KitikiPlot

A Python library to visualize categorical sliding window data

Boddu Sri Pavan **Chandrasheker Thummanagoti** **Boddu Swathi Sree**

ABSTRACT

The Sliding Window technique is a versatile approach frequently applied in domains ranging from genomics and electronics to climate analysis and algorithm development. While Matplotlib facilitates the visualization of sequential continuous data, ***KitikiPlot****, our Python library, is dedicated to complementing this by focusing on the visualization of sequential categorical data using sliding windows. By offering intuitive and customizable visualizations, it empowers researchers, data scientists, and analysts across diverse fields to derive rapid insights into categorical trends, promoting informed decision-making. Built on top of Matplotlib, it seamlessly integrates with various research projects and demonstrates significant potential across multiple fields. This library is open-source and available on GitHub:* https://github.com/BodduSriPavan-111/kitikiplot.

Keywords Sliding Window · Categorical data · Genomics · Electronics · Climate Analysis · Air Quality Monitoring

1 Introduction

The sliding window technique is a fundamental concept extensively utilized across various domains and algorithms. At its core, this technique operates as a dynamic queue, where simultaneous enqueue and dequeue actions create the sliding effect. Its versatility finds applications across industries, from genomics, where it aids in the analysis of DNA sequences, to signal processing for analyzing discrete signals, to climate analysis for historical trend analysis, and air quality monitoring to identify rising pollutant levels, etc. Visualization plays a critical role in these domains, offering not only quick insights but also enhancing the interpretability of data through aesthetic representation.

Data is broadly classified into two categories: continuous data and categorical data. Continuous data comprises measurable quantities that vary fluidly over intervals, while categorical data represents distinct categories or states, often occurring in sequences. While tools like Matplotlib excel in supporting the visualization of continuous data with functionalities such as the *matplotlib.pyplot.plot()* function, KitikiPlot complements these capabilities by addressing the unique requirements of sequential categorical data visualization. By offering a clear and efficient representation of such data, KitikiPlot enhances the ability of users to derive meaningful insights, streamlining their analysis workflows and advancing visualization capabilities for categorical data.

2 Related Works

Matplotlib [1], a widely used Python library for data visualization, offers an extensive set of tools to create plots such as line plots, bar plots, and scatter plots, etc. Its flexible APIs enable the customization of every visual element, but it often requires verbose code to implement advanced or domain-specific visualizations. NumPy [2], another core library in Python, excels in numerical computations and serves as the backbone for data handling in visualization libraries. Despite its efficiency in array-based operations, it does not provide direct visualization capabilities, necessitating additional tools to transform numerical insights into graphical representations. Pandas [3] is a foundational Python library for data manipulation and analysis. It provides robust support for handling structured datasets, including indexing, slicing, and aggregation, making it indispensable in pre-visualization data preprocessing workflows. However, its visualization capabilities, built on Matplotlib, lack the sophistication and customization required for specialized tasks.

VisualGesturesJS [4], an open-source TypeScript package, enables touchless interaction by employing finger movements for cursor actions such as hover, click, drag, and drop. This library leverages a sliding window mechanism to enhance gesture detection precision, powered by the *Fluid Kink Ratio Algorithm* which we designed earlier. The algorithm was conceived with the visual representation of sliding windows in our mind, which sow the seed for the development of a dedicated Python library for sliding window visualization of categorical data, 'KitikiPlot'.

3 Proposed Methodology

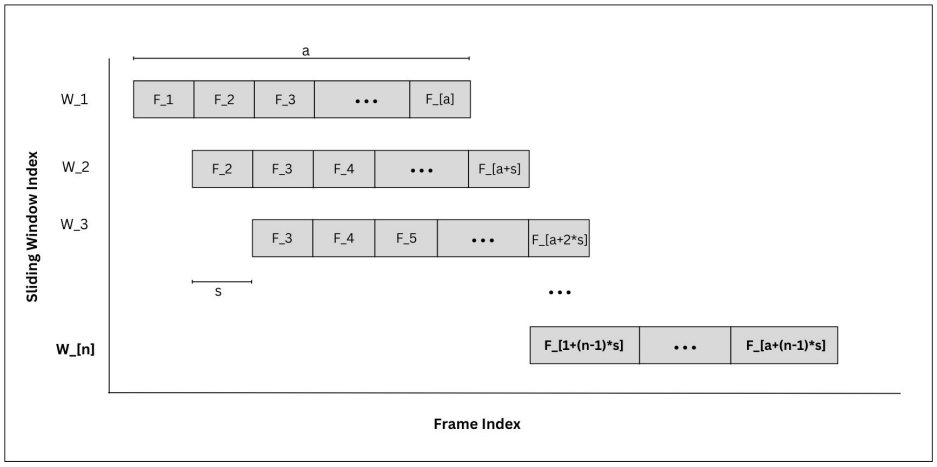

Figure 1: Visualization of the proposed KitikiPlot methodology, illustrating the relationship between sliding windows and their constituent frames. Each rectangular bar (window) represents sequential categorical data segmented by a sliding window of length a and stride s (here s= 1). The layout effectively demonstrates the indexing and overlap of frames across successive windows.

KitikiPlot provides an intuitive visual representation of sliding window data using rectangular bars, each corresponding to a specific sliding window. These bars are further divided into smaller rectangular cells, termed "Frames", arranged side by side to depict sequential categorical data values within the window. The methodology employs a sliding window technique characterized by two parameters: window length ($a \in$ N) and stride ($s \in$ N). A given sliding window W_n encompasses data points from $F_{1+(n-1)s}$ to $F_{a+(n-1)s}$, where F denotes an atomic categorical data unit, and $n \in$ N. This mathematical formulation ensures that the visualization accurately represents overlapping or non-overlapping sequences based on the chosen stride and window length, thereby supporting diverse analytical requirements, as shown in Figure 1.

It offers extensive customization capabilities to cater to varied visualization needs. Users can modify visual elements such as color schemes, hatching patterns, and bar alignments, ensuring adaptability to different datasets and aesthetic preferences. The library also supports dynamic labeling, allowing for precise adjustments to axis labels, tick marks, titles, and legends. These features make KitikiPlot a versatile and robust tool for presenting categorical data in a compact, visually interpretable manner. By converting complex sequential data into an organized visual structure, KitikiPlot bridges the gap between raw data and actionable insights, offering a unique solution for researchers and analysts in domains requiring temporal or sequential data visualization using sliding windows.

4 Case Studies

4.1 Genomics

Genomic analysis involves studying the genetic material of living organisms, which consists of sequential arrangements of nucleotides (in Deoxyribonucleic acid): Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). These nucleotide sequences dictate the biological traits and functions of a living organism. Analyzing large genomic sequences often requires segmentation into smaller, manageable components using computational techniques such as sliding window approaches.

The KitikiPlot visualization method plays a crucial role in genomics by transforming complex nucleotide sequences into intuitive visual representations. For example, consider a sample genome dataset sourced from the UCI repository

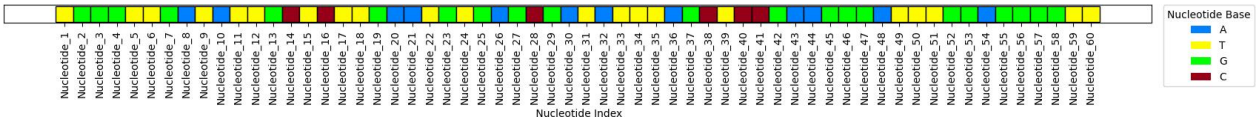

Figure 2: KitikiPlot visualization of a single genome sequence without a sliding window. The x-axis represents the nucleotide index across the sequence, while each nucleotide (A, T, G, or C) is color-coded to provide a visual understanding of nucleotide distribution.

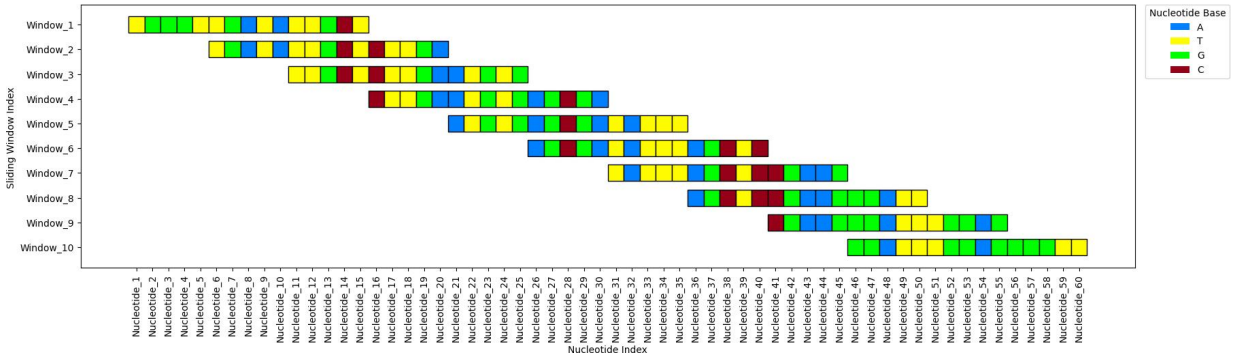

Figure 3: KitikiPlot visualization of a single genome sequence from the Figure 2, with a sliding window. The x-axis represents the nucleotide index across the sequence, while the y-axis denotes sliding windows with a window length of 15 and stride size of 5. Each nucleotide base (Adenine (A) - Blue, Thymine (T) - Yellow, Guanine (G) - Green, Cytosine (C) - Brown) is distinctly color-coded. This visualization enables clear identification of nucleotide patterns and sequential arrangements across genomic segments.

on molecular biology splice junction gene sequences [5]. Using a sliding window technique with a fixed window length of 15 and stride of 5 as shown in Figure 3, it maps the nucleotide sequence to a visual grid, where each cell represents a nucleotide within a given window. This visualization enables scientists to observe patterns and relationships that are not easily discernible in raw genomic data. From Figure 2, and Figure 3, it becomes apparent how nucleotides are spatially arranged, such as Adenine (Blue) may frequently co-occur with Thymine (Yellow) or Guanine (Green) but not with Cytosine (Brown). Such co-occurrence patterns can provide insights into nucleotide pairing tendencies, sequence conservation, and genomic motifs. By leveraging sliding window visualization, KitikiPlot facilitates pattern recognition by enabling the identification of recurring motifs such as promoter regions, introns, or exons within nucleotide sequences. It supports sequence similarity analysis by allowing comparative visualization of genomic segments, thereby providing insights into evolutionary relationships. Additionally, it aids in detecting mutations by highlighting anomalies or nucleotide substitutions in sequential arrangements. By transforming raw alphanumeric sequence data into intuitive visual representations, it enhances interpretability for genomic researchers, offering a novel perspective for visual intuition.

4.2 Electronics

In the field of electronics, signal processing plays a pivotal role in analyzing, interpreting, and manipulating electrical signals to derive meaningful insights. Electrical signals, which can vary in amplitude or frequency over time, are broadly categorized into analog signals and digital signals. Analog signals are continuous, representing real-valued measurements such as voltage or current, while digital signals are discrete, typically represented by binary values (0 and 1). Effective signal processing is essential for various applications, from telecommunications and audio processing to sensor data analysis and Internet of Things (IoT) systems.

KitikiPlot introduces a novel approach to visualizing signal data by transforming raw signal measurements into structured, interpretable visual representations as shown in Figure 4, 5. Sliding window analysis provides an intuitive framework for identifying patterns, transitions, and anomalies in signal datasets.

For example, consider telemetry data (first 50 records) from an environmental sensor dataset [6], which tracks the intensity of light over time (0 or 1). Using KitikiPlot, the temporal variability of light intensity can be visualized effectively, revealing non-continuous fluctuations and distinct temporal patterns, as shown in Figure 6. The sliding window visualization highlights transitions and trends within predefined intervals, offering a granular view of signal

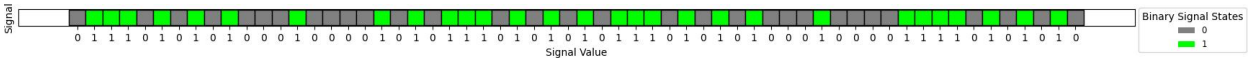

Figure 4: KitikiPlot visualization of a discrete binary signal sequence (01110101010001000010101110101010111010101000100001111010101O). The x-axis represents the signal values (binary states: 0 and 1), and the y-axis corresponds to the index of each bit in the sequence. This visualization aids in identifying patterns or transitions within the signal, facilitating a deeper understanding of discrete binary data dynamics.

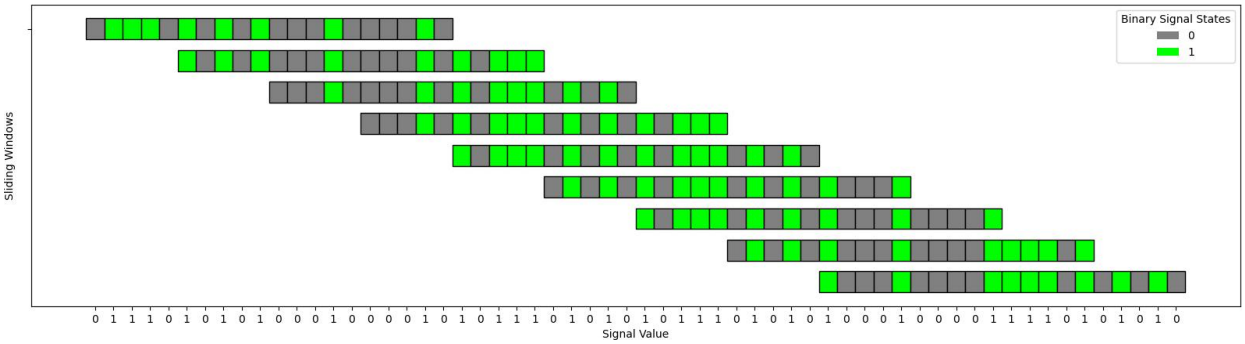

Figure 5: KitikiPlot visualization of a discrete binary signal sequence (in Figure 4) using sliding window approach with a window length of 20 and a stride of 5. The x-axis represents the signal values (binary states: 0 and 1), while the y-axis corresponds to sliding window indices.

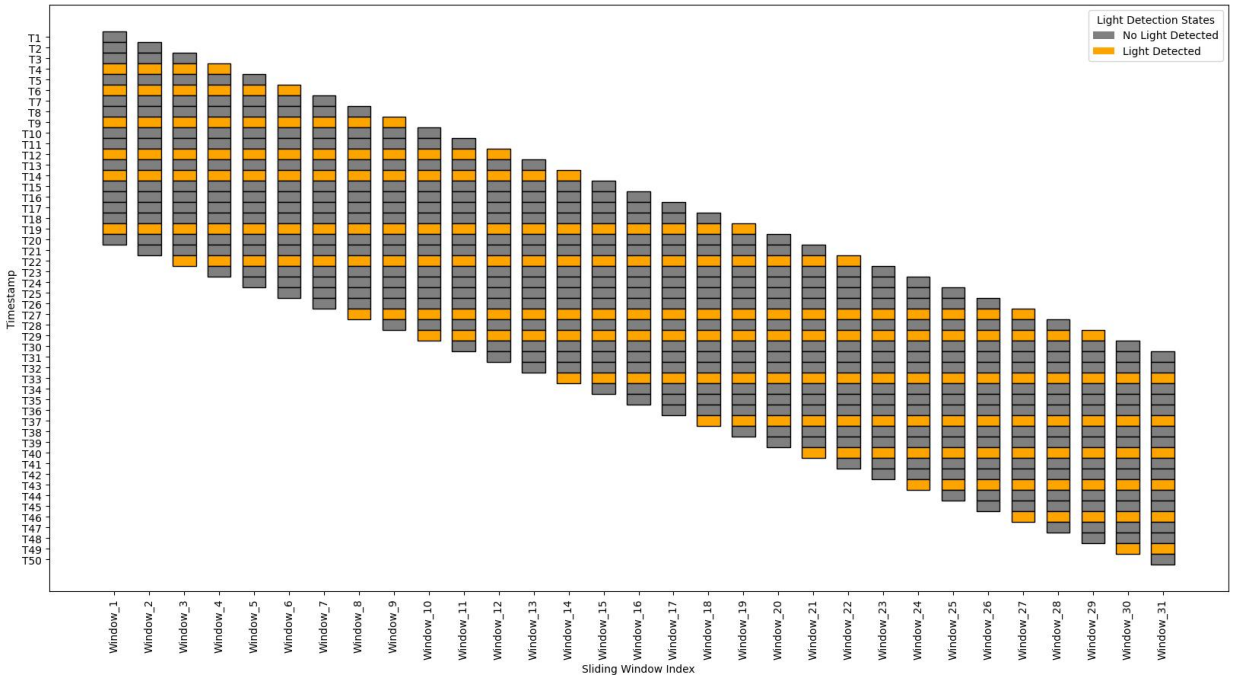

Figure 6: KitikiPlot visualization of a discrete binary signal sequence describing temporal variability of light intensity using sliding window approach with a window length of 20 and a stride of 1. The x-axis represents the sliding window indices, while the y-axis corresponds to signal values (binary states: 0 and 1) across continuous timestamps. This transformation enables the analysis of temporal patterns and transitions within the signal, offering a clearer representation of binary signal dynamics over consecutive segments.

behavior. This enables researchers and engineers to detect irregularities, such as signal dropouts or spikes, that might be harder to find from raw numerical data.

It extends the utility to various domains in electronics by enabling intuitive visualization of complex signal data. For instance, it facilitates anomaly detection in IoT sensor networks by uncovering correlations between multi-sensor streams, enhances the analysis of modulated signals in telecommunications by highlighting noise patterns or signal loss, and optimizes control systems by identifying trends in real-time feedback signals. By transforming raw signal data into actionable visual representations, KitikiPlot equips researchers with tools to analyze signal dynamics effectively, fostering smarter diagnostics and improved decision-making in sophisticated electronic systems.

4.3 Delhi Climate Analysis

Climate analysis systematically examines historical weather data over a specified period, focusing on key factors such as temperature, humidity, atmospheric pressure, etc. These parameters exhibit variations over time, necessitating effective methods for visualization and analysis. Consider the mean temperature trends in Delhi between 2013-01-01 and 2013-04-10 from DELHI Weather dataset [7]. While temperature is inherently a continuous variable, discretizing it into categorical bins facilitates better interpretability and pattern identification. In this case, temperature values are grouped into five categories: temperatures below 10°C were classified as "Freezing", values within [10, 15) as "Chilly," within [15, 20) as "Mild", within [20, 25) as "Warm", within [25, 30) as "Hot", and values exceeding or equal to 30°C as "Scorching". Each category is associated with a unique color to enhance visual differentiation.

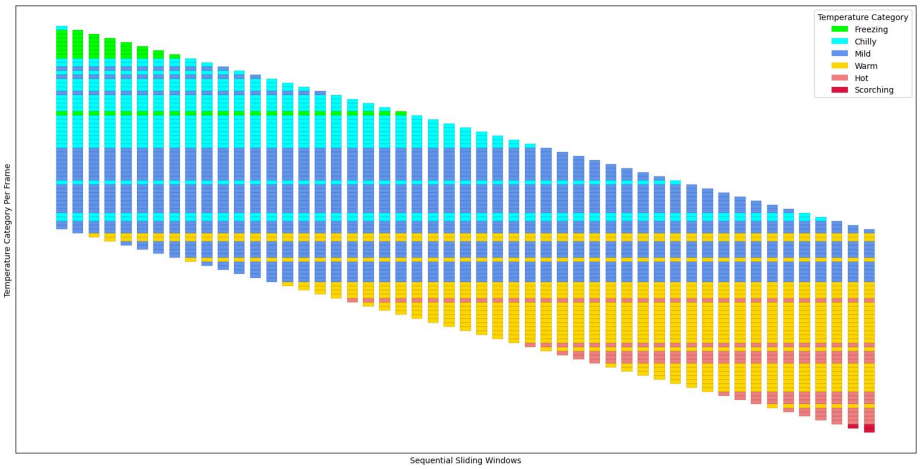

Figure 7: Visualization of Delhi's mean temperature data using KitikiPlot. The x-axis represents sliding windows of 50 consecutive days, and the y-axis denotes each frame with the corresponding temperature category for each date. The window length is set to 50 with a stride of 1, facilitating the analysis of temperature transitions across time and identifying patterns in the temperature distribution.

Using KitikiPlot, sequential temperature patterns are plotted, Figure 7 reveals that abrupt changes exceeding five temperature units are absent. Instead, transitions predominantly occurred between adjacent categories, such as from "Chilly" to "Mild" or "Warm" to "Hot" without skipping intermediate classes. This visualization underscores the consistency and gradual nature of temperature variation in Delhi during the analyzed period. KitikiPlot thus emerges as a valuable tool for climatologists and meteorologists, offering clear, interpretable visualizations that aid in identifying trends and anomalies, ultimately providing actionable insights into climate dynamics.

4.4 Air Quality Monitoring

Air quality is a critical environmental metric, reflecting the presence of pollutants and their potential impact on health and ecological systems. It is influenced by various factors, including the concentration of gases such as carbon monoxide (CO), carbon dioxide (CO2), and particulate matter (PM), which are released primarily through human activity, industrial emissions, and combustion processes. Effective monitoring of these pollutants is essential for public health, urban planning, and environmental conservation.

Consider the CO(GT) records on 2004-11-11 from the Air Quality dataset [8] . Through KitikiPlot, a sliding window approach has been applied to visualize temporal patterns of CO(GT) levels. The six distinct categories of

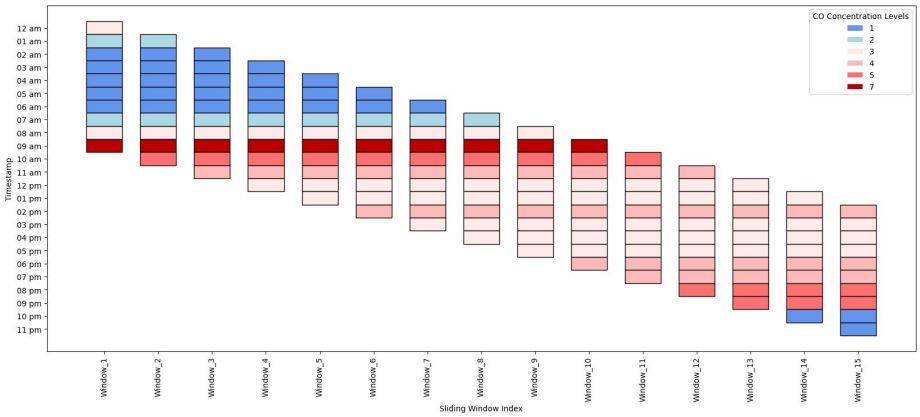

Figure 8: KitikiPlot visualization of the CO(GT) levels from the Air Quality dataset, utilizing a sliding window approach with a window length of 10 and stride of 1. The x-axis represents the sliding window index, with values labeled as Window_1 to Window_n (n=15) and the y-axis corresponds to timestamps from 12 am to 11 pm. The legend categorizes CO concentrations into six levels, ranging from low to high, providing insights into temporal pollution patterns associated with human activity cycles.

CO concentration are represented as discrete classes with distinct colors, namely: 1 , 2, 3, 4, 5, and 7, which can be associated with varying levels of pollution intensity. The legend in the visualization maps each class to a specific color, aiding in the interpretation of pollutant levels across time. The visualization from Figure 8 reveals that CO levels are typically at their lowest around 4 AM, sharply rise to a peak around 9 AM, and then gradually decrease toward a minimum around 1 PM. A subsequent increase is observed between 6-9 PM, followed by another decline toward midnight. These fluctuations align with human activity cycles, with higher CO concentrations during periods of increased activity and lower levels during late-night hours when activity is minimal. This temporal behavior of CO levels suggests a correlation between pollution spikes and human mobility patterns, underscoring the significance of using KitikiPlot for analyzing environmental data.

KitikiPlot, when applied to this dataset, significantly enhances the visualization and interpretation of time-series data. By transforming raw air quality data into visual representations, it empowers environmental researchers to gain deeper insights into air pollution dynamics. This can lead to more accurate assessments of pollution sources and more effective mitigation strategies. It not only simplifies the analysis of complex environmental datasets but also enhances the ability to communicate pollution trends and make data-driven decisions in air quality monitoring. Its sliding window-based visualization approach offers a more comprehensive understanding of time-series environmental data, aiding the identification of critical pollution patterns that would otherwise be challenging to detect.

4.5 VisualGesturesJS

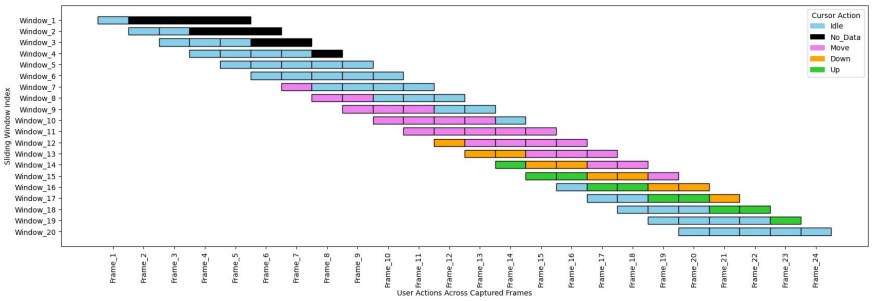

Figure 9: Visualization of user action sequences using KitikiPlot where the x-axis represents user action detected for each frame, and the y-axis consecutive sliding windows of length 5, stride 1, where distinct colors represent different events, enabling clear identification and differentiation of user interactions during gesture-based operations.

VisualGesturesJS [4] is a Node Package Manager (NPM) package designed to enable cursor control and associated operations through hand gestures, replacing conventional touch-based mechanisms. The system integrates a novel *Fluid-*

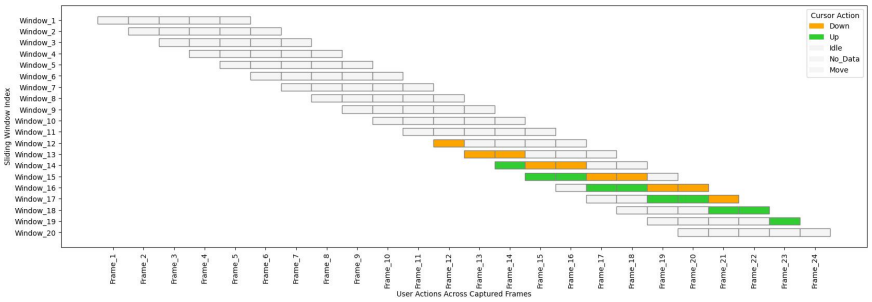

Figure 10: Sliding window visualization focusing specifically on cursor up and down events, where the x-axis represents user action detected for each frame, and the y-axis consecutive sliding windows of length 5, highlighting the frame-level sequential relationship within the user gesture actions.

Kink Ratio Algorithm, which utilizes a sliding window technique to distinguish gesture-based events. For instance, in the case of a click event, the algorithm identifies it as a sequence comprising consecutive cursor up and down movements, provided there is no significant displacement of the hand. Notably, any substantial movement during this process is interpreted as a drag-and-drop action instead.

To analyze this behavior, Figure 9 and Figure 10 present the corresponding visualization plots, where the x-axis denotes user actions across captured frames, and the y-axis represents a sliding window of five consecutive frames with a stride size of 1. The KitikiPlot visualization method has been employed to capture event sequences and generate actionable insights. The analysis focuses particularly on cursor up and down events. Observational findings indicate that each cursor up and down action spans two consecutive frames, resulting in a total of four frames to complete a click event. This empirical evidence highlights the frame utilization efficiency of the *Fluid Kink Ratio Algorithm* for event classification and recognition. By leveraging the sliding window framework, the KitikiPlot visualization technique plays a crucial role in providing clear, interpretable insights into event sequences. Specifically, it enables the identification of patterns such as frame allocation during cursor actions, helping validate the algorithm's effectiveness and ensuring precise analysis of gesture-based interactions.

5 Conclusion

In conclusion, KitikiPlot bridges a critical gap in data visualization by providing an efficient and user-friendly framework for representing categorical sliding window data, complementing existing tools like Matplotlib. By enabling quick insights into complex datasets, it supports researchers and analysts in various domains, from genomics, and electronics to time-series analysis. As an open-source project, KitikiPlot invites contributions from the community to enhance its capabilities, ensuring its continual evolution and relevance across diverse applications.

6 Acknowledgements

We express our heartfelt gratitude to our parents and family members for their unwavering support. Our sincere appreciation also goes to Mathala Vasu and Nagendra Dharmireddi for their domain-specific insights. This work is conducted with the gracious blessings of Sri Ramalinga Chowdeswari Devi.

References

[1] J. D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, doi: 10.1109/MCSE.2007.55, May-June 2007.

[2] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. *Nature* 585, 357–362. doi: 10.1038/s41586-020-2649-2, 2020.

[3] McKinney, W., & others. Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, Vol. 445, pp. 51–56, 2010.

[4] Nagendra Dharmireddi& Boddu Sri Pavan. visual-gestures v0.0.1. Lets you control the cursor with hand or finger movements. https://www.npmjs.com/package/@learn-hunger/visual-gestures, 2024.

[5] Molecular Biology (Splice-junction Gene Sequences) Dataset UCI Machine Learning Repository. https://doi.org/10.24432/C5M888. https://archive.ics.uci.edu/dataset/69/molecular+biology+splice+junction+gene+sequences, 1991.

[6] Gary A. Stafford Environmental Sensor Telemetry Data. Kaggle. https://www.kaggle.com/datasets/garystafford/environmental-sensor-data-132k, 2020.

[7] R Kiran Kumar Reddy. DELHI Weather dataset. Kaggle. https://www.kaggle.com/datasets/devitachi/delhi-weather-dataset, 2024.

[8] Saverio Vito. Air Quality. UCI Machine Learning Repository. https://doi.org/10.24432/C59K5F. https://archive.ics.uci.edu/dataset/360/air+quality, 2008.