Time Sentence

Mohsen Larni
Department of Computer Science
University of Nevada, Las Vegas
USA
mohsen.larni@unlv.edu

***Abstract*—Understanding system dynamics and behaviors requires the discovery of meaningful, recurring patterns or "motifs" in time series data. In this project, "Time Sentence," I introduce a pipeline for the unsupervised discovery of a vocabulary of shape-based motifs from general time series, tested on high-frequency cryptocurrency (BTC-USDT). My methodology includes (I) time series preprocessing, including log-transformation and Z-score normalization, (II) extraction of fixed-length subsequences, (III) learning 5-dimensional latent representations using a Variational Autoencoder (VAE) with an experimental DTW-based triplet loss aimed to encourage shape-awareness, and (IV) constructing a k-Nearest Neighbor (k-NN) similarity graph based on the closeness of these embeddings in the Euclidean space, followed by Leiden community detection to identify motif candidates. While the VAE produced a structured latent space, my results indicated the DTW-based triplet loss component remained largely inactive in its initial configuration, suggesting the VAE's other objectives already separated the selected triplets. The Leiden algorithm identified 8,151 communities from the k-NN graph ($k = 50$) of VAE embeddings, with 321 communities containing at least 100 members considered primary motif candidates. Visual inspection of these candidates shows varying degrees of internal shape coherence. This work highlights the challenges in effectively using domain-specific similarity metrics like DTW into VAE training and the complexities of hyperparameter tuning in such unsupervised multi-stage pipelines.**

***Index Terms*—Time Series Motif Discovery, Variational Autoencoder, DTW-Based Triplet Loss, k-Nearest Neighbor Similarity Graph, Leiden Community Detection**

I. Introduction

Across many fields, from biology and engineering to economics and finance, we encounter data that unfold over time. This "time series" data, whether it is from a heart monitor, a factory sensor, or the stock market, often contains repeating patterns, namely "motifs". Finding these motifs is like finding the key building blocks of a complex sequence; it helps us understand how a system behaves, predict what it might do next, or spot when something unusual is happening. However, current methods for finding these patterns can be limited.

This project presents a new way to automatically discover these fundamental patterns in any kind of time series. Its idea was inspired by how language works: just as sentences are built from words, I think of a time series as being built from a set of basic, recurring "shape-based words", or motifs. The main difference is that, unlike words in a sentence which have whitespaces between them, these motifs in a time series are often hidden within a continuous flow of data. To develop and test my approach, I decided to work with high-frequency financial data from the cryptocurrency market, which is known for its complexity and rapid changes, making it a good challenge for my methods.

A. Motivation

The conceptual foundation of this project is inspired by the profound connection between human consciousness and language, a primary differentiator between humans and other species. We observe that language is not merely a tool for communication, but the very framework through which we often describe, categorize, and comprehend the objects and subjects surrounding us. We instinctively seek to parse complex information into understandable units, akin to words, to make sense of the larger narrative. This project explores whether such a "linguistic" approach can be formalized to deconstruct and understand general time series data by identifying its fundamental, recurring "shape-based words" or motifs.

B. Problem Statement

This project, titled "Time Sentence," addresses the challenge of unsupervised discovery of a fundamental and meaningful vocabulary of shape-based patterns, termed "motifs", from general time series data. The core idea is to conceptualize a time series as a sentence in which these motifs act as the constituent "words." The primary objective is to identify a minimal dictionary of such motifs that, through sequential concatenation, can effectively reconstruct or represent the original time series. Although the methodology aims for broad applicability, this study utilizes financial time series, specifically 1-minute price data from cryptocurrency pairs (e.g., BTC-USDT, ETH-USDT), as a rich and complex domain for development and demonstration.

A central hypothesis is that as the diversity and volume of analyzed time series increase (for example, by incorporating more cryptocurrency pairs or longer data histories), the set of newly extracted unique motifs should converge to a finite number or show a significantly decreasing rate of discovery (similar to a logarithmic trend). This convergence would imply the identification of a core "alphabet" of underlying patterns.

The scope of this current work focuses on the extraction of these motifs by identifying communities of similarly-shaped subsequences. This is achieved by first generating low-dimensional embeddings of fixed-length subsequences from the normalized time series. Subsequently, a similarity graph is constructed where these embeddings serve as nodes, and edges

connect nodes (embeddings) that are deemed similar based on their proximity in the embedding space (i.e. neighbors). Finally, community detection algorithms are applied to this graph to isolate densely connected groups of embeddings, which correspond to the candidate motifs.

II. Background

Time series motif discovery has been extensively studied as a primitive for tasks like classification, clustering, and summarization. Mueen et al.'s algorithm uses efficient pruning and early abandonment strategies to find highly similar subsequences ("motifs") in large datasets, resulting in speedups when compared to brute force search [1]. Dynamic Time Warping (DTW) is a key component of many motif methods for detecting shape-based similarities. Berndt et al. showed that DTW's dynamic programming alignment can match patterns despite temporal distortions, enabling pattern detection in various domains [2].

Although traditional motif algorithms use fixed distance measures and window lengths, deep representation learning provides a more expressive alternative. The Variational Autoencoder framework optimizes a variational bound on data likelihood to learn continuous latent embeddings. This allows for compact and informative representation of high-dimensional inputs [3]. In addition to unsupervised embeddings, supervised metric-learning techniques, such as Schroff et al.'s FaceNet system's triplet loss, improve the discriminative power of learned features by bringing semantically similar inputs closer together in the embedding space [4].

Embedding subsequences in a graph structure enables community-detection methods to identify motifs without predetermined cluster counts. Traag et al. introduced the Leiden algorithm, which improves the Louvain method by ensuring well-connected, high-quality communities through improved local-move heuristics and faster convergence. This makes it ideal for clustering and embedding graphs at scale [5]. Modern frameworks combine shape-aware metrics, deep latent models, and robust graph clustering to achieve unsupervised, scalable motif discovery in complex time series data.

III. Methodology

The primary goal of this "Time Sentence" project is to create a systematic pipeline for identifying fundamental shape-based patterns, or "motifs," within general time series data. My approach includes several stages: (I) gathering and preparing data, (II) transforming data segments into more useful formats (embeddings), and (III) analyzing these transformed segments to identify groups that represent distinct motifs. Although the framework is intended to be widely applicable, its components are demonstrated with 1-minute closing price data from cryptocurrency markets, resulting in a univariate time series of price over time.

A. Technical Challenges

Before digging the pipeline, it is important to recognize the technical challenges that this project will pose. It is worth mentioning that the primary challenge of this idea is to mathematically define a "minimal dictionary" of motifs capable of reconstructing the original series. Evaluating the success and correctness of discovered motifs is also challenging, particularly in an unsupervised setting. I did not focus on solving this challenge since it is out of the scope of this project.

Time series data, especially financial data, can be extremely noisy and appear random, making pattern extraction challenging. The core task is similar to trying to find words in a sentence without spaces or clearly defined characters. Furthermore, several pipeline parameters, such as the length of the subsequence (window size), the extraction step size (stride used to reduce the effect of overlapping windows), and similarity grouping parameters (such as $'k'$ in $k - NN$, which will be discussed later in this report), must be carefully considered and tuned.

Another key challenge I sought to address was moving beyond simple Euclidean distance for similarity, which frequently fails to capture the true shape of time series segments; this led to investigation of Dynamic Time Warping (DTW) concepts in the embedding process.

Aside from all of this, the pipeline design should be such that it can handle computations on a large number of data points and extracted subsequences of time series data. For example, using $O(N^2)$ operations would be computationally expensive.

B. Data Collection

The initial dataset for this study is composed of high-frequency financial time series, specifically 1-minute interval data for the cryptocurrency pair BTC-USDT. Data for other pairs, such as ETH-USDT and LTC-USDT, were also collected but not used in this project. This data was gathered by scraping publicly available historical k-line (candlestick) data from *Binance Public Data project*[1]. I relied on the 1-minute timeframe to ensure an adequate volume of data points for the mining pattern. A script was developed to systematically download and organize these daily k-line files for the selected pairs, covering a period from January 2018 to the near present (e.g., March 2025, as indicated in initial data collection efforts). An important hypothesis guiding my data collection strategy is that as I analyze more data from an increasing number of different pairs, the set of newly discovered unique motifs should eventually converge, or at the very least, the rate of new motif discovery should slow significantly, indicating that I am on the right track. Unfortunately, this hypothesis was not taken into account because I only used time series data from one pair.

C. Data Preprocessing

After collecting the raw k-line data, which includes OHLCV features (Open, High, Low, and Close prices, and Volume), several preprocessing steps were performed to prepare it for

[1]https://data.binance.vision/?prefix=data/spot/daily/klines/

a proper and scale-free comparison of the subsequence values and shapes. Initially, I focused on the 'Close' price and its corresponding 'Close_Time'.

1) Basic Cleaning and Timestamp Handling: The raw data was aggregated and timestamps in various formats (milliseconds or microseconds) were standardized into a single datetime format. Fortunately, my initial raw data contained no missing values (NaNs). Figure 1 shows the close price of BTC-USDT in 1-minute interval, spanning from January 2018 to March 2025, prior to any transformation and normalization, with absolute values.

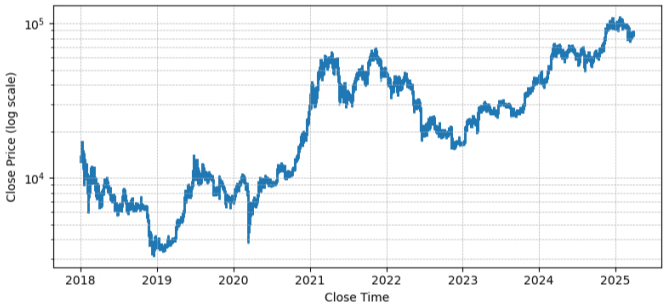

Fig. 1. BTC-USDT Close Price over Close Time, 1-minute timeframe

2) Log Transformation and Standardization: To stabilize the variance and work with data that are less sensitive to absolute price levels, the 'Close' price series was first log-tranformed. The log-transformed prices were then standardized using Z-score normalization method. This rescales the data so that the mean is zero and the standard deviation is one, making subsequent distance and similarity calculations more comparable and improving the stability of neural network training for embeddings. Figure 2 shows the close price of BTC-USDT over time, just like Figure 1, but after log-transformation and standardization. By comparing Figures 1 and 2, it is observable that the shape of the whole plot has been preserved, even after the preprocessings.

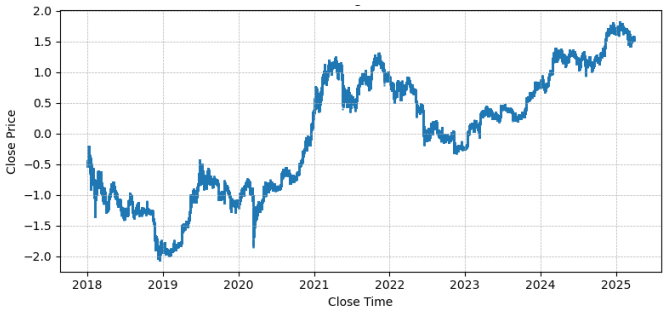

Fig. 2. BTC-USDT Z-score Close Price over Close Time, 1-minute timeframe

3) Subsequence Extraction: I used a sliding window approach to extract fixed-length subsequences from the standardized 'Close' price time series. For the current phase, a window size of 20 data points (representing 20 minutes) was chosen. To reduce redundancy and computational load caused by highly overlapping windows, a stride of 5 data points was used, which means that the window moved forward 5 data points at a time to extract the next subsequence. Each of these 20-point subsequences is a potential motif.

D. Embedding

To transform the 20-dimensional subsequences (where 20 is the fixed size of the sliding window for subsequence extraction) into a lower-dimensional space where similarities can be computed and analyzed more efficiently, I used an embedding technique. The goal of this embedding is not just dimensionality reduction, but to create representations where subsequences with similar shapes are located close to each other.

Initially, a standard Autoencoder (AE) was trained to compress each 20-point subsequence into a 5-dimensional embedding vector. However, standard AEs primarily optimize for reconstruction [6] and may not inherently learn an embedding space that emphasizes shape similarity in a way that aligns with metrics like DTW.

To address this, I evolved my approach towards a Variational Autoencoder (VAE). The VAE architecture consists of:

- An **encoder** that maps an input subsequence to the parameters (mean μ and log-variance $\log(\sigma^2)$ of a probability distribution in the 5-dimensional latent space.
- A reparameterization trick to **sample from this distribution** in a differentiable way. In a VAE, the encoder does not simply output a single point for embedding (as a standard Autoencoder does). Instead, it learns how to describe each input's probability distribution in the latent (embedding) space. The encoder typically outputs the mean (μ) and log-variance ($\log(\sigma^2)$) for a Gaussian distribution. To generate an embedding vector z for the decoder, a point from the learned distribution $N(\mu, \sigma^2)$ must be sampled. However, sampling is a random process that is generally not differentiable (i.e. gradients cannot be calculated or used to train the encoder using standard backpropagation). The gradients must flow from the decoder's reconstruction loss (and KLD loss) back through the sampled z to update the encoder's parameters (μ and $\log(\sigma^2)$. Thus, to make the sampling process differentiable. Instead of directly sampling z from $N(\mu, \sigma^2)$:
  1) Sample a random noise vector ϵ from a fixed, standard normal distribution (mean 0, variance 1), i.e., $\epsilon \sim N(0, I)$.
  2) Compute the latent vector z as $z = \mu + \sigma + \epsilon$.
- A **decoder** that attempts to reconstruct the original 20-point subsequence from the sampled latent vector.

The VAE is trained by minimizing a combined loss function which includes:

1) **Reconstruction Loss:** Typically, Mean Squared Error (MSE) between the original and the reconstructed subsequence.
2) **KL Divergence (KLD) Loss:** This regularizes the latent space, encouraging the learned distributions to be close

to a standard normal distribution, resulting in a more continuous and structured embedding space.

3) **DTW-Based Triplet Loss:** To explicitly incentivize the VAE to learn shape-aware embeddings, I introduced a triplet loss term. This loss aims to ensure that if two original subsequences are similar according to DTW, their latent representations (μ vectors) are closer together in Euclidean distance than a subsequence that is dissimilar by DTW. This involves mining triplets (anchor, positive, negative) within each training batch based on their pairwise DTW distances (calculated on the original 20-dimensional subsequences) and then applying the triplet loss to their corresponding latent embeddings. Triplet loss is a common practice in metric learning and representation learning to teach a model about relative similarities. Instead of just looking at pairs, it looks at triplets of data points:
   - *Anchor (A):* A reference data point (in this case, an original subsequence x_A and its embedding z_A).
   - *Positive (P):* A data point that is similar to the anchor (original subsequence x_P is DTW-similar to x_A; I want its embedding z_P to be close to z_A).
   - *Negative (N):* A data point that is dissimilar to the anchor (original subsequence x_N is DTW-dissimilar to x_A; I want its embedding z_N to be far from z_A).

   Finally, its loss function is formulated as:

$$L = max(0, D(z_A, z_P)^2 - D(z_A, z_N)^2 + margin),$$

   where $D(z_i, z_j)^2$ is the squared Euclidean distance between z_i and z_j, and $margin$ is a small positive value. In other words, the distance between the anchor and the positive should be smaller than the distance between the anchor and the negative by at least $margin$. If this condition is already met, the $max(0, \cdots)$ part makes the loss for that triplet zero, so the model will not be penalized. If the condition is violated (e.g., the negative is too close or the positive is too far), the loss becomes positive, and the model's weights are adjusted to satisfy the condition.

This stage outputs a 5-dimensional embedding vector for each subsequence, which is used as input in the subsequent community detection phase.

E. Community Detection

After representing all subsequences as 5-dimensional embeddings, the next step is to group them according to similarity to find potential motifs. Instead of traditional clustering methods that may require pre-specifying the number of clusters (such as K-Means), which is unknown, or density-based clustering methods that are time-consuming and computationally expensive (such as DBSCAN), I chose a graph-based community detection approach.

1) k-Nearest Neighbor (k-NN) Similarity Graph Construction: The nodes in my graph are the individual subsequence embeddings. I define edges using the *k-Nearest Neighbors* algorithm. The Euclidean distance is used to find k closest neighbors of each embedding in the 5-dimensional embedding space. In my experiments, I set k equal to 50. Then, an edge is created between an embedding and each of its k neighbors. The edges are weighted with a Gaussian kernel based on the Euclidean distance between connected embeddings:

$$w_{ij} = exp(\frac{-D_E(z_i, z_j)^2}{2 \cdot \sigma^2}),$$

where $D_E(z_i, z_j$ is the Euclidean distance between embeddings z_i and z_j, and σ is a bandwidth parameter, calculated as the mean distance to the k-th neighbor across all embeddings. This method assigns higher weights to closer (more similar) embeddings. Finally, the graph is symmetrized to become undirected.

2) Leiden Community Detection: On the weighted, undirected similarity graph, I use the Leiden algorithm. This algorithm is designed to partition the graph into communities (groups of nodes) so that nodes within a community are densely interconnected (that is, very similar to each other), while connections between communities are sparse [5]. The Leiden algorithm optimizes a quality function (in this case, modularity) and does not require the number of communities to be specified beforehand. Each community identified by the Leiden algorithm represents a cluster of subsequences with similar embeddings. These communities are the primary candidates for the underlying motifs in the time series data. The subsequences within each robust community are then further analyzed to determine the motif's representative shape.

This pipeline, which takes raw data to motif candidates, aims to provide a data-driven and shape-sensitive method for understanding the "language" of time series.

IV. Results

This section details the outcomes of applying my motif extraction methodology for "Time Sentence", from data preprocessing and subsequence embedding to the discovery of motif candidates via community detection. The experiments were primarily conducted on 1-minute BTC-USDT close price data from January 2018 to March 2025.

A. Subsequence Extraction

After standardizing timestamps and focusing on closing prices, log-transformation followed by Z-score normalization was applied to the time series. This produced a normalized series with a mean of approximately zero and a standard deviation of one (Figure 2), suitable for subsequent analysis. From this normalized series, a total of 759,789 fixed-length subsequences, each 20 minutes (20 data points) long, were extracted using a sliding window with a stride of 5 minutes.

B. Variational Autoencoder (VAE) for Shape-Aware Embeddings

A Variational Autoencoder (VAE) with an encoder architecture of ($20 \rightarrow 64 \rightarrow 32 \rightarrow 5$) and a symmetric decoder was trained to learn 5-dimensional latent representations (μ

vectors) for the 20-dimensional input subsequences. The VAE was trained using a combined loss function consisting of:

- Mean Squared Error (MSE) for reconstruction.
- KL Divergence (KLD) loss for latent space regularization (with $\beta_{KLD} = 1.0$).
- An experimental DTW-based triplet loss, designed to encourage shape similarity (measured by Dynamic Time Warping on original subsequences) to be reflected as Euclidean proximity in the latent space. This component was weighted by $\gamma_{DTW_triplet} = 1.0$, with a $margin = 1.0$. Later, I also tested values 0.1 and 0.05 for $margin$ as well.

Training was performed for a planned 10 epochs. During a representative training run of the VAE with the DTW-triplet loss component (using $margin = 1.0$, $k_positive = 1$, and $k_negative = 1.0$), the model's validation loss (based on reconstruction and KLD) converged relatively quickly. The best validation loss of approximately 0.8334 was achieved at epoch 2, after which it did not significantly improve for the subsequent 3 epochs, leading to early stopping at epoch 5.

Diagnostic observations of the DTW-triplet loss component during initial batch runs (with $margin = 1.0, 0.01, 0.05$) revealed that the calculated raw triplet loss was consistently at or very near zero, despite the mining process identifying 64 triplets per batch. This indicated that for the selected triplets (k-th DTW-closest positive and k-th DTW-furthest negative within the batch), the VAE's latent space, primarily shaped by the reconstruction and KLD objectives, already satisfied the triplet margin condition ($D_E(\mu_A, \mu_P)^2 - D_E(\mu_A, \mu_N)^2 + margin \leq 0$). Unfortunately, this specificly-designed loss component did not provide a strong learning signal to further structure the latent space based on DTW similarity in these initial configurations.

The final μ embeddings from the best VAE model (based on reconstruction and KLD validation loss) were extracted for all 759,789 subsequences and saved.

C. Latent Space Visualization and Preliminary Cluster Analysis

To qualitatively assess the structure of the learned 5-dimensional latent space, t-SNE was applied to a random sample of 5,000 μ embedding samples, with cluster labels derived from `MiniBatchKMeans` with $k = 70$ (Figure 3). The resulting plot of Figure 3 reveals a non-random organization; the embeddings form distinct, elongated, and sometimes looping structures or filaments. There are also regions of varying point density, suggesting that some patterns (represented by embeddings in denser areas) are more common than others. This structured appearance indicates that the VAE has learned to differentiate subsequences based on their features, mapping them to different regions of the latent space.

Moreover, a Silhouette Score of 0.5294 was achieved. This score, applied to a random sample of 50,000 embeddings, moderately above 0 and approaching 0.6, suggests that the `KMeans` algorithm could find reasonably distinct clusters in the VAE's latent space, indicating some level of inherent separability among the embeddings (though the number of clusters, k, must be specified manually and needs more careful tuning).

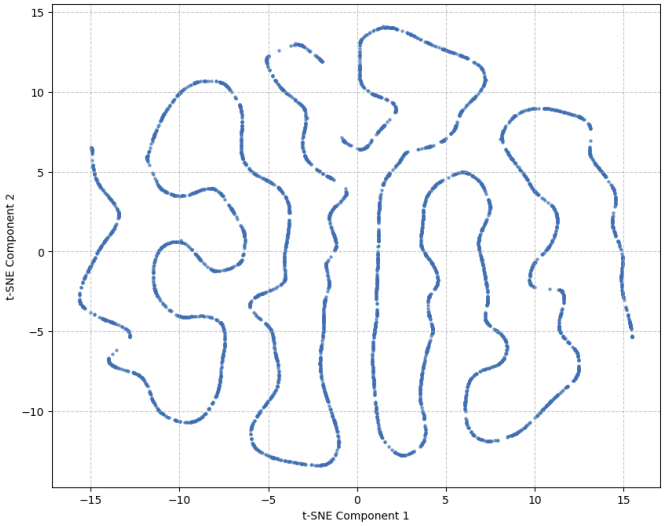

Fig. 3. t-SNE of VAE-DTW latent space (5,000 μ embedding samples)

D. Motif Candidate Discovery via Community Detection

he full set of 759,789 5-dimensional VAE embeddings served as input for graph-based community detection.

1) Similarity Graph Construction: A k-Nearest Neighbor (k-NN) graph was built. Nodes represent the subsequence embeddings. Edges were created by connecting each embedding to its $k = 50$ nearest neighbors in the 5-dimensional latent space, using Euclidean distance as the similarity metric. Edge weights were assigned using a Gaussian kernel, with the bandwidth σ determined by the mean distance to the 50th neighbor. The graph was then made undirected.

2) Leiden Community Detection: The Leiden algorithm with the resolution parameter of 1.0 was applied to this weighted, undirected graph. The algorithm identified **8,151 communities** in the graph. Figure **??** shows that a vast majority of these communities were very small. For instance, 7,361 communities consisted of only a single member, likely representing highly unique or noisy subsequences. On the other hand, **321 distinct communities have at least 100 members**, which constitute my primary set of candidate motifs. These larger communities represent approximately 98.87% of the subsequences that were initially assigned to any community.

3) Motif Representation: The original 20-point subsequences belonging to these communities were assigned a "motif ID" corresponding to their community. Visual inspection of sample subsequences from selected communities indicates varying degrees of internal shape coherence, with some communities representing clear visual patterns such as flat trends, slight inclines/declines, or more volatile V-shapes. This sequence of assigned motif IDs tokenizes the original time series, transforming it into a sequence of learned "words", which is the intended input for subsequent analysis stages.

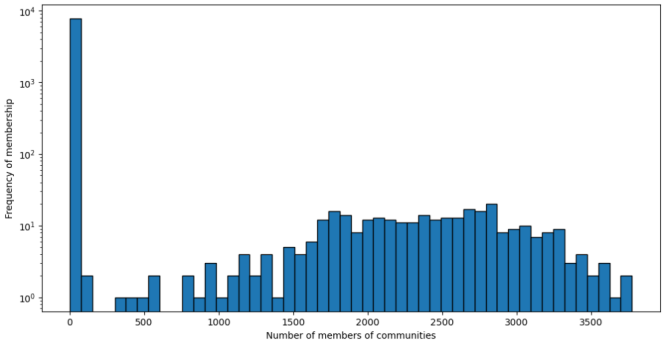

Fig. 4. Histogram of community sized

Figure 5 demonstrates a few samples of the motifs within the same cluster. Figure 5 clearly shows that in some cases, my method can put similarly-shaped motifs within the same community, while I assume because of the comparison in the Euclidean space, it cannot maintain their whole shape similarity in many cases, which is a downside of using this method, which generates a similarity graph based on the adjacency of the embeddings in the Euclidean space, and then, performs a community detection algorithm to cluster the similarly-shaped motifs in a very fast way.

V. Discussion

This project set out to develop a framework for the unsupervised discovery of shape-based motifs in time series data, drawing an analogy to words forming sentences. The methodology, involving subsequence extraction, Variational Autoencoder (VAE) based embedding with an experimental DTW-based triplet loss, and graph-based community detection, has yielded several important insights and highlighted areas for further development.

The results show that the VAE successfully learned a structured latent space from the 1-minute BTC-USDT subsequences, as evidenced by the non-random organization in the t-SNE visualization (Figure 3) and a moderate Silhouette Score (0.5294 for $k = 70$ with KMeans). This suggests that the VAE, primarily driven by its reconstruction and KLD regularization objectives, was able to differentiate and group subsequences based on features learned during its training.

The subsequent application of k-NN graph construction on these VAE embeddings, followed by Leiden community detection, proved to be an effective and computationally efficient strategy for identifying a large number of initial communities (8,151), which were then filtered to a more manageable set of 321 candidate motifs. This graph-based community detection approach is particularly advantageous as it does not require pre-specifying the number of clusters, unlike methods such as `KMeans`. Moreover, for very large datasets of embeddings, building a k-NN graph and running Leiden can be more scalable and memory-efficient than some density-based clustering algorithms that might require computation of a full distance matrix. The Leiden algorithm's ability to find communities of varying shapes and sizes within the graph structure is also well-suited for discovering a diverse motif vocabulary.

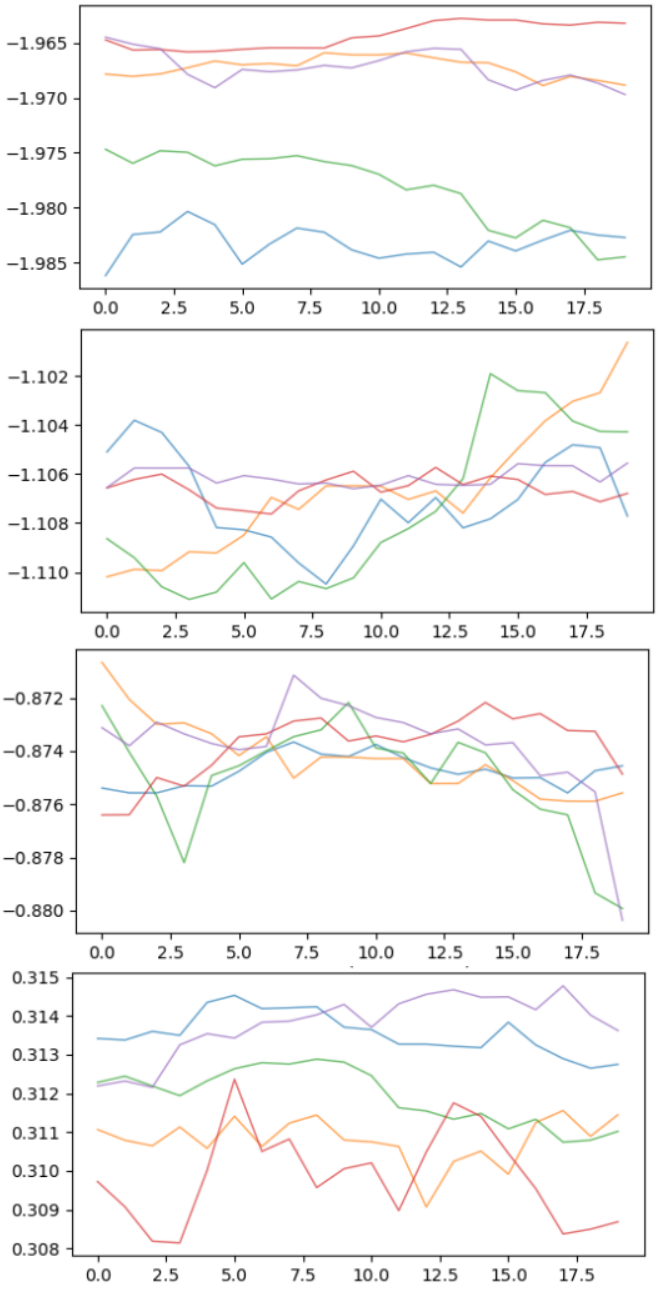

Fig. 5. Motif samples from different communities with at least 100 members.

However, a key objective was to make the VAE's latent space explicitly "shape-aware" through a DTW-based triplet loss. The results from training indicated that this triplet loss component remained largely inactive. Even with adjustments to $margin$ parameter, the latent space configuration achieved by the VAE (driven by reconstruction and KLD) already satisfied the triplet conditions for the positives and negatives selected by my in-batch DTW-based mining strategy. This implies that while the VAE embeddings are structured, their structure is not being significantly refined by the DTW-triplet

loss in its current implementation to more strongly reflect DTW-defined shape similarities beyond what the other VAE objectives incidentally achieve. The visual coherence of shapes within the final 321 communities would therefore primarily come from the VAE's ability to group generally similar patterns for reconstruction, rather than a strong, explicit DTW-driven shaping of the latent space.

A. Challenges

During this project, I faced several technical and conceptual challenges that are common in unsupervised time series analysis and representation learning:

1) Effectiveness of the VAE's DTW-Based Triplet Loss: Despite ensuring triplets were being mined based on DTW distances of original subsequences, the Euclidean distances of their corresponding latent embeddings often already satisfied the triplet margin. This problem suggests that either the in-batch mining strategy was not selecting sufficiently "hard" triplets, the margin was not optimally scaled to the latent space distances, or the VAE's reconstruction/KLD objectives created a latent space where these specific triplets were already well-separated. Tuning the relationship of the VAE's standard losses and the metric learning component for shape awareness remains a significant challenge.

2) Fixed Window Size and Stride: he current methodology relies on a fixed window size (20 minutes) for subsequence extraction. This approach limits the scale of motifs that can be discovered; shorter or longer characteristic patterns might be missed or improperly segmented. The choice of stride (5 minutes) also involves a trade-off: smaller strides capture more overlapping (and potentially redundant) information but reduce the chance of missing a pattern due to arbitrary window placement, while larger strides are more computationally efficient but risk skipping over relevant segments.

B. Parameter Tuning Complexity

The multi-stage pipeline involves numerous hyperparameters that require careful tuning. These include the VAE architecture itself (depth, width, latent dimension), weights for different loss components (β_{KLD}, $\gamma_{DTW_triplet}$), the triplet loss margin, parameters for k-NN graph construction (k), and the resolution parameter in the Leiden algorithm. Finding an optimal set of parameters in such an unsupervised setting without clear, direct ground truth for motifs is a demanding task.

C. Future Work

The current results and identified challenges show the direction for my future research and development:

1) Enhancing Shape-Awareness in Embeddings: Implementing harder negative/positive mining strategies for my DTW-based loss that specifically look for triplets/pairs that violate the desired metric structure in the latent space.

2) Multi-Scale Motif Discovery: Incorporating methods for extracting and analyzing subsequences of varying window sizes. This would allow the discovery of a richer hierarchy of motifs, from short, atomic patterns to longer, composite ones. Additionally, investigating dynamic stride adjustment or adaptive segmentation techniques.

3) Motif Dictionary Formation, Validation, and Convergence Testing: Formalizing the process of creating the "motif dictionary" from the detected communities, developing quantitative metrics to evaluate the reconstructive power of the motif dictionary on unseen time series or segments, and Formalizing the process of creating the "motif dictionary" from the detected communities, developing quantitative metrics to evaluate the reconstructive power of the motif dictionary on unseen time series or segments, and systematically testing the **convergence hypothesis**.

4) Improving Parameter Robustness and Automation: Investigating methods for more automated or adaptive tuning of key hyperparameters (e.g., k in k-NN, resolution in Leiden, VAE loss weights).

REFERENCES

[1] Mueen, Abdullah, et al. "Exact Discovery of Time Series Motifs." Proceedings of the 2009 SIAM International Conference on Data Mining, SIAM, 2009, pp. 473–484. DOI:10.1137/1.9781611972795.41.

[2] Berndt, Daniel J., and James Clifford. "Using Dynamic Time Warping to Find Patterns in Time Series." Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases, AAAI Press, 1994, pp. 229–248.

[3] Kingma, Diederik P., and Max Welling. "Auto-Encoding Variational Bayes." arXiv, 20 Dec. 2013, arXiv:1312.6114, https://arxiv.org/abs/1312.6114.

[4] Schroff, Florian, et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2015, pp. 815–823. DOI:10.1109/CVPR.2015.7298682.

[5] Traag, Vincent A., Ludo Waltman, and Nees Jan van Eck. "From Louvain to Leiden: Guaranteeing Well-Connected Communities." Scientific Reports, vol. 9, 2019, article 5233. DOI:10.1038/s41598-019-41695-z.

[6] Denouden, Taylor, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Buu Phan, and Sachin Vernekar. "Improving Reconstruction Autoencoder Out-of-Distribution Detection with Mahalanobis Distance." arXiv, 6 Dec. 2018, arXiv:1812.02765, https://arxiv.org/abs/1812.02765.