Introduction:

I aim to develop an algorithm for downsampling time series datasets that can be effectively applied to both real-time data streams and batch processing environments.

Objectives:

- Develop an algorithm that downsamples time series data while preserving its key features and important characteristics.
- Ensure compatibility with both real-time streaming and batch datasets.
- Design an adaptive algorithm capable of adjusting itself based on the characteristics of the input data
- Address the challenge of long-range dependencies among data points within the time series.

Architecture Overview:

The core concept behind this algorithm was inspired by the image processing coursework I completed in Semester 2, where I developed an image denoising algorithm using detail coefficients (a project that received positive feedback from my lecturer).

Building on that experience, and after reviewing several relevant research papers and methodologies, I propose a similar approach for time series data. The signal is first decomposed using wavelet transforms into approximation and detail coefficients. Previous studies have typically relied on approximation coefficients for downsampling, as they effectively capture the general trend of the signal. However, this often leads to the loss of critical information such as peaks and dips,features that are essential for accurate representation and analysis.

To overcome this limitation, I argue that detail coefficients must be included in the downsampling process. Therefore, my approach focuses on processing the detail coefficients using a Transformer-based model, enabling downsampling in the frequency domain while preserving fine-grained signal features.

The choice of a Transformer-based model is deliberate. Transformers are well-suited for learning complex patterns in sequential data and are capable of capturing long-range dependencies between data points,an essential characteristic when dealing with time series. Moreover, their adaptive nature allows them to dynamically recognize and retain significant features within varying datasets. The transformer can potentially learn which high-frequency details are most critical to retain, tailoring the downsampling to the data's specific characteristics.

This architecture combines the multi-resolution capability of wavelet transforms with the powerful pattern recognition and sequence modeling abilities of Transformers. Compared to traditional methods, this approach aims to achieve a balanced representation of both the signal's overall trend and its high-frequency components.This forms the core of my downsampling architecture.

To make the architecture suitable for stream processing, I incorporated Apache Kafka and Apache Flink, which together provide a robust infrastructure for handling real-time data streams. For this initial prototype, I set up two Kafka topics: one for input data and another for storing the processed output (sink). Apache Flink acts as the processing engine,reading data from the Kafka source topic, passing it through the proposed architecture for downsampling, and writing the results to the output topic.

Dataset:

The ECG200 dataset is a publicly available collection of 200 electrocardiogram (ECG) recordings, each capturing the electrical activity during a single heartbeat. It is a well-established benchmark in time series classification. While this dataset has been used for the initial development and evaluation of the algorithm, a larger dataset will be required to achieve more robust performance and higher accuracy.

How the Algorithm Works:

- **Wavelet Transform**:

  It starts by applying a wavelet transform to decompose the time series into two types of coefficients:

  - Approximation Coefficients: These represent the low-frequency components, capturing the smooth, overall trend of the data.
  - Detail Coefficients: These capture the high-frequency components, such as noise, sudden changes, or fine details.
- **Retaining Approximation Coefficients**:

  I keep the approximation coefficients as they are, preserving the general behavior of the time series. Since these coefficients are fewer in number (depending on the decomposition level), this step naturally reduces the data size to some extent.
- **Processing Detail Coefficients with a Transformer**:

  The detail coefficients are passed to a transformer model, which downsamples them. Transformers are adept at capturing complex patterns and dependencies, so this step likely compresses or selects the most important high-frequency details into a lower-dimensional representation.
- **Reconstruction**:

  Finally, I combine the retained approximation coefficients with the downsampled detail coefficients to produce a reduced version of the original time series.

Strengths of the Algorithm

- **Multi-Scale Preservation**:

  By leveraging the wavelet transform, my approach separates the time series into trend (approximation) and details (high-frequency components). This ensures that both the "big picture" and key fine-grained features are considered during downsampling, which is a significant advantage over methods that treat all data uniformly.

- **Flexible Detail Handling**:
  Using a transformer to downsample the detail coefficients adds adaptability. The transformer can potentially learn which high-frequency details are most critical to retain, tailoring the downsampling to the data's specific characteristics. This is more sophisticated than simply discarding details or applying a fixed rule.
- **High Fidelity Potential**:
  If the transformer is well-designed and trained, the algorithm could preserve important features,both trends and selective details,that might be lost in simpler methods.This makes it promising for applications where maintaining signal integrity is crucial.
- **Hybrid:** It combines classical signal processing (Wavelet Transform) with modern deep learning techniques (Transformers).
- **Frequency-Selective:** It leverages the DWT's strength in separating a signal into different frequency components (low-frequency approximations vs. high-frequency details).
- **Differential Processing:** It treats these components differently. Low-frequency information (approximations) is downsampled more simply (pooling), assuming it's smoother and less complex. High-frequency information (details) is processed through a more powerful Transformer architecture before downsampling, acknowledging that important, complex patterns might reside in the details.
- **Learned Downsampling:** While the DWT and pooling are fixed operations, the downsampling applied to the detail coefficients via the `Conv1D` layers within the `DownsampleTransformerBlock` is learned. The weights of the convolution (and the transformer) are optimized during the model training (likely for the reconstruction task), meaning the model learns the best way to reduce dimensionality while preserving information relevant to the task.

The Real-Time Approach:

- **Appropriate Framework:** Using Flink is a strong choice for demanding real-time stream processing tasks. It offers robust processing guarantees, fault tolerance, and excellent performance potential.

- **Scalability:** Both Flink and Kafka are designed for horizontal scalability. If the input data rate increases, we can potentially scale out the system by adding more Kafka brokers and Flink TaskManagers (increasing parallelism).
- **Decoupling:** Using Kafka effectively decouples the data producers from the Flink processing application. This makes the system more resilient; the processing part can be restarted or updated without stopping data collection, and vice-versa.
- **Leveraging Pre-trained Models:** It effectively integrates a complex, pre-trained deep learning model into a real-time pipeline, allowing sophisticated analysis to be applied to streaming data.

Comparison to Alternatives:

- **Wavelet-Only Downsampling**:
  Keeping just the approximation coefficients is a common wavelet-based approach, but it discards all details. The use of a transformer to retain some details improves on this.
- **Largest Triangle Three Buckets (LTTB)**:
  LTTB is great for visual downsampling and is simpler, but it doesn’t explicitly handle frequency components. My algorithm might outperform it in analytical contexts where both trend and details matter.