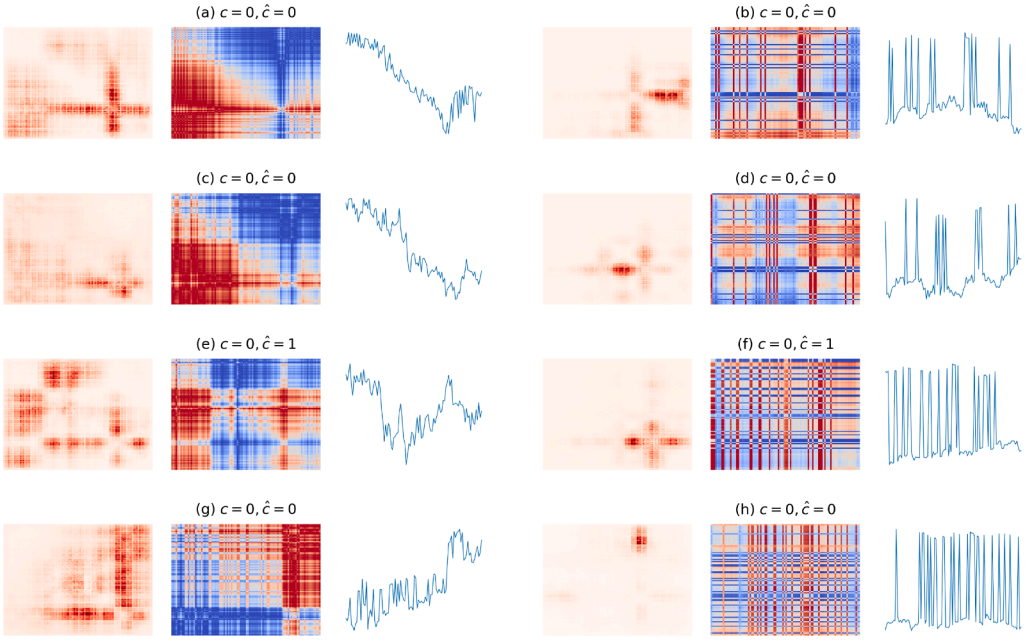


Fig. 4. Samples from Cluster 7: Relevance Maps, GADFs, and Input Data.
Note: As an illustrative example of a substantial cluster, eight observations from Cluster 7 are presented. For each observation, the corresponding relevance map, GADF image, and normalized BTC price sequence are shown. The captions above the GADF images indicate the true target value c and the predicted value $\hat{c}$ for each case.

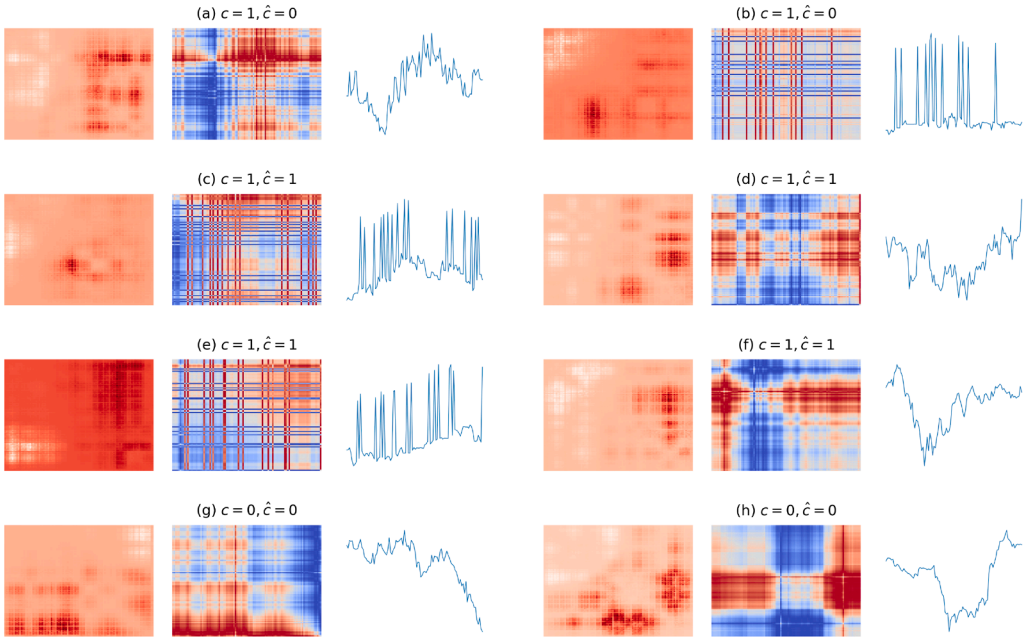


Fig. 5. Samples from Cluster 14: Relevance Maps, GADFs, and Input Data.
Note: As an illustrative example of a substantial cluster, eight observations from Cluster 14 are presented. For each observation, the corresponding relevance map, GADF image, and normalized BTC price sequence are shown. The captions above the GADF images indicate the true target value c and the predicted value $\hat{c}$ for each case.

within the subsequent 60 time steps. The movement repetition model simply predicts that the next value will follow the direction of the most recent change. We then dummy-code the prediction as 1 if the predicted direction is upward (i.e., price increases), and 0 otherwise. In contrast, our model generates predictions only for a subset of time windows-specifically those assigned to clusters that are classified as substantial. This selective strategy results in varying coverage, defined as the proportion of time windows for which a prediction is made. Consequently, standard metrics such as overall accuracy can be misleading if not interpreted alongside coverage.

Our model demonstrates substantially improved predictive accuracy compared to the two benchmark approaches, particularly in accuracy and precision (Table 4). Although it covers only 14.21 % of the time windows, it achieves statistically significant improvements in accuracy, indicating a more targeted and higher-quality prediction on selected clusters. The benchmark models' modest results highlight the inherent difficulty of forecasting financial time series. Second, to evaluate the robustness of our method regarding the choice of time series encoding, we conducted our main experiments again, using MTFs instead of GADFs. While MTF emphasizes transition dynamics rather than angular relationships, it retains key temporal structures through a fundamentally different mechanism (see Wang & Oates, 2015b). The results obtained with MTF are comparable to those using GADF across all evaluation metrics, including accuracy, precision, and coverage: The overall accuracy of the MTF-trained model is 0.5221 (GADF: 0.5148), which is slightly better than the overall model trained with GADFs ($acc = 0.5148$). 58 substantials cluster were identified, containing 15,843 observations, which is 55.14 % more observations than when using GADFs. The mean accuracy of the substantial clusters is with 0.5727 slightly - but not significantly - lower than that using GADFs ($p = .1987$, $t = 1.3076$, $df = 39.0554$). These findings suggest that our approach is robust to the choice of encoding technique, and that its predictive power does not rely on GADF-specific characteristics. Rather, it appears to generalize well across different types of time series representations.

Third, we examine the role of relevance-based explanations in the clustering procedure by applying the spectral clustering directly to the raw CNN predictions, thereby omitting the LRP step. This modification allows us to assess whether the discovered structure of the clusters is already encoded in the model outputs or whether the relevance maps are essential for identifying coherent patterns. When clustering the GADF-encoded time series without applying LRP, we observe a substantial reduction in clustering effectiveness. Specifically, the number of

Table 4
Benchmark comparison of predictive models across key metrics.

Modell	NIR	Accuracy	p-value	Precision	Recall	F1-Score	Coverage
Logistic regression	0.5152	0.5159	0.3754	0.5008	0.4529	0.4791	100 %
Previous movement	0.5152	0.5050	1	0.4799	0.4970	0.4883	100 %
Our approach	0.5600	0.5802	0.0120**	0.5347	0.3658	0.4324	14.21 %

Note: The table compares the mean predictive performance of our clustering-based approach to two benchmark models - autoregressive logistic regression and previous movement repetition - over a forecasting horizon of $h = 60$. *Accuracy* values are evaluated against the corresponding NIR, and p-values are derived from one-sided binomial tests of the null hypothesis H_0: Accuracy $\leq$ NIR, using standard significance thresholds ($^*p < .1$, $^{**}p < .05$, $^{***}p < .01$). *Precision*, *Recall*, and *F1-Score* measure classification quality, while *Coverage* indicates the percentage of observations for which a prediction was made. The results highlight the trade-off between accuracy and selectivity in our approach, which significantly outperforms both baselines despite covering only a subset of the data.

observations assigned to substantial clusters declines from 7107 to 4,999, corresponding to a 29.66 % decrease. This suggests that many time series cannot be meaningfully grouped based on raw model outputs alone. Moreover, the mean predictive accuracy of the resulting clusters decreases from 0.5802 (with LRP) to 0.5571 (without LRP). This difference is statistically significant ($p = .0319$, $t = 2.2875$, $df = 22.59$), indicating that the inclusion of relevance maps contributes not only to more stable cluster assignments but also to improved alignment with future price movements. These findings further support the conclusion that the LRP transformation plays a crucial role in extracting informative structures from the data.

Finally, we assess the generalizability of the methodology by applying it in the FX market. The FX assets differ from BTC in their volatility profiles, trading volumes, and historical development, thus providing a robust test for methodological transferability. Our framework yields an overall performance that is consistent with those obtained for Bitcoin: The number of observations assigned to substantial clusters is 6877 for FX compared to 7107 for BTC, corresponding to an increase of 3.24 %. Moreover, the mean predictive accuracy of the resulting clusters is 0.5802 for BTC and 0.5730 for FX. This difference is not statistically significant ($p = .2558$, $t = 1.1505$, $df = 47.1255$), indicating that our approach is not tailored to a single currency, but can be successfully extended to other financial time series with similar structure.

Taken together, these robustness checks demonstrate that the effectiveness of our approach hinges on the use of relevance-based feature extraction, and that the proposed method exhibits strong generalizability across different types of cryptocurrency markets.

5. Conclusion

This paper aims to explore the potential of AI-based computer vision techniques, combined with relevance-based clustering, for predicting movements in financial time series. Our research was motivated by the exceptional performance of convolutional neural networks (CNNs) in the field of computer vision - a domain traditionally distinct from finance, as financial data is rarely represented as images.

Recent advancements in time series imaging techniques, such as GADF maps, have enabled the application of well-established computer vision models to financial forecasting. Additionally, the emerging field of interpretable machine learning has introduced methods that harness the predictive power of deep learning while maintaining transparency in decision-making processes (Adadi & Berrada, 2018). Finally, clustering techniques for high-dimensional data provide a means to group images or matrices effectively.

In this study, we integrate these three areas to develop a methodology for identifying patterns associated with exceptional predictive performance. We encode BTC price time series as GADF matrices and train a ResNet-50 CNN to predict price movement direction across different time horizons. Our findings demonstrate that BTC price movements exhibit a degree of forecastability. The best-performing model achieves an out-of-sample (OS) accuracy of 0.5148 (in-sample (IS) accuracy: 0.5489) when predicting whether the average BTC price over the next 60 minutes will exceed the most recent price.

To further investigate the model's decision-making process, we generate LRP matrices for 50,000 randomly selected samples. We hypothesize that patterns exist within these relevance matrices that correspond to particularly accurate model predictions. To explore this, we apply SC, a method well-suited for high-dimensional data, to the relevance matrices. Our results reveal 29 clusters that demonstrate both exceptional forecasting accuracy and substantial sample sizes. Each of these clusters outperforms the model's overall in-sample accuracy. The best-performing cluster achieves an accuracy of 0.6606 and consists of 109 observations.

These findings suggest that a trading strategy based on cluster membership could significantly enhance performance. In this approach, a trade is executed only when the relevance matrix - corresponding to the GADF representation of the most recent 100 prices - belongs to one of the identified high-performance clusters. If the relevance matrix does not match any cluster, no trade is made. Similarity measures can be used to determine cluster membership in real-time.

Beyond demonstrating that BTC price forecasting is feasible using CNNs with GADF-encoded input, our study introduces relevance-based clustering to the domain of financial time series classification. This method identifies meaningful patterns in the relevance maps rather than in the raw price data itself. Our findings also suggest that these patterns, as exemplified by two clusters, can be visually interpreted by humans. To the best of our knowledge, this is the first study to search for patterns in relevance maps of historical price data rather than in the price data directly.

The findings of this study open several potential avenues for further research. While we focused on SpRAy, other clustering methods, such as density-based clustering, could be investigated to identify outliers and noise within high-dimensional data. The results motivate future work that incorporates additional explainability methods such as Grad-CAM or SHAP, to enhace the validation of model decisions. Moreover, testing ensemble clustering approaches and integrating macroeconomic variables into multichannel GADF matrices could provide richer representations and potentially improve predictive accuracy.

Overall, our findings highlight the great potential of relevance-based clustering for discovering predictive patterns in price data and emphasize the effectiveness of encoding time series as images for financial forecasting.

CRediT authorship contribution statement

Christine Distler: Methodology, Conceptualization, Software, Writing – original draft; **Yarema Okhrin:** Supervision, Methodology, Conceptualization, Writing – original draft, Writing – review & editing; **Jonathan Pfahler:** Methodology, Conceptualization, Software, Writing – original draft.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, *6*, 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

Anders, C. J., Weber, L., Neumann, D., Samek, W., Müller, K.-R., & Lapuschkin, S. (2022). Finding and removing clever hans: Using explanation methods to debug and improve deep models. *Information Fusion*, *77*, 261–295.

Arratia, A., & Sepúlveda, E. (2020). Convolutional neural networks, image recognition and financial time series forecasting. In V. Bitetta, I. Bordino, A. Ferretti, F. Gullo, S. Pascolutti, & G. Ponti (Eds.), *Mining data for financial applications* (pp. 60–69). Cham: Springer International Publishing.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS One*, *10*(7), e0130140. https://doi.org/10.1371/journal.pone.0130140

Barra, S., Carta, S. M., Corriga, A., Podda, A. S., & Recupero, D. R. (2020). Deep learning and time series-to-image encoding for financial forecasting. *IEEE/CAA Journal of Automatica Sinica*, *7*(3), 683–692. https://doi.org/10.1109/JAS.2020.1003132

Bose, A., Hsu, C.-H., Roy, S. S., Lee, K. C., Mohammadi-ivatloo, B., & Abimannan, S. (2021). Forecasting stock price by hybrid model of cascading multivariate adaptive regression splines and deep neural network. *Computers and Electrical Engineering*, *95*, 107405. https://doi.org/10.1016/j.compeleceng.2021.107405

Carta, S., Podda, A. S., Reforgiato Recupero, D., & Stanciu, M. M. (2022). Explainable AI for financial forecasting. In G. Nicosia, V. Ojha, E. La Malfa, G. La Malfa, G. Jansen, P. M. Pardalos, G. Giuffrida, & R. Umeton (Eds.), *Machine learning, optimization, and data science* (pp. 51–69). Cham: Springer International Publishing.

Chai, J., Zeng, H., Li, A., & Ngai, E. W. T. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, *6*, 100134. https://doi.org/10.1016/j.mlwa.2021.100134

Chen, J.-F., Chen, W.-L., Huang, C.-P., Huang, S.-H., & Chen, A.-P. (2016). Financial time-series data analysis using deep convolutional neural networks. In *7th International conference on cloud computing and big data (CCBD)* (pp. 87–92). IEEE. https://doi.org/10.1109/CCBD.2016.028

Cohen, N., Balch, T., & Veloso, M. (2020). Trading via image classification. In *Proceedings of the 1st ACM international conference on AI in finance (ICAIF '20)* (pp. 1–6). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3383455.3422543

Dourra, H., & Siy, P. (2002). Investment using technical analysis and fuzzy logic. *Fuzzy Sets and Systems*, *127*(2), 221–240. https://doi.org/10.1016/S0165-0114(01)00099-5

Gerritsen, D. F. (2016). Are chartists artists? The determinants and profitability of recommendations based on technical analysis. *International Review of Financial Analysis*, *47*, 179–196. https://doi.org/10.1016/j.irfa.2016.06.004

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. https://doi.org/10.1109/CVPR.2016.90

He, Z., Yang, J., Alroobaea, R., & Yee Por, L. (2024). SeizureLSTM: An optimal attention-based trans-LSTM network for epileptic seizure detection using optimal weighted feature integration. *Biomedical Signal Processing and Control*, *96*, 106603. https://doi.org/10.1016/j.bspc.2024.106603

Hoffmann, A. O. I., & Shefrin, H. (2014). Technical analysis and individual investors. *Journal of Economic Behavior & Organization*, *107*, 487–511. https://doi.org/10.1016/j.jebo.2014.04.002

Huang, X., Hu, Y., & Liu, H. (2023). Retracted: A hybrid fcm-cnn method to cluster and forecast financial performance of listed companies. *Journal of Intelligent & Fuzzy Systems*, *44*(2), 1991–2006. https://doi.org/10.3233/JIFS-221995

Jia, H., Ding, S., Xinzheng, X., & Nie, R. (2014). The latest research progress on spectral clustering. *Neural Computing & Applications*, *24*, 1477–1486.

Jiang, J., Kelly, B. T., & Xiu, D. (2023). Re-imagining price trends. *Chicago Booth Research Paper*, *78*(6), 3193–3249.

Ku, C. S., Xiong, J., Chen, Y.-L., Cheah, S. D., Soong, H. C., & Por, L. Y. (2023). Improving stock market predictions: An equity forecasting scanner using long short-term memory method with dynamic indicators for malaysia stock market. *Mathematics*, *11*(11), 2470. https://doi.org/10.3390/math11112470

Kumar, S. N. A., Madhavi, R., & Rajan, M. S. (2025). Explainable AI in financial forecasting using time series analysis. *International Journal for Research in Applied Science and Engineering Technology*. https://api.semanticscholar.org/CorpusID:278318649.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., & Müller, K.-R. (2019). Unmasking clever hans predictors and assessing what machines really learn. *Nature Communications*, *10*(1), 1096. https://doi.org/10.1038/s41467-019-08987-4

Li, X., Kang, Y., & Li, F. (2020). Forecasting with time series imaging. *Expert Systems with Applications*, *160*, 113680.

Miller, N., Yang, Y., Sun, B., & Zhang, G. (2019). Identification of technical analysis patterns with smoothing splines for bitcoin prices. *Journal of Applied Statistics*, *46*(12), 2289–2297. https://doi.org/10.1080/02664763.2019.1580251

Mohar, B. (1991). Eigenvalues, diameter, and mean distance in graphs. *Graphs and Combinatorics*, *7*(1), 53–64. https://doi.org/10.1007/BF01789463

Molnar, C. (2025). Interpretable machine learning. (3rd Ed.). https://christophm.github.io/interpretable-ml-book.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, *14*, 849–856.

Singh, J., & Singh, G. (2024). Deep learning for financial forecasting: Evaluating CNN and CNN-LSTM in indian stock market prediction. *Journal of Management World*, *2024*(5), 217–237. https://doi.org/10.53935/jomw.v2024i4.1070

Wang, Z., & Oates, T. (2015a). Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at the 29th AAAI conference on artificial intelligence* (pp. 1–7). (*vol. 1*).

Wang, Z., & Oates, T. (2015b). Imaging time-series to improve classification and imputation. In *24th International joint conference on artificial intelligence* (pp. 3939–3945).

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395–416. https://doi.org/10.1007/s11222-007-9033-z

Xu, Z., & Lin, H. (2023). Quantum-enhanced forecasting: Leveraging quantum gramian angular field and CNNs for stock return predictions. *ArXiv*, *abs/2310.07427*. https://api.semanticscholar.org/CorpusID:263834861.

Yang, J., Wu, Y., Yuan, Y., Xue, H., Bourouis, S., Abdel-Salam, M., Prajapat, S., & Por, L. Y. (2025). Llm-ae-mp: Web attack detection using a large language model with autoencoder and multilayer perceptron. *Expert Systems with Applications*, *274*, 126982. https://doi.org/10.1016/j.eswa.2025.126982