Gramian Angular Fields and Recurrence Plots for Volatility Prediction

Ayush Singh

April 8, 2025

Abstract

Predicting financial market volatility is a critical task for risk management, portfolio optimization, and algorithmic trading. High-frequency data, particularly Level 2 Limit Order Book (LOB) data, contains rich information about market microstructure dynamics that can be indicative of future volatility. However, effectively extracting predictive features from these complex, high-dimensional time series is challenging. This document details a methodology for encoding time series derived from LOB data into image representations using two techniques: Gramian Angular Fields (GAF) and Recurrence Plots (RP). By transforming time series into images, we can leverage the power of Convolutional Neural Networks (CNNs), which excel at capturing spatial hierarchies and patterns in image data. This document provides a self-contained, comprehensive explanation of the underlying theory, mathematical formulations, and practical steps involved, with a particular focus on the intricacies of Gramian Angular Fields. The goal is to equip readers with the foundational knowledge required to apply these techniques for volatility prediction using LOB data.

Contents

1 Introduction

Financial market volatility, typically defined as the degree of variation of a trading price series over time, is a key indicator of market risk and uncertainty. Accurate volatility prediction is crucial for various financial applications. The advent of high-frequency trading has led to the availability of vast amounts of market microstructure data, such as the Level 2 Limit Order Book (LOB), which offers a detailed snapshot of supply and demand at different price levels.

Level 2 LOB data consists of time series representing the price and volume of buy (bid) and sell (ask) orders. Analyzing these time series directly using traditional time series models can be difficult due to their high dimensionality, non-stationarity, and complex temporal dependencies.

An innovative approach is to encode these time series as images. This transformation allows the application of state-of-the-art deep learning models, specifically Convolutional Neural Networks (CNNs), which have demonstrated remarkable success in image recognition tasks. CNNs can automatically learn hierarchical features and spatial patterns within the image representation of the time series, potentially capturing complex dynamics relevant to volatility.

This document focuses on two prominent time series-to-image encoding techniques:

1. **Gramian Angular Fields (GAF):** Encodes time series information in a Gramian matrix based on a polar coordinate representation, preserving temporal dependencies.

2. **Recurrence Plots (RP):** Visualizes recurrences of states in the phase space of a dynamical system, revealing patterns related to periodicity and chaos.

We will delve into the theoretical foundations and mathematical details of both methods, paying special attention to GAF. We will then outline how these techniques can be applied to time series derived from Level 2 LOB data for the specific task of volatility prediction. This document aims to be self-sustained, providing necessary definitions and explanations for a reader with a basic understanding of machine learning concepts.

2 Background

2.1 Level 2 Limit Order Book (LOB) Data

A Limit Order Book (LOB) is a record of outstanding buy (bid) and sell (ask) orders for a specific financial instrument, organized by price level.

- **Bid Side:** Contains buy orders, sorted with the highest price (best bid) at the top.
- **Ask Side:** Contains sell orders, sorted with the lowest price (best ask) at the top.
- **Level 1 Data:** Shows only the best bid and ask prices and their associated volumes.
- **Level 2 Data:** Provides more depth, showing the prices and aggregated volumes for multiple price levels away from the best bid and ask. For example, Level 2 might show the top 10 bid prices and volumes and the top 10 ask prices and volumes.

LOB data is dynamic, updating constantly as new orders arrive, existing orders are cancelled, or orders are executed. From raw Level 2 LOB data, several informative time series can be derived, including:

- **Mid-Price:** The average of the best bid and best ask price. $P_{mid} = (P_{bid}^{(1)} + P_{ask}^{(1)})/2$. Often used as a proxy for the efficient price.
- **Spread:** The difference between the best ask and best bid price. $Spread = P_{ask}^{(1)} - P_{bid}^{(1)}$. Indicates market liquidity and transaction costs.
- **Price Levels:** Time series of prices at specific bid/ask levels (e.g., $P_{bid}^{(k)}$, $P_{ask}^{(k)}$).
- **Volume Levels:** Time series of volumes at specific bid/ask levels (e.g., $V_{bid}^{(k)}$, $V_{ask}^{(k)}$).
- **Order Flow Imbalance (OFI):** Measures the net pressure from incoming buy versus sell orders. Various definitions exist, often involving changes in price and volume at different levels. E.g., a simple version could track the difference between volume changes at the best bid and best ask.

- **Volume Order Imbalance (VOI):** Measures the imbalance based on volume changes at the best bid/ask prices relative to the spread.

These derived time series capture different aspects of market dynamics and are potential inputs for volatility prediction models.

2.2 Financial Volatility

Volatility measures the dispersion of returns for a given security or market index. It is often quantified by the standard deviation or variance of returns. In high-frequency finance, a common measure is **Realized Volatility (RV)**.

Given a time series of prices P_t sampled at high frequency (e.g., every second or minute) over a time interval T (e.g., a day or an hour), let the log-returns be $r_t = \log(P_t) - \log(P_{t-1})$. If there are N high-frequency returns within the interval T, the Realized Volatility is calculated as:

$$RV_T = \sqrt{\sum_{i=1}^{N} r_{t_i}^2} \tag{1}$$

Often, RV is annualized for comparison purposes. The goal of volatility prediction in this context is typically to predict the RV over a future interval (e.g., the next hour or day) based on past LOB data.

2.3 Motivation for Time Series Encoding as Images

Traditional time series models like ARIMA or GARCH often struggle with the high dimensionality and complex, non-linear patterns found in high-frequency financial data. Deep learning models, particularly CNNs, offer powerful capabilities for feature extraction but are primarily designed for grid-like data such as images.

Encoding a time series $S = \{s_1, s_2, ..., s_n\}$ into an image (a 2D matrix) allows us to apply CNNs directly. The key challenge is to perform this encoding in a way that preserves the essential information, especially the temporal dependencies within the series, and transforms it into a format where spatial patterns (in the image) correspond to meaningful temporal patterns (in the original series). GAF and RP are two methods designed to achieve this.

3 Gramian Angular Fields (GAF)

Gramian Angular Fields, introduced by Wang and Oates (2015), provide a method to represent a time series in a unique way by leveraging a polar coordinate system and Gramian matrix properties. The core idea is to preserve temporal correlation by encoding time series values as angular perspectives and temporal relationships as trigonometric functions between these angles.

3.1 Motivation and Core Idea

GAF aims to create a 2D representation (an image) from a 1D time series while preserving temporal order. Unlike methods that simply plot the time series or use spectrograms, GAF explicitly encodes the temporal correlation structure within the geometry of the resulting matrix. It achieves this through a multi-step process involving rescaling, transformation to polar coordinates, and calculating Gramian matrices based on trigonometric sums or differences.

3.2 Mathematical Formulation

Let $S = \{s_1, s_2, ..., s_n\}$ be a time series of length n. The process to generate GAF images involves the following steps:

Step 1: Rescaling the Time Series

The time series S is first rescaled to fall within a specific range, typically $[-1, 1]$ or $[0, 1]$. Rescaling to $[-1, 1]$ is common for GAF as it aligns well with the domain of the inverse cosine function used later. A common rescaling formula to map S to $\tilde{S} = \{\tilde{s}_1, \tilde{s}_2, ..., \tilde{s}_n\}$ in $[-1, 1]$ is:

$$\tilde{s}_i = \frac{(s_i - \max(S)) + (s_i - \min(S))}{\max(S) - \min(S)} \tag{2}$$

If $\max(S) = \min(S)$ (i.e., the series is constant), all $\tilde{s}_i$ can be set to 0 or another constant value in $[-1, 1]$. This formula ensures that values are mapped linearly to $[-1, 1]$. Note: $\tilde{s}_i = 2\frac{s_i - \min(S)}{\max(S) - \min(S)} - 1$ is an equivalent linear mapping to $[-1, 1]$.

Step 2: Transformation to Polar Coordinates

The rescaled time series $\tilde{S}$ is transformed into polar coordinates (r, ϕ).

- **Angle (ϕ):** The rescaled value $\tilde{s}_i \in [-1, 1]$ is interpreted as the cosine of an angle. The angle ϕ_i corresponding to the value $\tilde{s}_i$ is obtained using the

inverse cosine function:

$$\phi_i = \arccos(\tilde{s}_i), \quad \phi_i \in [0, \pi] \tag{3}$$

This mapping encodes the value of the time series point into an angle. Higher values (closer to 1) map to smaller angles (closer to 0), and lower values (closer to -1) map to larger angles (closer to π).

- **Radius (r):** The radius r_i represents the time stamp. It can be set simply as the time index i, or more commonly, normalized to the interval $[0, 1]$ to prevent the radius from dominating the representation:

$$r_i = \frac{t_i}{N}, \quad \text{or simply} \quad r_i = \frac{i}{n} \tag{4}$$

where t_i is the time stamp of s_i and N is a normalization factor (e.g., the time span of the series or simply n). In many practical GAF applications, the radius r is implicitly handled by the matrix structure (row/column indices relate to time) and doesn't appear explicitly in the final GAF matrix calculation itself, but the conceptual mapping to polar coordinates (r_i, ϕ_i) is key to the intuition.

This step maps the time series from a Cartesian coordinate system (time vs. value) to a polar coordinate system (time-encoded radius vs. value-encoded angle).

Step 3: Calculating the Gramian Angular Fields

The core of GAF lies in creating an $n \times n$ matrix where each element (i, j) represents a relationship between the points at time i and time j, based on their angles ϕ_i and ϕ_j. This is analogous to a Gramian matrix $G = X^T X$ where X contains vectors, measuring the inner products between them. Here, we use trigonometric functions of the angles.

There are two main types of GAF:

1. Gramian Angular Summation Field (GASF): The GASF matrix is defined by considering the cosine of the sum of angles:

$$\text{GASF}_{i,j} = \cos(\phi_i + \phi_j) = \cos(\arccos(\tilde{s}_i) + \arccos(\tilde{s}_j)) \tag{5}$$

Using the trigonometric identity $\cos(a + b) = \cos(a)\cos(b) - \sin(a)\sin(b)$, and knowing $\cos(\phi_i) = \tilde{s}_i$ and $\sin(\phi_i) = \sqrt{1 - \cos^2(\phi_i)} = \sqrt{1 - \tilde{s}_i^2}$ (since $\phi_i \in [0, \pi]$, $\sin(\phi_i) \geq 0$), we get:

$$\text{GASF}_{i,j} = \tilde{s}_i \tilde{s}_j - \sqrt{1 - \tilde{s}_i^2}\sqrt{1 - \tilde{s}_j^2} \tag{6}$$

The GASF matrix is symmetric ($\text{GASF}_{i,j} = \text{GASF}_{j,i}$). It captures the correlation structure based on angle sums. The main diagonal $\text{GASF}_{i,i} = \cos(2\phi_i) = \cos(2\arccos(\tilde{s}_i))$ contains information about the original value $\tilde{s}_i$. Using the identity $\cos(2a) = 2\cos^2(a) - 1$, we have $\text{GASF}_{i,i} = 2\tilde{s}_i^2 - 1$.

2. Gramian Angular Difference Field (GADF): The GADF matrix is defined by considering the sine (or sometimes cosine) of the difference of angles. The original formulation uses sine:

$$\text{GADF}_{i,j} = \sin(\phi_i - \phi_j) = \sin(\arccos(\tilde{s}_i) - \arccos(\tilde{s}_j)) \tag{7}$$

Using the trigonometric identity $\sin(a - b) = \sin(a)\cos(b) - \cos(a)\sin(b)$, we get:

$$\text{GADF}_{i,j} = \sqrt{1 - \tilde{s}_i^2}\tilde{s}_j - \tilde{s}_i\sqrt{1 - \tilde{s}_j^2} \tag{8}$$

The GADF matrix defined using sine is anti-symmetric ($\text{GADF}_{i,j} = -\text{GADF}_{j,i}$). It captures correlation based on angle differences. The main diagonal $\text{GADF}_{i,i} = \sin(\phi_i - \phi_i) = \sin(0) = 0$.

Alternative GADF Definition: Some implementations might use $\cos(\phi_i - \phi_j)$ instead of $\sin(\phi_i - \phi_j)$:

$$\text{GADF}'_{i,j} = \cos(\phi_i - \phi_j) = \tilde{s}_i\tilde{s}_j + \sqrt{1 - \tilde{s}_i^2}\sqrt{1 - \tilde{s}_j^2} \tag{9}$$

This version is symmetric and the diagonal is $\text{GADF}'_{i,i} = \cos(0) = 1$. It captures similar information but results in a different matrix structure. Throughout this document, we will refer to the original sine-based GADF unless otherwise specified.

The resulting GASF and GADF matrices are $n \times n$ square matrices where n is the length of the input time series window. These matrices can be directly visualized or used as input images for CNNs.

3.3 Properties of GAF

- **Bijectivity:** The transformation from the rescaled time series $\tilde{S}$ to the polar coordinates (ϕ_i) using $\phi_i = \arccos(\tilde{s}_i)$ is bijective because $\arccos$ maps $[-1, 1]$ uniquely to $[0, \pi]$. The GAF matrix itself (GASF or GADF) contains information about all pairwise combinations $\cos(\phi_i + \phi_j)$ or $\sin(\phi_i - \phi_j)$. While the mapping from the time series to the GAF matrix is unique, reconstructing the exact original time series *solely* from the GAF matrix can

be complex. The main diagonal of GASF ($2\tilde{s}_i^2 - 1$) allows recovery of the magnitude $|\tilde{s}_i|$ but not the sign if the original range was $[-1, 1]$. However, the off-diagonal elements provide relational information that helps disambiguate. Wang & Oates (2015) argue that the mapping is bijective "up to the sign of the actual time series values" when considering the diagonal alone, but the full matrix largely preserves the original information content and temporal ordering. The choice of rescaling (e.g., to $[0, 1]$ vs $[-1, 1]$) affects the exact reconstruction properties. If rescaling is to $[0, 1]$, $\arccos$ maps to $[0, \pi/2]$, and $\text{GASF}_{i,i}$ allows full recovery.

- **Preservation of Temporal Dependency:** GAF inherently encodes time dependency. The position (i, j) in the matrix directly relates to time steps t_i and t_j. The value $\text{GAF}_{i,j}$ reflects the relationship (based on value/angle) between these two time points. Moving away from the main diagonal corresponds to increasing the time gap $|i - j|$. Temporal correlations are thus mapped into spatial patterns in the GAF image.

- **Matrix Size and Computation:** The GAF matrix size is $n \times n$, where n is the length of the time series segment. For very long time series, this can result in large images. Often, a sliding window approach is used, generating smaller GAF images for overlapping or non-overlapping segments of the original series. The computation involves $O(n^2)$ trigonometric operations.

- **Stability:** The use of $\arccos$ and trigonometric functions makes the transformation robust to small perturbations in the input time series, provided the rescaling handles outliers appropriately.

- **Symmetry/Anti-symmetry:** GASF is symmetric, GADF (sine version) is anti-symmetric. This structure can be exploited by certain neural network architectures if needed.

3.4 Interpretation of GAF Images

Patterns in GASF/GADF images correspond to structures in the time series:

- **Main Diagonal (GASF):** Represents the value at each time point (transformed as $2\tilde{s}_i^2 - 1$). Constant values appear as constant diagonal entries.

- **Main Diagonal (GADF):** Is always zero (for the sine version).

- **Off-Diagonal Elements:** Represent the relationship between pairs of time points (i, j). High correlation or similarity in values (after mapping to angles) results in specific patterns. For example, periodic patterns in the time series might lead to repetitive textures or structures in the GAF image. Sharp changes or spikes in the time series will create distinct lines or blocks.
- **Texture:** The overall texture provides a visual signature of the time series' dynamics. Smooth time series yield smooth GAF textures, while volatile or noisy series result in more complex, high-frequency textures.

Visualizing GASF and GADF images (e.g., using heatmaps) can provide qualitative insights into the time series characteristics.

4 Recurrence Plots (RP)

Recurrence Plots, introduced by Eckmann, Kamphorst, and Ruelle (1987), are a graphical tool primarily used in dynamical systems theory to visualize the recurrences of states in a phase space trajectory. They reveal patterns related to periodicity, chaos, and non-stationarity in time series.

4.1 Motivation and Core Idea

Many systems, including financial markets, can be viewed as complex dynamical systems. The state of the system evolves over time in a high-dimensional space called the phase space. A time series often represents a projection of this trajectory onto a single observable dimension. RP aims to visualize when the system revisits previously occupied regions of the phase space. If a state at time i is very close to a state at time j, it indicates a recurrence. Plotting these recurrences creates an image revealing the system's temporal structure.

4.2 Mathematical Formulation

Generating an RP from a time series $S = \{s_1, s_2, ..., s_n\}$ involves these steps:

Step 1: Phase Space Reconstruction (Time-Delay Embedding)

Since we usually only observe a single variable s_t, we need to reconstruct the higher-dimensional phase space trajectory. Takens' Embedding Theorem provides

the theoretical basis for this. We create m-dimensional state vectors $\mathbf{x}_i$ using time-delayed copies of the original series:

$$\mathbf{x}_i = (s_i, s_{i+\tau}, s_{i+2\tau}, ..., s_{i+(m-1)\tau}) \tag{10}$$

where:

- m is the **embedding dimension**: The dimension of the reconstructed phase space. It should be large enough to unfold the attractor of the dynamical system (typically $m > 2D$, where D is the fractal dimension of the attractor).
- τ is the **time delay**: The lag used between successive components of the state vector. It should be chosen such that s_i and $s_{i+\tau}$ are sufficiently independent but not completely uncorrelated. Common methods for choosing m and τ include the False Nearest Neighbors (FNN) algorithm and the Average Mutual Information (AMI) function, respectively.

This process yields a trajectory of state vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, where $N = n - (m-1)\tau$.

Step 2: Recurrence Matrix Calculation

An $N \times N$ Recurrence Matrix $\mathbf{R}$ is computed, where each element $R_{i,j}$ indicates whether the state $\mathbf{x}_j$ is close to the state $\mathbf{x}_i$.

$$R_{i,j} = \Theta(\epsilon - \|\mathbf{x}_i - \mathbf{x}_j\|), \quad i, j = 1, ..., N \tag{11}$$

where:

- $\|\cdot\|$ is a norm function, typically the Euclidean norm (L_2), but sometimes the Maximum norm (L_∞) or Manhattan norm (L_1) is used.
- ϵ is a **threshold** or radius: A predefined distance. If the distance between states $\mathbf{x}_i$ and $\mathbf{x}_j$ is less than or equal to ϵ, they are considered recurrent ($R_{i,j} = 1$), otherwise not ($R_{i,j} = 0$).
- $\Theta(\cdot)$ is the Heaviside step function: $\Theta(x) = 1$ if $x \geq 0$, and $\Theta(x) = 0$ if $x < 0$.

The resulting matrix $\mathbf{R}$ is binary (containing 0s and 1s) and symmetric, with $R_{i,i} = 1$ along the main diagonal (since $\|\mathbf{x}_i - \mathbf{x}_i\| = 0 \leq \epsilon$).

Step 3: Threshold Selection (ϵ)

Choosing ϵ is crucial.

- **Fixed Threshold:** A constant value is chosen. If ϵ is too small, the RP will be sparse with few recurrence points. If too large, almost all points will be marked as recurrent, obscuring the structure. A common rule of thumb is to choose ϵ such that the recurrence rate (percentage of $R_{i,j} = 1$ pairs) is small, e.g., a few percent, or related to the standard deviation of the time series.

- **Fixed Recurrence Rate / Nearest Neighbors:** Instead of fixing ϵ, one can fix the number of neighbors k for each $\mathbf{x}_i$ that are considered recurrent. This leads to an adaptive threshold ϵ_i for each row, ensuring a constant density of recurrence points. This is often preferred as it is less sensitive to the data's distribution and scale.

4.3 Interpretation of Recurrence Plots

The Recurrence Plot is typically visualized by plotting a black dot at position (i, j) if $R_{i,j} = 1$ and leaving it white otherwise. Patterns in the RP reveal characteristics of the system's dynamics:

- **Homogeneous/Random Dots:** Suggests stochastic or high-dimensional chaotic behavior.

- **Diagonal Lines ($R_{i+k,j+k} = 1$):** Indicate that segments of the trajectory run parallel to each other. The length of these lines relates to the duration of similar local behavior (determinism). Lines parallel to the main diagonal represent periodic behavior.

- **Vertical/Horizontal Lines ($R_{i,j:j+k} = 1$ or $R_{i:i+k,j} = 1$):** Indicate that a state persists for some time or changes slowly (laminarity).

- **Blocks/Rectangles:** Show periods where the trajectory remains trapped in a specific region of the phase space.

- **Overall Texture:** Can distinguish between different dynamical regimes (e.g., periodic, chaotic, random). Fading borders or changing textures can indicate non-stationarity.

Recurrence Quantification Analysis (RQA) involves computing quantitative measures from the RP (like recurrence rate, determinism, laminarity, entropy) to characterize the dynamics numerically.

5 Application to LOB Data for Volatility Prediction

Here we outline the pipeline for using GAF and RP images derived from LOB data to predict future volatility using CNNs.

1. Data Preprocessing:

- **Cleaning:** Handle potential anomalies or errors in the raw LOB data feeds.
- **Time Synchronization:** Ensure consistent timing across different data points (e.g., using event time or fixed sampling intervals).
- **Feature Extraction:** Derive relevant time series from the LOB data. Examples:
  - Mid-price returns: $r_t = \log(P_{mid,t}) - \log(P_{mid,t-1})$. Log returns are often preferred due to better statistical properties.
  - Order Flow Imbalance (OFI) or Volume Order Imbalance (VOI).
  - Spread changes.
  - Log-volumes at specific levels.

  The choice of time series depends on the hypothesis about which factors drive volatility. Multiple time series can be used.
- **Normalization/Standardization:** Before encoding, especially for GAF which requires rescaling, normalize or standardize the chosen time series. For RP, normalization might be done before phase space reconstruction.

2. Windowing:

- Divide the long input time series into shorter, fixed-length windows (segments). Let the window size be n. This size n will determine the dimension of the GAF/RP images ($n \times n$ or $N \times N$ for RP).
- Choose whether the windows should be overlapping or non-overlapping. Overlapping windows provide more training samples but introduce redundancy.
- For each window k, representing time interval $[T_k, T_k + \Delta T]$, we will generate an image (or multiple images).

3. Time Series Encoding (Image Generation):

- For each time series window S_k:
  - **GAF Encoding:**
    1. Rescale S_k to $[-1, 1]$ (or $[0, 1]$).
    2. Compute angles $\phi_i = \arccos(\tilde{s}_i)$.
    3. Calculate the $n \times n$ GASF matrix using Eq. (5).
    4. Calculate the $n \times n$ GADF matrix using Eq. (7).

    This yields two images (GASF and GADF) per time series window.
  - **RP Encoding:**
    1. Choose embedding parameters m and τ.
    2. Reconstruct the phase space trajectory $\mathbf{X}_k$ using Eq. (10). Let the trajectory length be $N = n - (m - 1)\tau$.
    3. Choose a thresholding method (fixed ϵ or fixed neighbors).
    4. Calculate the $N \times N$ Recurrence Matrix $\mathbf{R}_k$ using Eq. (11).

    This yields one binary image per time series window. Note that the RP image size $N \times N$ might be different from the GAF image size $n \times n$. Resizing might be necessary if combining them or feeding to a fixed-input CNN.
- **Multi-Channel Images:** If using multiple input time series (e.g., mid-price returns and OFI) or both GASF and GADF, these can be stacked along the channel dimension to create multi-channel input images for the CNN (similar to RGB channels in standard images). For example, one could create a 3-channel image: Channel 1=GASF(mid-price), Channel 2=GADF(mid-price), Channel 3=RP(OFI).

4. Volatility Target Calculation:

- For each input window k (ending at time $T_k + \Delta T$), define the target volatility. This is typically the realized volatility (RV) calculated over a subsequent future time interval, e.g., $[T_k + \Delta T + \delta, T_k + \Delta T + \delta + H]$, where δ is a small gap (optional) and H is the prediction horizon.
- Calculate the target RV_k using high-frequency returns within the horizon H.
- The prediction task can be regression (predicting the continuous RV value) or classification (predicting volatility regimes, e.g., high/low volatility).

5. CNN Model Training and Prediction:

- **Model Architecture:** Design a CNN suitable for the generated image size and task. Typical architectures include:
  - Input Layer: Accepts images of size (height, width, channels). Height=Width=n for GAF, N for RP (or resized dimensions). Channels depend on how many encodings/series are combined.
  - Convolutional Layers: Apply filters to detect spatial patterns (corresponding to temporal patterns). Use activation functions like ReLU.
  - Pooling Layers (e.g., MaxPooling): Reduce dimensionality and provide translation invariance.
  - Flatten Layer: Convert the 2D feature maps into a 1D vector.
  - Dense (Fully Connected) Layers: Perform classification or regression based on the extracted features.
  - Output Layer: Single neuron with linear activation for regression (predicting RV), or multiple neurons with softmax activation for classification.
- **Training:**
  - Split the dataset (image-volatility pairs) into training, validation, and test sets, ensuring chronological order is respected to avoid lookahead bias.
  - Choose a loss function (e.g., Mean Squared Error (MSE) for regression, Cross-Entropy for classification).
  - Choose an optimizer (e.g., Adam, SGD).
  - Train the model on the training set, tune hyperparameters (learning rate, batch size, network architecture, GAF/RP parameters like m, τ, ϵ) using the validation set.
- **Evaluation:** Evaluate the final model performance on the held-out test set using appropriate metrics (e.g., RMSE, MAE for regression; Accuracy, F1-score for classification).
- **Prediction:** Use the trained model to predict future volatility based on new incoming LOB data encoded as images.

6 Conclusion

Encoding time series derived from Level 2 LOB data as images using Gramian Angular Fields (GAF) and Recurrence Plots (RP) offers a powerful approach to leverage Convolutional Neural Networks for financial volatility prediction. GAF transforms time series into a polar coordinate representation and uses trigonometric functions to create images that preserve temporal correlations, with GASF and GADF capturing complementary aspects of these correlations. RP visualizes the recurrence dynamics of the system in phase space, revealing patterns related to periodicity and determinism.

This document has provided a detailed theoretical and mathematical overview of these methods, emphasizing the intricacies of GAF, and outlined a practical pipeline for their application. By converting complex temporal patterns into spatial patterns within images, these techniques allow CNNs to automatically learn relevant features predictive of future volatility from rich LOB data.

Challenges remain, including the optimal selection of input time series, window size, GAF/RP parameters (m, τ, ϵ), and CNN architecture. However, the potential to capture non-linear dynamics missed by traditional models makes this image-based encoding approach a promising avenue for advancing high-frequency volatility prediction.