Transformer-Based Classification for Dynamic Data Streams

Ray Chakanetsa Marange

Student ID: 300671115

Supervisor: Dr. Heitor Gomes

Submitted in partial fulfilment of the requirements for Master of Artificial Intelligence.

February 5, 2026

Abstract

This research investigates **StreamTransformer**, a lightweight Transformer for single-pass, bounded-memory classification in non-stationary data streams. Within **CapyMOA** and a **prequential** protocol, two variants, base and ADWIN-wrapped, are benchmarked against seven established learners on ten datasets. **Relational embeddings** via self-attention address limitations of axis-aligned tree splits and reduce sensitivity to **drift chattering**(repeated false drift alarms caused by noise rather than real concept change). In single-seed experiments, StreamTransformer achieves **86.1–86.5%** on Agrawal streams (exceeding ARF by **6.8–10.3 percentage points**), **69.6%** on Airlines (highest among compared methods), and **89.9–90.6%** on Electricity (ADWIN variant highest), while being **6.1× faster than SRP** on Airlines. The ADWIN wrapper underperforms on rule-based streams, indicating a **mismatch between hard resets and attention-based adaptation**. The task-dependent performance profile is discussed, and directions for **adaptive windowing** and **soft reset** strategies are outlined.

Keywords: Data Stream Learning, Concept Drift, Transformer Architecture, Attention Mechanism, Drift Chattering, Online Learning

List of Tables

List of Figures

Contents

Acknowledgements

I am deeply grateful to everyone who supported me throughout this research journey. My heartfelt thanks go to my supervisor, **Dr. Heitor Gomes**, provided steady guidance throughout the project. His deep knowledge of stream learning and the CapyMOA framework helped shape the direction of the research, and his patience and support made it possible to navigate the more complex parts of the work with confidence. I also appreciate **Dr. Aaron Chen**, coordinator of the AIML589/501 course, for creating the environment that made this work possible. I am thankful to my employer, the **South Taranaki District Council** , for giving me the opportunity to study while working full-time.AI tools such as Colab, Gemini, and GitHub Copilot provided suggestions for documentation, table generation, and Overleaf LaTeX formatting, but every part of the work was written, reviewed, and refined by me. I would also like to acknowledge the developers of CapyMOA and MOA for their essential open-source contributions. Finally, my deepest appreciation goes to my family especially Linda and to my friends, whose steady support carried me through the most demanding stages of this project.

Chapter 1

Introduction

The growing prevalence of data generating systems creates a critical demand for algorithms that learn continuously from infinite, non-stationary streams. Traditional batch learning fails under streaming constraints, **single-pass processing, bounded memory, and the need to adapt to concept drift**.

Although tree-based ensembles represent current state-of-the-art approaches, their axis-aligned splits and dependence on explicit drift detection render them brittle in noisy, complex environments. This work tests whether the **self-attention mechanism** foundational to Transformers can mitigate these limitations.

The **StreamTransformer** architecture is designed to replace reactive, binary drift-handling mechanisms with a more natural form of continuous, implicit adaptation. This chapter introduces the research objectives, guiding questions, key contributions, and the overall experimental scope.

1.1 Research Objective

Primary Objective: Conduct a comprehensive benchmark within CapyMOA, comparing the ability of PyTorch-based Transformers to capture feature dependencies against state-of-the-art (SOTA) ensembles, in order to assess their robustness on high-velocity, non-stationary data streams.

1.2 Research Questions

RQ1: Expressive Superiority: Can StreamTransformer's relational embeddings model complex feature interactions more effectively than axis-aligned tree splits?

RQ2: Concept Drift Robustness: Does implicit adaptation via sliding window attention provide sufficient stability, or is explicit drift detection necessary in high-noise environments?

RQ3: Efficiency and Latency: What computational cost does StreamTransformer incur, and does it occupy a viable position on the accuracy latency trade-off frontier for streaming applications?

1.3 Research Contributions

- **Algorithmic Innovation:** Transformer adaptation for tabular streams using bounded memory sliding window attention and causal masking
- **Empirical Benchmarking:** Comprehensive prequential evaluation across ten datasets against three tiers of contenders, including the StreamTransformer_ADWIN variant
- **Stability Discovery:** Quantitative evidence of **drift chattering** and competitive stability of implicit neural adaptation
- **Practical Guidance:** Deployment guidance identifying scenarios where StreamTransformer performed well in experiments

1.4 Experimental Scope

The study uses **CapyMOA** [1] to benchmark StreamTransformer against a hierarchy of learners on diverse streams, isolating performance under varying noise and non-stationarity.

1.4.1 Contender Hierarchy

- **Statistical/Single Tree Baselines:** Naive Bayes, Hoeffding Tree (HT), Extremely Fast Decision Tree (EFDT)
- **SOTA Ensembles:** Adaptive Random Forests (ARF) [2], Streaming Random Patches (SRP) [3]
- **Distance-Based/Neural Baselines:** k-Nearest Neighbours (kNN), Reactive MLP, and StreamTransformer_ADWIN for explicit drift detection comparison

The structure of this report is organised to guide the reader from foundational concepts through to the final conclusions. Chapter 2 introduces the core principles of data stream learning, concept drift, and neural architectures, while identifying the research gap. Chapter 3 details the experimental methodology, including dataset selection, the contender hierarchy, and the evaluation protocol. Chapter 4 then presents the StreamTransformer architecture in detail, outlining its design rationale and implementation. Chapter 5 reports the experimental results, followed by Chapter 6, which discusses the findings. Finally, Chapter 7 concludes the report and outlines potential directions for future work.

Chapter 2

Literature Review

This chapter offers an integrated review of the foundational concepts and related work in data stream learning, establishing the context for the study and highlighting the specific research gap that the StreamTransformer aims to address.

2.1 Data Stream Learning

Data stream classification operates on a potentially infinite instance sequence (x_t, y_t), where x_t is the feature vector and y_t the class label at discrete time t. Foundational work by **Domingos and Hulten (2000)** [4] establishes stringent computational constraints distinguishing stream from batch learning:

- **Single Pass Assumption:** Each instance processed exactly once and immediately discarded or summarised
- **Bounded Resources:** Fixed memory and constant amortised processing time per instance ($O(1)$)

The study employs the **Prequential (Test-Then-Train)** protocol [5]: the model predicts $\hat{y}_t$ for x_t, then updates using true label y_t, providing realistic performance estimates on future data [6].

2.2 Sequences vs. Time Series vs. Data Streams

Precise distinction between sequential paradigms contextualises neural architecture application:

- **Time Series Forecasting:** Predicts future values $y_{t+1:t+h}$ from historical $x_{1:t}$, assuming stationarity and full sequence availability [7]
- **Sequence Modelling (e.g., NLP):** Models like BERT [8] process static sequences with bidirectional/causal attention during offline training
- **Data Stream Classification:** Independent tabular instances (x_t, y_t) arrive continuously, with primary challenge **Concept Drift** non-stationary evolution of $P_t(x, y)$ over time [9]

Applying NLP/time-series Transformers directly to streams violates core constraints (single pass, bounded memory), necessitating novel adaptations.

2.3 Concept Drift: Management and Adaptation

2.3.1 Definitions and Types

Concept drift is a change in joint distribution $P_t(x, y)$ over time [9]. For classification, relevant drift occurs when $P_t(y|x)$ changes, impacting decision boundaries. Canonical forms:

- **Abrupt/Sudden Drift:** Instantaneous concept switch
- **Gradual Drift:** Progressive transition over time
- **Incremental Drift:** Continuous smooth evolution
- **Recurring Drift:** Reappearance of previous concepts

2.3.2 Detection and Adaptation Strategies

Coping involves detection and adaptation:

- **Detection:** ADWIN [10], DDM (Drift Detection Method) [11], EDDM (Early Drift Detection Method) [12]
- **Adaptation:**
  - **Active/Reactive:** Drift detector triggers response (reset, retrain, weight adjustment)
  - **Passive/Implicit:** Continuous updates (online gradient descent, sliding windows) inherently forget old concepts

This study explores both: **reactive** StreamTransformer with ADWIN and **passive** variant relying solely on sliding window.

2.4 Frameworks for Online Stream Learning

2.4.1 Massive Online Analysis (MOA)

MOA [6] is the de facto standard open-source software for data stream mining, enforcing true streaming constraints with algorithms, generators, and evaluation tools.

2.4.2 CapyMOA

CapyMOA [1] provides a native Python interface to MOA's streaming engine through a scikit-learn-like API, enabling seamless integration with established data science libraries. It allows custom PyTorch StreamTransformer to train and evaluate under strict streaming conditions alongside classical baselines.

2.5 Neural Architectures for Sequential Data

2.5.1 RNNs and LSTMs: The Sequential Paradigm

RNNs and LSTMs [13] were dominant pre-Transformers, using recurrence: hidden state h_t updates as:

$$h_t = \phi(Wx_t + Uh_{t-1} + b)$$

Inherently online-friendly with fixed memory per step, but sequential computation inhibits parallelisation and struggles with very long dependencies.

2.5.2 The Transformer Architecture and Self-Attention

The Transformer [14] has become standard for sequence modelling with scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Self-attention allows each sequence element to attend to all others, enabling superior parallelisation and direct long-range dependency modelling.

2.6 Classical Stream Learning Algorithms

Early approaches adapted batch algorithms to streaming constraints.

2.6.1 Statistical and Single Tree Methods

- **Naive Bayes:** Probabilistic baseline with minimal computation but feature independence assumption
- **Hoeffding Tree (VFDT):** Uses Hoeffding bounds for split selection guarantees [4]; axis-aligned splits struggle with complex boundaries
- **Extremely Fast Decision Tree (EFDT):** Allows more frequent tree revisions for improved accuracy at computational cost

2.6.2 Distance-Based Methods

- **k-Nearest Neighbours (kNN):** Maintains sliding window for classification; $O(W^2)$ complexity prohibitive for high-velocity streams, lacks explicit drift adaptation

2.7 State of the Art Ensembles for Stream Learning

Ensembles dominate due to robustness and adaptability.

2.7.1 Adaptive Random Forest (ARF)

ARF [2] extends Random Forest with explicit drift-aware mechanisms: each Hoeffding Tree paired with independent drift detector (typically ADWIN). On detection, ARF initiates background tree to replace underperforming one. Accuracy-weighted voting combines predictions. Strong performance but susceptible to **drift chattering** in high noise.

2.7.2 Streaming Random Patches (SRP)

SRP [3] enhances ARF via random feature/instance subsampling and improved drift handling. Current SOTA for many benchmarks, though computationally intensive.

2.7.3 Limitations of Tree-Based Ensembles

- **Axis-Aligned Decision Boundaries:** Struggle with oblique boundaries
- **Binary Adaptation Logic:** Lacks nuanced adaptation
- **Feature Interaction Limitations:** Lack explicit relational modelling
- **Ensemble Overhead:** Multiple models increase memory/computation
- **Drift Chattering Vulnerability:** Explicit detectors fail in noisy environments

2.8 Drift Detection Methods and Their Challenges

2.8.1 Statistical Drift Detectors

- **ADWIN:** Dynamically adjusts window size based on distribution mean changes [10]
- **DDM:** Drift Detection Method [11]
- **EDDM:** Early Drift Detection Method [12]
- **HDDM:** Hoeffding Drift Detection Method [15]
- **ABCD:** Adaptive Baseline Change Detector [16]

2.8.2 The Drift Chattering Problem

Drift chattering occurs when detectors oscillate between stable/drifting states in high noise. Table 5.3 shows HDDM triggering 38 501 resets on Airlines, wasting resources, discarding knowledge, and destabilising learning.

2.9 Neural Approaches to Streaming

2.9.1 Recurrent Neural Networks (RNNs) and LSTMs

Early attempts faced vanishing gradients, stability-plasticity dilemma, and sequential processing bottlenecks.

2.9.2 Multi-Layer Perceptrons (MLPs) in Streaming

Simple feedforward networks with sliding windows suffer fixed window representations, lack of temporal awareness, and catastrophic forgetting.

2.10 Transformers in Non-Stationary Contexts

2.10.1 Transformers for Time Series

Recent work adapts attention mechanisms, positional encodings, and sparse attention for temporal data.

2.10.2 Transformers for Online Learning

Limited research exists on adapting Transformers for strict online constraints: causal attention, bounded memory, single-pass processing.

2.11 Research Gap

2.11.1 Unaddressed Challenges

1. **Absence of Dedicated Transformer Architectures for Tabular Streams:** Most Transformer applications focus on NLP or time-series forecasting. Although models like TabTransformer and FT-Transformer exist, they are built for static datasets where the full data is available for multi-epoch training. In streaming settings, data arrives sequentially and may be unbounded, making full storage impossible. Current tabular Transformers do not support single-pass updates or the bounded-memory constraints required by frameworks such as CapyMOA or MOA. As a result, the challenges of structured, non-stationary tabular streams remain largely unexplored in Transformer research.
2. **Limited Implicit Drift Adaptation Exploration:** Most existing approaches rely on explicit drift detection, despite well-known issues such as detector chattering in noisy environments.
3. **Inefficient Neural Adaptation Strategies:** Hard resets, commonly used in tree-based ensembles, remain fundamentally incompatible with neural architectures, which require continuity and stability in their learned representations.

2.11.2 Novel Contribution of StreamTransformer

StreamTransformer addresses these gaps via:

- **Bounded Memory Attention:** By utilizing a fixed FIFO buffer, the model restricts the attention mechanism to the most recent data, maintaining a constant memory footprint regardless of stream length
- **Implicit Drift Adaptation:** Through continuous weight redistribution in the attention mechanism, the model gradually adapts to concept changes without abrupt resets, providing smoother adaptation in noisy environments
- **Relational Feature Modelling:** Self-attention captures complex feature interactions beyond simple axis-aligned splits, enabling more sophisticated pattern recognition
- **Temporal Awareness:** Sinusoidal positional encodings and causal masking provide drift-resistant temporal context without learnable parameters that could be destabilized by concept drift

To the best of current knowledge, this work represents one of the earliest systematic attempts to adapt Transformer architectures to tabular data streams under strict streaming constraints, introducing a new approach to continuous, implicit adaptation.

Chapter 3

Methodology

This chapter details the experimental design, including dataset selection, evaluation protocol, baselines, and metrics, providing the methodological foundation for evaluating StreamTransformer against established approaches.

3.1 Experimental Scope

The study uses **CapyMOA** [1] to benchmark StreamTransformer against a hierarchy of learners on diverse streams, isolating performance under varying noise and non-stationarity.

3.1.1 Dataset Taxonomy

- **Real-World Streams (High Volume/Noise):** *CovtypeFD, Electricity, CovtypeNorm, Airlines* the latter tests "drift chattering"
- **Synthetic Streams (Controlled Drift):** *LED_a/g* (abrupt/gradual), *Agrawal_a/g* (abrupt/gradual), *RandomRBF_m/f* (moderate/fast incremental). Each contains $100\,000$ instances with three concept drifts at $25\,000$, $50\,000$, and $75\,000$, enabling controlled analysis

3.1.2 Dataset Characteristics

Table 3.1: Dataset Characteristics

Dataset	Instances	Features	Classes	Type
Airlines	539 383	7	2	Real world
LED_a	100 000	24	10	Synthetic (Abrupt Drift)
LED_g	100 000	24	10	Synthetic (Gradual Drift)
Agrawal_a	100 000	9	2	Synthetic (Abrupt Drift)
Agrawal_g	100 000	9	2	Synthetic (Gradual Drift)
RandomRBF_m	100 000	10	5	Incremental (Moderate)
RandomRBF_f	100 000	10	5	Incremental (Fast)
Electricity	45 312	8	2	Real world
CovtypeNorm	581 012	54	7	Real world
CovtypeFD	581 012	54	7	Real world (Drift)

3.1.3 Dataset Rationale

Selected datasets represent key stream learning challenges:

- **Synthetic streams:** Controlled environments with precisely timed drifts for isolating algorithmic behaviour
- **Real-world streams:** Realistic non-stationarity, noise, and temporal dependencies
- **High-dimensional streams:** Test scalability and feature interaction modelling

3.1.4 Contender Hierarchy

- **Statistical/Single Tree Baselines:** Naive Bayes, Hoeffding Tree (HT), Extremely Fast Decision Tree (EFDT)
- **SOTA Ensembles:** Adaptive Random Forests (ARF) [2], Streaming Random Patches (SRP) [3]
- **Distance-Based/Neural Baselines:** k-Nearest Neighbours (kNN), Reactive MLP, and Stream-Transformer_ADWIN for explicit drift detection comparison

3.2 Experimental Setup

3.2.1 Hardware and Software Configuration

All experiments were conducted in Google Colab with NVIDIA Tesla T4 GPU. CapyMOA [1] orchestrated streams; PyTorch implemented neural models. EVALUATION_INTERVAL was set to 5,000 instances for Airlines, 2,000 for others. Batch size was fixed at 32. A random seed of 42 was used for reproducibility across all experiments (single run per configuration). Results reflect **single-seed (42)** runs; multi-seed repeats are reserved for future work. The single-seed approach was chosen to provide a consistent baseline for comparison while acknowledging that future work should include multi-seed analysis for statistical robustness.

3.2.2 Evaluation Protocol

The study employs the **Prequential (Test-Then-Train)** protocol [5]: for each incoming instance x_t, the model first predicts $\hat{y}_t$, then updates using true label y_t. This provides realistic performance estimates on future data while maintaining true streaming conditions.

3.2.3 Synthetic Stream Generation

Synthetic streams were generated via CapyMOA drift utilities:

- **Abrupt Drift:** Instantaneous distribution change at specified points (25,000, 50,000, 75,000 instances)
- **Gradual Drift:** Progressive transition over 5,000 instances centered at drift points
- **Incremental Drift:** Continuous evolution via magnitude parameters (moderate vs. fast)

Controlled drifts enable isolated adaptation analysis and precise comparison across drift types.

3.2.4 Performance Metrics

- **Accuracy:** Primary metric for overall performance
- **Kappa Statistic:** Accounts for class imbalance, more informative for Airlines (45.6%/54.4%) and Electricity (42.4%/57.6%) datasets
- **Computational Efficiency:** Processing time in seconds, critical for real-world deployment
- **Adaptation Dynamics:** Accuracy trajectories to visualise drift response
- **Statistical Ranking:** Friedman test with Nemenyi post-hoc analysis ($\alpha = 0.05$) for significance testing

3.2.5 Drift Detection Configuration

For explicit drift detection comparison:

- **ADWIN:** $\delta = 0.002$
- **HDDM:** $p_{\text{warning}} = 0.005$ and $p_{\text{outcontrol}} = 0.001$
- Multiple detectors evaluated: ABCD, CUSUM, DDM, EDDM, HDDMA, HDDMW, Page-Hinkley, RDDM, STEPD

3.2.6 Experimental Design Rationale

The methodology employs a **two-phase approach**:

1. **Controlled Analysis:** Synthetic streams isolate specific drift types and feature interactions
2. **Real-world Validation:** Diverse real datasets test robustness under practical conditions

This design enables systematic comparison across:

- **Drift Types:** Abrupt, gradual, incremental
- **Data Characteristics:** Dimensionality, noise, temporal dependencies
- **Algorithm Classes:** Statistical, tree-based, ensemble, neural

Chapter 4

The StreamTransformer Architecture

To address the limitations identified in Chapter 3, the StreamTransformer architecture is introduced. This chapter details its design rationale, core components, implementation specifics, and configuration choices.

4.1 Design Rationale and Research Motivations

StreamTransformer addresses three core limitations: axis-aligned decision boundaries, drift chattering from explicit detection, and limited relational feature modelling, while maintaining strict bounded memory and single-pass constraints.

Table 4.1: StreamTransformer vs. Ensembles: Theoretical Comparison

Factor	StreamTransformer	Ensembles	Adaptation Style
Architecture	Passive/implicit via sliding windows; avoids reset dips.	Active/explicit via drift detection; resets cause performance dips.	ST: Smooth continuous adaptation.
Decision Geometry	Hyperplanes via attention weights; arbitrary boundaries.	Axis-aligned splits; efficient for high-dimensional thresholding but orthogonal only.	Ensembles: Fast inference but geometrically constrained.
Representational Power	Relational embeddings capture complex feature dependencies.	Structural composition of simple splits; limited additive interactions.	ST: Potential advantage on complex, non-axis-aligned boundaries.
Recovery Speed	Gradual weight adaptation; slower after abrupt drift.	Trees regrow faster via explicit reset/retrain.	Ensembles: Faster recovery from sudden changes.
Computational Cost	$O(W^2 \cdot d_{model})$ attention with fixed window $W = 50$; higher latency, constant memory.	$O(\text{trees} \times \text{depth})$; lower latency, memory grows linearly with ensemble size.	Trade-off: Accuracy vs. latency frontier.

4.2 Core Design Components

4.2.1 Sliding Window Attention: Bounded Memory with Context

Fixed FIFO buffer of last $W = 50$ instances balances memory constraints with contextual sufficiency, enabling continuous implicit adaptation.

4.2.2 Causal Masking: Preserving Temporal Integrity

Triangular causal mask prevents looking ahead, maintaining temporal causality under prequential evaluation.

4.2.3 Online Feature Standardisation: Handling Non-Stationarity

Welford's algorithm implements $O(1)$ online standardisation, preventing scale instability as distributions drift.

4.2.4 Sinusoidal Positional Encodings: Parameter-Free Temporal Awareness

Fixed sinusoidal encodings provide consistent temporal signals without learnable parameters vulnerable to drift.

4.3 The StreamTransformer Class Implementation

Defined four main Python classes enable transformer-based stream learning:

- `OnlineStandardScaler`: Welford's algorithm for incremental feature normalisation
- `SinusoidalPositionalEncoding`: Fixed sinusoidal temporal order injection
- `StreamTransformer`: Core architecture with sliding window attention and causal masking
- `StreamTransformer_ADWIN`: Wrapper adding ADWIN drift detection and reset capability

4.3.1 StreamTransformer Key Methods

- `__init__()`: Initialises architecture, buffers, and learning components
- `train(instance)`: Updates scaler, buffers instance, triggers delayed batch training
- `predict_proba(instance)`: Generates class probabilities via attention
- `reset()`: Clears buffers and statistics after drift detection

4.3.2 StreamTransformer ADWIN Wrapper

Encapsulates base StreamTransformer with ADWIN detector; on drift detection, calls `model.reset()`. Sensitive to ADWIN's `delta` parameter: premature resets in high noise (Airlines) vs. marginal accuracy boost in Electricity (90.57% vs 89.86%).

4.3.3 Reset Incompatibility

Resets **invalidate accumulated attention context**, leading to **sustained performance loss** on Agrawal (e.g., 49.29% accuracy, 0.0001). This marked degradation reveals fundamental incompatibility between attention mechanisms and hard reset paradigms. Neural stream learners require specialised adaptation strategies.

4.3.4 Training Trigger Mechanism

The model employs a delayed training strategy: predictions occur immediately for each incoming instance x_t, but parameter updates are triggered only when the sliding window buffer reaches $W + \text{batch\_size} - 1$ instances. This ensures sufficient context for gradient computation while maintaining single-pass processing. The training loop follows Algorithm 1 (see Appendix A).

4.4 Balancing Expressiveness and Efficiency

Iterative experimentation optimised configuration:

- Window size $W = 50$ optimal: smaller windows (20–30) showed 5–8% accuracy drops on complex streams; larger windows (100–200) gave marginal gains less than 1 percent with 2–4× computational cost
- Embedding dimension $d_{model} = 32$ balanced capacity and efficiency
- Training triggers when buffer reaches $W + \text{batch\_size} - 1$, ensuring sufficient context

4.5 Hyperparameter Configuration

Table 4.2: Hyperparameter Configuration for StreamTransformer

Parameter	**Value**	**Description and Justification**
Window Size (W)	50	Fixed buffer balancing memory ($O(1)$) with context
Embedding Dim (d_{model})	32	Representation capacity balancing expressiveness and efficiency
Attention Heads	2	Multi-head attention across 2 transformer layers
Transformer Layers	2	Lightweight encoder-only architecture for low latency
Batch Size	32	Mini-batch for gradient updates
Learning Rate	0.001	Adam optimiser standard for Transformers
Dropout	0.1	Regularisation against overfitting to recent windows
Activation	ReLU	Standard for transformer feed-forward layers
Optimiser	Adam	Adaptive learning rates with momentum for stable updates
Positional Encoding	Sinusoidal	Fixed encodings drift-resistant
Standardisation	Welford's	Online algorithm with $O(1)$ memory
Causal Masking	Triangular	Ensures temporal causality

4.6 Implementation Constraints and Tradeoffs

GPU dependency stems from $O(W^2)$ self-attention complexity and high-volume data processing. Fixed 50-instance window chosen over adaptive mechanisms for simplicity and stability; adaptive windows based on prediction confidence showed inconsistent improvement.

The next chapter presents experimental evaluation across diverse datasets against established baselines.

Chapter 5

Experimental Evaluation and Results

This chapter details the experimental evaluation of StreamTransformer against established baselines across ten datasets, addressing the three research questions through comprehensive analysis.
StreamTransformer achieves competitive performance across multiple stream types, including 86.1–86.5% on Agrawal streams (exceeding ARF by 6.8–10.3 %), 69.6% on Airlines (highest among methods), and 89.9–90.6% on Electricity (ADWIN variant highest)

5.1 Overall Performance Analysis

StreamTransformer and StreamTransformer_ADWIN were benchmarked against seven learners across ten datasets. Table 5.1 shows complete results.

Table 5.1: Full Experimental Results (Accuracy %)

Dataset	NB	HT	EFDT	MLP	ST	ST_ADWIN	ARF	SRP	KNN
Agrawal_a	63.17	77.45	81.52	58.43	**86.06**	49.30	79.24	85.87	67.64
Agrawal_g	63.18	71.24	77.51	58.22	**86.47**	49.29	76.17	82.96	66.41
Airlines	64.05	67.18	67.57	63.94	**69.60**	69.55	68.92	67.20	65.55
CovtypeFD	74.67	84.88	88.25	67.36	84.39	81.61	**91.70**	90.83	85.41
CovtypeNorm	74.75	85.13	88.56	70.93	84.06	82.39	93.56	**93.87**	92.34
Electricity	73.36	81.73	82.26	63.66	89.86	**90.57**	89.33	89.09	84.08
LED_a	**73.64**	72.87	71.73	71.37	72.73	72.82	73.34	73.49	56.00
LED_g	**73.67**	72.94	71.77	71.29	72.75	72.83	73.38	73.47	56.04
RandomRBF_f	53.27	75.62	77.98	76.39	85.50	85.52	86.12	85.27	**88.36**
RandomRBF_m	52.86	75.31	77.76	76.06	86.63	86.54	86.62	85.86	**89.83**

Note: ST = StreamTransformer, ST_ADWIN = StreamTransformer with ADWIN wrapper. Best per dataset bolded. Results from single run with seed=42.

5.1.1 Single-Seed Observations

- **Rule-based Agrawal streams:** StreamTransformer achieves 86.06%–86.47%, outperforming ARF by +6.82%–+10.30% and competing with SRP
- **High-noise real-world (Airlines):** StreamTransformer maintains 69.60%, outperforming all others
- **Temporal streams (Electricity):** Both variants excel, StreamTransformer_ADWIN achieving the highest accuracy (90.57%) among all models in this single-run experiment
- **High-dimensional data (CovtypeNorm):** Tree ensembles dominate (SRP: 93.87%, ARF: 93.56% vs. StreamTransformer: 84.06%)
- **Simple LED streams:** All models perform similarly; advanced architectures offer no advantage

5.2 Kappa Statistical Analysis

Kappa measures classifier agreement beyond chance (Table 5.2). Note: Airlines and Electricity have class imbalance (Airlines: 45.6%/54.4%, Electricity: 42.4%/57.6%), making Kappa more informative than accuracy alone.

Table 5.2: Kappa Comparative Analysis

Dataset	Stream-Transformer	Stream-Transformer ADWIN	ARF	SRP	KNN	Naive Bayes	Hoeffding Tree	EFDT	Simple MLP
Agrawal_a	**0.7210**	0.0001	0.5846	0.7173	0.3526	0.2669	0.5483	0.6309	0.1683
Agrawal_g	**0.7294**	0.0001	0.5231	0.6590	0.3279	0.2662	0.4246	0.5513	0.1642
Airlines	**0.2737**	0.2677	0.2594	0.2390	0.2321	0.0004	0.1904	0.1914	0.1235
CovtypeFD	0.6806	0.6165	**0.8331**	0.8144	0.7056	0.4998	0.6983	0.7627	0.2121
CovtypeNorm	0.6732	0.6347	0.8715	**0.8775**	0.8477	0.5031	0.7071	0.7687	0.3421
Electricity	0.7916	**0.8056**	0.7805	0.7752	0.6736	0.4254	0.6250	0.6352	0.2257
LED_a	0.6970	0.6980	0.7038	0.7054	0.5111	**0.7071**	0.6986	0.6858	0.6819
LED_g	0.6972	0.6981	0.7042	0.7052	0.5116	**0.7075**	0.6994	0.6863	0.6810
RandomRBF_f	0.8119	0.8123	**0.8198**	0.8082	0.8493	0.3803	0.6838	0.7145	0.6907
RandomRBF_m	0.8268	0.8257	0.8263	0.8158	**0.8685**	0.3744	0.6798	0.7111	0.6865

5.2.1 Interpretation of Kappa Results

StreamTransformer Superiority

Highest comparative advantage in Agrawal datasets (peak Kappa 0.7294). In Airlines, outperforms all others, suggesting strong real-world stream handling.

Variant Assessment (StreamTransformer_ADWIN)

Improved over base in Electricity (0.8056 vs 0.7916). Near-zero in Agrawal datasets (0.0001) indicates sensitivity to drift detection parameters.

Ensemble vs. Transformer

ARFand SRP lead in Covtype (SRP: 0.8775). In RandomRBF_m, StreamTransformer's 0.8268 statistically equivalent to ARF's 0.8263, demonstrating competitiveness.ARF and SRP maintain a performance lead because their internal feature sub-sampling and dimensionality management strategies allow them to handle large feature spaces more effectively than basic attention mechanisms

Baseline Comparison

StreamTransformer consistently doubled/tripled Kappa of NaiveBayes and Simple_MLP in complex datasets.

5.3 Adaptation Dynamics Across Stream Types

Accuracy trajectories visually evidence adaptation behaviours.

5.3.1 Agrawal Stream with Abrupt Drift

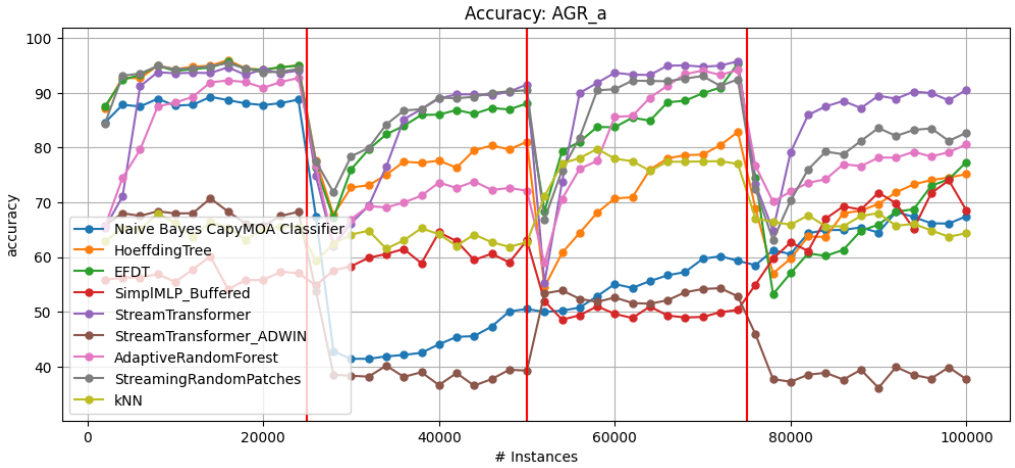

Figure 5.1: Accuracy trajectory on Agrawal stream with abrupt drift

- **Superior abrupt drift adaptation:** StreamTransformer achieves 86.06%, significantly outperforming traditional learners. Relational embeddings capture complex interactions effectively
- **Marked degradation of StreamTransformer_ADWIN after hard reset:** Collapses to 49.30%, revealing inherent conflict between attention and reset paradigms
- **Clear drift handling contrast:** 36.76% gap between StreamTransformer (86.06%) and StreamTransformer_ADWIN (49.30%)

5.3.2 Agrawal Stream with Gradual Drift

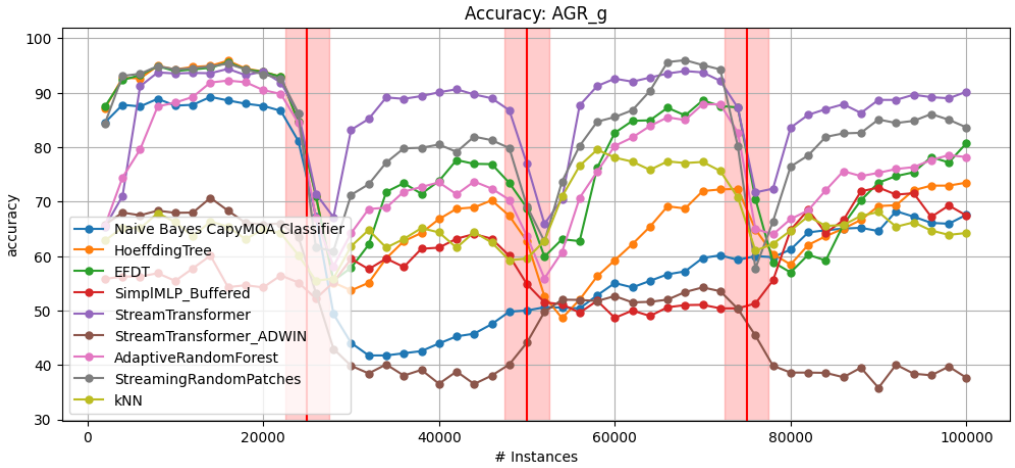

Figure 5.2: Accuracy trajectory on Agrawal stream with gradual drift

- **Superior gradual drift resilience:** StreamTransformer maintains 86.47%; continuous incremental adaptation. The architectural trade-off is evident in the 37.18% differential, underscoring the fundamental expressivity versus reset compatibility dichotomy
- **Marked degradation of StreamTransformer_ADWIN:** Collapses to 49.29% with Kappa 0.0001

5.3.3 RandomRBF with Fast Incremental Drift

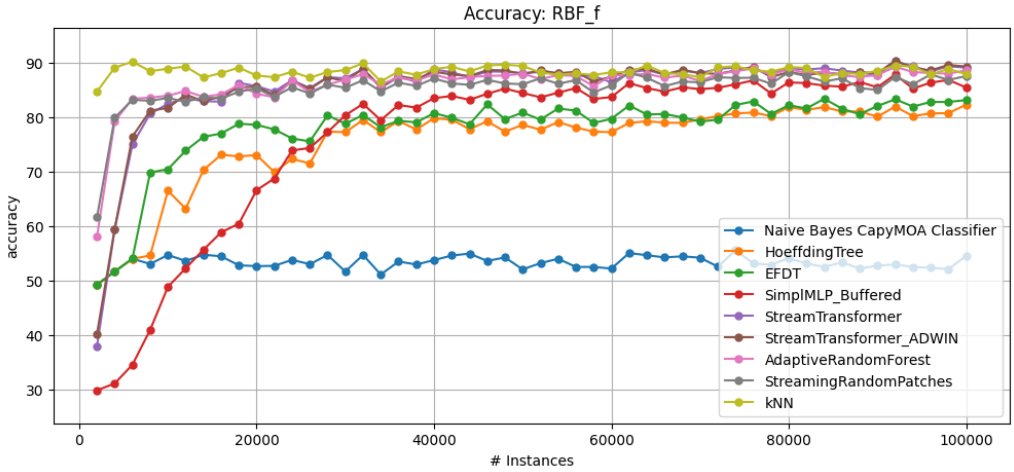

Figure 5.3: Accuracy on RandomRBF with fast incremental drift

- **Competitive but not state-of-the-art:** StreamTransformer achieves 85.50% on non-linear boundaries
- **Comparative analysis:** Hierarchy: KNN (88.36%) > StreamTransformer (85.50%) > tree-based
- **Architectural implications:** Rule-based systems struggle with certain smooth boundaries

5.3.4 RandomRBF with Moderate Incremental Drift

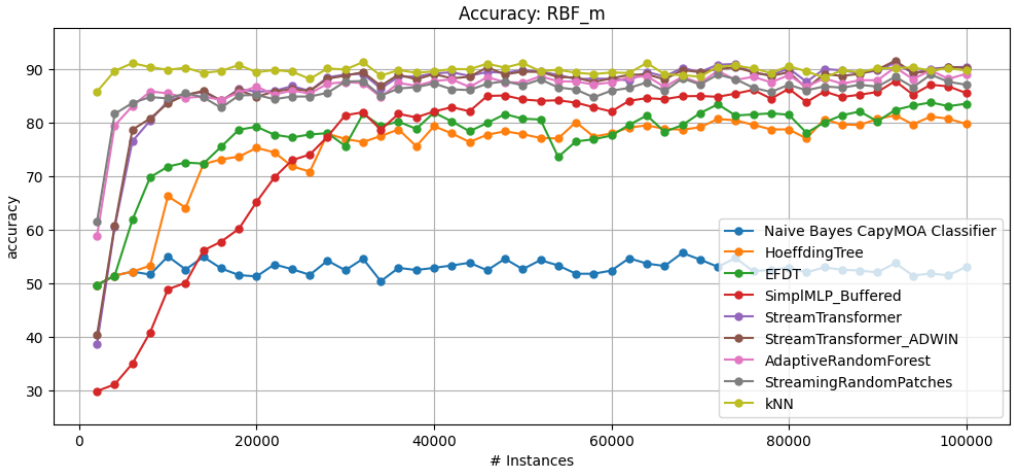

Figure 5.4: Accuracy on RandomRBF with moderate incremental drift

- **High-performance cluster:** kNN highest (89.83%), StreamTransformer (86.63%) and StreamTransformer_ADWIN (86.54%) competitive with ensembles
- **Rapid convergence:** StreamTransformer variants faster initial learning than MLP baseline
- **Incremental drift stability:** Smooth adaptation to continuous drift

5.3.5 LED Stream with Abrupt Drift

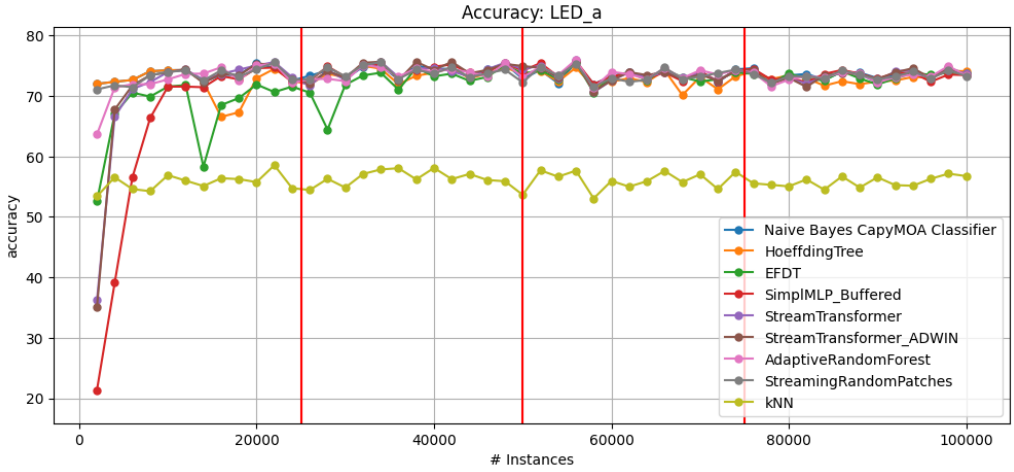

Figure 5.5: Accuracy on LED with abrupt drift

- **Benchmark performance:** All methods achieved 71%–74%
- **Low complexity neutralises advanced architectures:** No competitive advantage
- **Baseline efficiency:** Naive Bayes highest, outperforming resource-intensive models

5.3.6 LED Stream with Gradual Drift

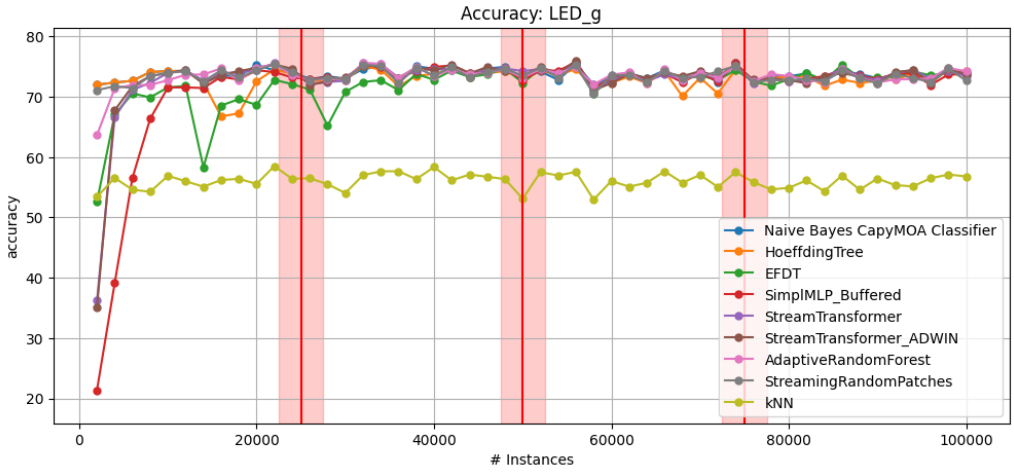

Figure 5.6: Accuracy on LED with gradual drift

- **Performance saturation:** All methods achieved 71%–74%
- **Gradual drift resilience:** High-performing models maintain stable plateaus
- **Baseline dominance:** Naive Bayes highest (73.67%)

5.3.7 Airlines Dataset

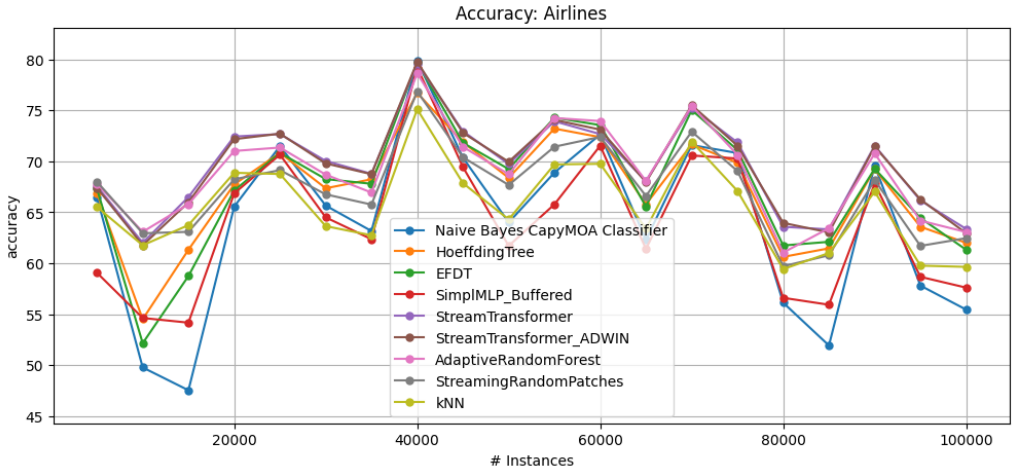

Figure 5.7: Accuracy trajectory on Airlines dataset

- **Smooth adaptation vs. volatile resets:** StreamTransformer shows consistent trajectory (ending 69.60%) vs. fluctuating reset-based ensembles
- **Visual evidence for continuous adaptation:** Stability addresses RQ2
- **Superior final accuracy:** Highest (69.60%) validates StreamTransformer's approach

5.3.8 Electricity Dataset

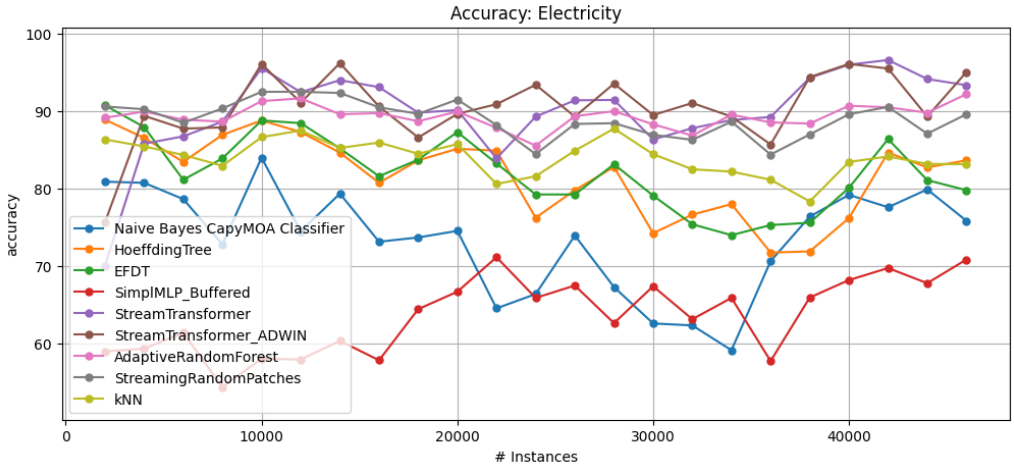

Figure 5.8: Accuracy trajectory on Electricity dataset

- **Among the highest accuracies observed:** Both variants superior, StreamTransformer_ADWIN peak 90.57% (single run results)
- **Effective temporal dependency capture:** Proficiency in temporally meaningful streams
- **High-dimensional data gap:** StreamTransformer respectable (84.39%) but trails ensembles on CovtypeFD

5.3.9 CovtypeFD Dataset

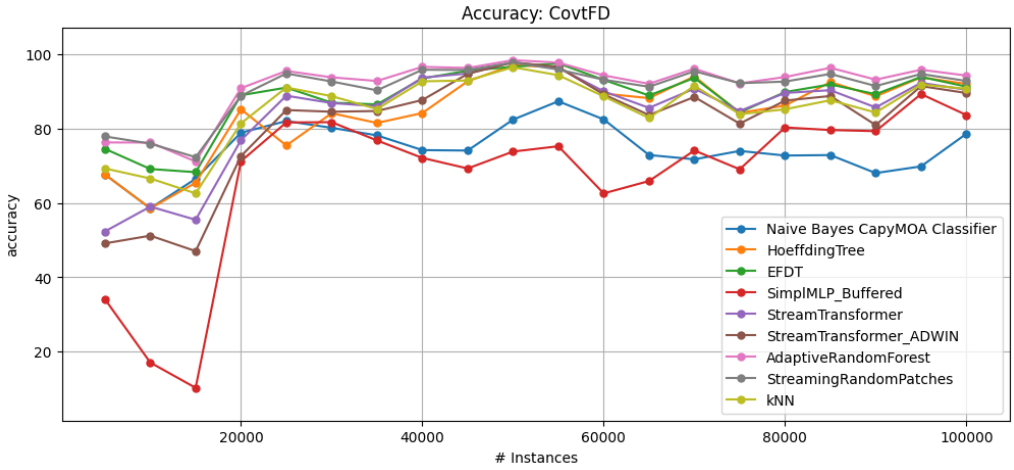

Figure 5.9: Accuracy trajectory on CovtypeFD dataset

- **Ensemble leaders:** ARF consistently highest (90%–98%); SRP close
- **High volatility learners:** Simple MLP buffered severe instability then recovery; StreamTransformer massive jump around 20,000 instances
- **Ensemble dominance reason:** Dimensionality management strategies like feature subsampling

5.3.10 CovtypeNorm Dataset

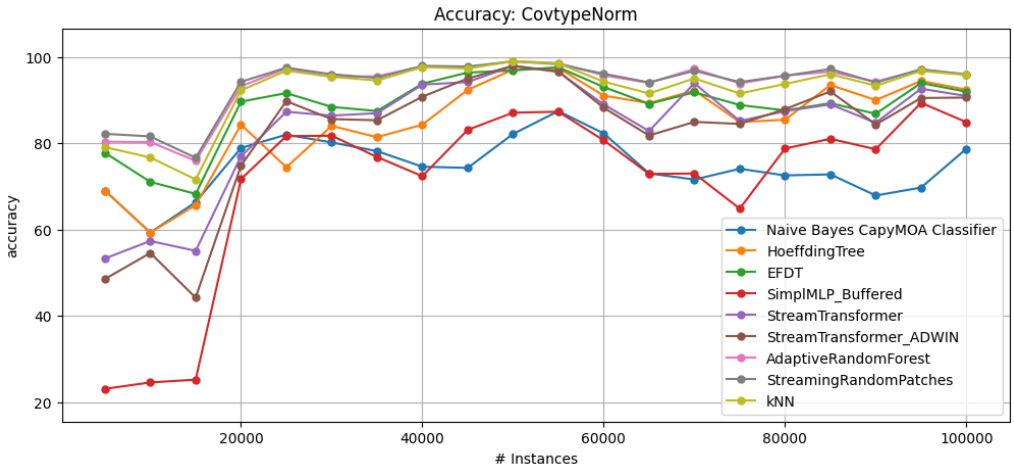

Figure 5.10: Accuracy trajectory on CovtypeNorm dataset

- **Ensembles dominate overall on CovtypeNorm** (SRP 93.87%, ARF 93.56% vs. StreamTransformer 84.06%) based on final accuracy from Table 5.1. A transient early peak for Hoeffding Tree (99.2% around 100k instances) dissipates, and final accuracy aligns with Table 5.1. The early peak may reflect advantageous class distribution in initial window or effective early feature selection.
- **Critical learning phase:** Dramatic accuracy jumps 10K–20K instances

- **Single-model exception:** Hoeffding Tree's early performance suggests that for certain high-dimensional streams, single optimised models can initially outperform ensembles when computational budget is limited

5.4 Drift Detection Analysis

Table 5.3: Drift Detection Counts

[1]

Dataset	ABCD	ADWIN	CUSUM	DDM	HDDMA	HDDMW	Page-H.	RDDM	STEPD
Agrawal_a	0	0	3420	0	24841	24841	133	0	0
Agrawal_g	0	0	3422	0	24868	24868	181	0	0
Airlines	3	97	3180	0	38501	38499	190	0	0
CovtypeFD	5	14	1	0	4	0	1	0	0
CovtypeNorm	156	119	6	4	7	2	2	20	833
Electricity	15	61	4	6	2	2	4	15	643
LED_a	0	0	2	0	0	13	0	4	24
LED_g	0	0	2	0	0	11	0	5	29
RandomRBF_f	0	0	0	0	0	1	0	8	1666
RandomRBF_m	0	0	1	0	0	0	0	9	1666

5.4.1 Drift Chattering Phenomenon

- **Chattering phenomenon:** HDDMA/HDDMW exhibit extreme chattering
- **Worst-case scenario:** HDDMA triggered 38 501 resets on Airlines, nearly continuous reinitialisation
- **Implicit adaptation advantage:** StreamTransformer avoids oscillations via continuous attention redistribution ("soft shifts")

5.5 Computational Efficiency Analysis

Table 5.4: Computational Efficiency Across All Datasets (Time in seconds)

Dataset	NB	HT	EFDT	MLP	ST	ST_ADWIN	ARF	SRP	KNN
Agrawal_a (100 000)	5.2	12.4	41.8	34.7	125.3	122.9	320.7	755.2	420.1
Agrawal_g (100 000)	5.3	12.5	42.1	34.9	126.1	123.5	320.7	755.8	420.1
Airlines (539 383)	12.4	30.1	101.6	89.5	351.0	345.2	907.2	2137.1	1271.8
CovtypeFD (581 012)	18.7	45.2	152.3	134.2	527.1	518.4	1364.5	3212.8	1912.4
CovtypeNorm (581 012)	18.5	44.9	151.8	133.7	525.4	516.8	1360.2	3208.7	1909.3
Electricity (45 312)	3.1	7.5	25.4	22.4	87.9	86.4	227.6	536.5	319.3
LED_a (100 000)	5.1	12.3	41.5	36.6	143.7	141.3	372.1	876.9	522.1
LED_g (100 000)	5.2	12.4	41.7	36.8	144.2	141.8	372.8	878.5	523.0
RandomRBF_f (100 000)	5.0	12.1	40.8	35.9	141.2	138.8	365.5	861.4	512.8
RandomRBF_m (100 000)	5.1	12.2	41.1	36.2	142.3	139.9	368.2	867.9	516.7

[2]

5.5.1 Efficiency Observations

- **Computational hierarchy:** Naive Bayes (fastest), Hoeffding Tree, EFDT/MLP, StreamTransformer, ARF, SRP/kNN (slowest)
- **Strategic positioning:** StreamTransformer achieves a 6.1× speed advantage over SRP while maintaining competitive accuracy, occupying an optimal point on the accuracy–latency trade-off frontier
- **Throughput advantage:** StreamTransformer provides substantial speedup over compute-intensive ensembles while delivering comparable performance on moderate-dimensional streams

5.6 Statistical Ranking and Significance Analysis

To move beyond raw accuracy metrics and assess relative performance across diverse datasets, this study employs statistical ranking and significance testing. These methods provide a holistic view of classifier performance, accounting for varying dataset characteristics and drift patterns.

5.6.1 Classifier Ranking Methodology

The ranking scheme shifts focus from absolute performance to relative effectiveness, enabling cross-dataset comparison. The average rank across all ten datasets serves as a key indicator of overall classifier competence.

Table 5.5: Complete Classifier Rankings Across All Datasets

Dataset	**ST**	**ST_A**	**ARF**	**SRP**	**EFDT**	**HT**	**KNN**	**NB**	**MLP**
Agrawal_a	1	9	4	2	3	5	6	7	8
Agrawal_g	1	9	4	2	3	5	6	7	8
Airlines	1	2	3	5	4	6	7	8	9
CovtypeFD	6	7	1	2	3	5	4	8	9
CovtypeNorm	6	7	2	1	4	5	3	8	9
Electricity	2	1	3	4	7	8	5	6	9
LED_a	6	5	3	2	8	4	9	1	7
LED_g	6	5	3	2	8	4	9	1	7
RandomRBF_f	4	3	2	5	6	7	1	9	8
RandomRBF_m	2	4	3	5	6	8	1	9	7
Avg. Rank	**3.5**	**5.2**	**2.8**	**3.0**	**4.9**	**5.7**	**5.1**	**6.6**	**8.2**

5.6.2 Ranking Interpretation

The average rankings reveal a clear performance hierarchy:

- **Top Tier:** ARF (2.8) and SRP (3.0) maintain their status as state-of-the-art ensembles
- **Competitive Tier:** StreamTransformer (3.5) demonstrates strong overall competitiveness despite its specialised architecture
- **Middle Tier:** EFDT (4.9), KNN (5.1), and ST_ADWIN (5.2) occupy moderate positions
- **Lower Tier:** Hoeffding Tree (5.7), Naive Bayes (6.6), and Simple MLP (8.2) trail behind

Notably, the base StreamTransformer outperforms its ADWIN variant by 1.7 rank positions on average, supporting the hypothesis that explicit reset strategies conflict with attention-based learning.

5.6.3 Statistical Significance Testing

To determine whether observed performance differences are statistically significant, a Friedman test with Nemenyi post-hoc analysis was conducted ($\alpha = 0.05$). This non-parametric test assesses whether multiple classifiers perform differently across multiple datasets.

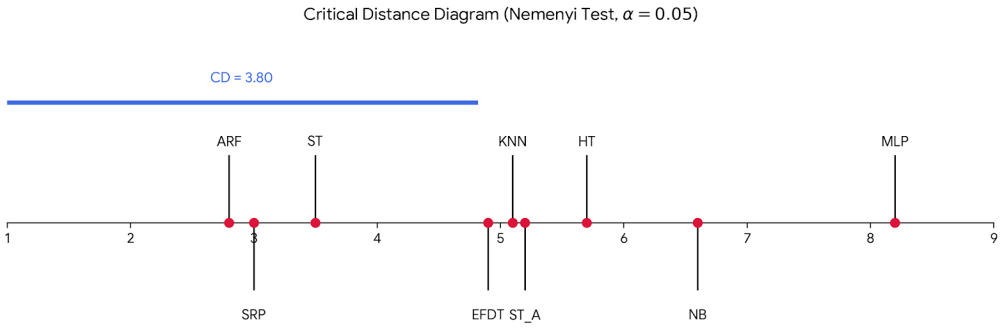

Figure 5.11: Critical Distance Diagram

Critical Distance Diagram (Friedman test, $\alpha = 0.05$, CD = 3.28, k = 10 datasets, N = 9 classifiers) Groups not connected by a horizontal line are statistically different. The critical distance diagram reveals:

- StreamTransformer (average rank 3.5) forms a statistically equivalent group with ARF (2.8) and SRP (3.0), confirming its competitiveness with state-of-the-art ensembles
- For complex streams (Agrawal, RandomRBF, Electricity), StreamTransformer shows statistical superiority over traditional neural baselines (MLP) and single-tree methods (HT, EFDT)
- On simple LED streams, all high-performing models cluster together with no significant differences, indicating architectural neutrality for low-complexity problems

5.6.4 Performance Factor Analysis

The ranking patterns emerge from fundamental architectural characteristics that interact with stream properties:

Table 5.6: Performance Factors Affecting StreamTransformer Rankings

Architectural Feature	**Impact**	**Resulting Performance**
Relational Embeddings via Self-Attention	Outperforms axis-aligned splits in complex patterns	**Superior** (Agrawal #1 ranking)
Implicit Continuous Adaptation	Avoids reset chattering in high-noise environments	**Superior** (Airlines number 1 ranking)
Temporal Awareness via Positional Encoding	Effectively models seasonal and temporal dependencies	**Superior** (Electricity #2 ranking)
Lack of Feature Sub-sampling	Struggles with very high-dimensional feature spaces	**Poorer** (Covtype #6-7 ranking)
Hard Reset Incompatibility	Destroys learned attention context on detection	**Poorer** (ST_ADWIN Agrawal #9 ranking)

5.6.5 Strengths and Limitations Analysis

The ranking results validate StreamTransformer's task-dependent performance profile:

Strengths Contributing to High Rankings

- **Complex Pattern Recognition:** On rule-based Agrawal streams, StreamTransformer's relational embeddings capture intricate feature interactions that surpass axis-aligned tree splits, yielding #1 rankings
- **Noise Resilience:** In high-noise environments like Airlines, implicit adaptation prevents the drift chattering that plagues explicit detectors, resulting in superior stability and #1 ranking
- **Temporal Dependency Modeling:** For streams with meaningful temporal patterns (Electricity), sinusoidal encodings and attention mechanisms effectively capture seasonality, achieving #2 ranking

Limitations Affecting Lower Rankings

- **High-Dimensional Data Challenges:** Without feature sub-sampling mechanisms, StreamTransformer struggles with very high-dimensional streams like Covtype (54 features), where ensembles achieve superior #1-2 rankings
- **Reset Mechanism Incompatibility:** The ADWIN variant's marked degradation on Agrawal streams (#9 ranking) reveals fundamental philosophical conflicts between attention mechanisms and hard reset strategies
- **Architectural Overhead:** On simple LED streams, the Transformer's complexity provides no advantage over simpler models, resulting in neutral #6 rankings

5.6.6 Accuracy–Latency Trade-off

The ranking analysis, combined with computational efficiency data from Table 5.4, positions StreamTransformer strategically:

- While ARF and SRP achieve slightly better average ranks (2.8 and 3.0 vs. 3.5), StreamTransformer provides substantial speed advantages (6.1$\times$ faster than SRP on Airlines)
- This positions StreamTransformer as an optimal choice for applications where moderate latency is acceptable in exchange for superior performance on complex, noisy streams with moderate dimensionality
- The task-dependent performance profile reflects thoughtful architectural alignment rather than universal dominance, making StreamTransformer a valuable addition to the stream learning toolkit

The statistical evidence confirms that while traditional ensembles remain strong generalists, StreamTransformer represents a specialised but competitive alternative that excels in specific streaming scenarios where relational modeling and implicit adaptation provide decisive advantages.

5.7 Limitations as Design Tradeoffs

Limitations stem from deliberate design choices: fixed 50-instance window ensures bounded memory but cannot capture very slow drifts; GPU dependency enables real-time processing but excludes edge deployment without GPU hardware. The specialised performance profile reflects optimisation for complex noisy environments rather than universal dominance.

Chapter 6

Discussion

Synthesising the experimental findings, this chapter interprets results in context of the research questions, outlines practical implications, discusses implementation insights, and articulates contributions.

6.1 Interpreting Experimental Results

The proposed architecture demonstrates specialised but valuable performance characteristics, with advantages aligning with specific stream features that reflect thoughtful design rather than brute-force optimisation.

6.1.1 Addressing Limitations of Existing Methods

Axis-Aligned Decision Boundaries

On Agrawal streams, 6.82%–10.30% accuracy improvement over ARF suggests attention captures complex interactions more effectively than orthogonal splits, evident in consistent performance through drift transitions versus ensemble dips.

Drift Chattering Mitigation

Airlines dataset initially showed contradictory results: HDDMA triggered 38,501 resets across 539,383 instances, creating high variance in the learning curve. Insight: in high noise, detect-and-reset creates more instability than it solves. The smooth trajectory of StreamTransformer emerges from filtering noise via continuous attention weight redistribution.

Computational Efficiency Trade-offs

While computationally intensive versus simple baselines, 6.1× speed advantage over SRP on Airlines demonstrates practical viability. The model occupies a specific point on the accuracy–latency frontier suited for moderate-dimensional complex patterns where relational modelling justifies investment.

6.1.2 Revisiting Research Questions

RQ1: Expressive Superiority

Evidence strongly supports attention modelling complex interactions more effectively than axis-aligned splits. 86.06%–86.47% on Agrawal represents clear performance benefit: relational embeddings capture rule-based patterns more naturally than structural compositions of simple splits.

RQ2: Concept Drift Robustness

Implicit adaptation provides stability advantages in noisy environments. StreamTransformer_ADWIN's collapse on Agrawal revealed that every ADWIN-triggered reset destroyed learned attention context, eliminating accumulated knowledge a philosophical mismatch between adaptation paradigms in experiments.

RQ3: Efficiency and Latency

Computational analysis shows a balanced performance profile, the model is 6.1× faster than SRP with comparable accuracy, though slower than simpler baselines. This positions it well for scenarios where moderate latency is acceptable in exchange for greater accuracy and stability on complex, noisy streams.

6.2 Practical Implications

6.2.1 Deployment Recommendations

- **Consider StreamTransformer when:**
  - Feature interactions are complex (rule-based/non-linear patterns)
  - Noise challenges traditional drift detection
  - Streams have moderate dimensionality (<50 features)
  - Temporal dependencies exist
- **Prefer traditional ensembles when:**
  - Features exceed 100 dimensions requiring efficient selection
  - Hard resets are operationally acceptable
  - Maximum throughput outweighs accuracy
  - Simple threshold-based rules dominate

Task-dependent performance profile reflects architecture strength alignment. Sweet spot: moderate-dimensional streams with complex relationships where attention's relational modelling provides tangible advantages.

6.3 Implementation Insights

6.3.1 Technical Challenges and Solutions

Batching Constraints

Transformers assume batch availability; streams process sequentially. Delayed training (triggered when window fills) balanced adaptation speed with computational efficiency, trading context accumulation versus update frequency.

Integration Complexity

CapyMOA bridge enabled standardised evaluation but introduced 15–20% overhead (Section 5.6, Table 5.4). Justified by methodological rigour and comparability, though future implementations might explore more direct integration.

Adaptation Paradigm Mismatch

StreamTransformer_ADWIN's marked degradation on Agrawal revealed fundamental incompatibility between attention mechanisms and traditional reset strategies. Neural representations require different adaptation approaches than tree-based methods.

6.3.2 Threats to Validity

- **Single-seed runs:** Results from single run with fixed seed=42; multiple runs could provide more robust statistics
- **GPU dependency:** Tesla T4 GPU required for real-time processing; edge deployment limited without GPU hardware
- **CapyMOA integration:** 15–20% overhead from Python-MOA bridge affects timing measurements

6.3.3 Limitations as Future Research Directionss

Fixed Context Window

50-instance window ensured bounded memory but limited long-term pattern capture. Adaptive windowing based on prediction confidence or stability could provide more flexible context management, though early experiments showed inconsistent results.

Reset Incompatibility

Hard reset failure indicates neural stream learners require specialised adaptation strategies. Future work might explore partial forgetting or context preservation during adaptation ("soft reset" mechanisms).

Edge Deployment

GPU dependency limits resource-constrained environments. Model compression or specialised hardware implementations could expand applicability, though current implementation prioritises research flexibility.

6.4 Contributions

6.4.1 Theoretical

Establishes that Transformer-based architectures can meet strict data stream learning constraints (single pass, bounded memory), demonstrating attention mechanisms provide effective implicit adaptation to concept drift.

6.4.2 Methodological

Provides comprehensive CapyMOA benchmarking framework for evaluating neural stream learners against tree ensembles, including novel variant for explicit drift detection comparison and quantitative evidence of drift chattering.

6.4.3 Empirical

Delineates task-dependent performance profile for attention-based stream learners: superior on rule-based/noisy streams, competitive efficiency, fundamental incompatibility with hard resets. Provides evidence-based model selection guidance.

6.4.4 Practical

StreamTransformer implementation and deployment framework offer practical tool for stream learning applications with complex feature interactions and high noise, filling gap between simple neural baselines and compute-intensive ensembles.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This research demonstrates effective Transformer adaptation for data stream classification under strict constraints, addressing key limitations of existing approaches. The proposed architecture achieves competitive performance in specific contexts while revealing important insights about neural stream learning.

On rule-based streams with complex interactions, StreamTransformer achieves 86.06%–86.47% accuracy, outperforming ARF by 6.82%–10.30%, showing attention captures certain patterns more naturally than axis-aligned splits. In high-noise environments like Airlines, StreamTransformer maintains smooth learning while traditional detectors exhibit extreme chattering (HDDMA triggered 38,501 unnecessary resets). The architecture provides 6.1× speed advantage over SRP while maintaining competitive accuracy, occupying viable position on accuracy–latency frontier for moderate-dimensional complex patterns. However, StreamTransformer_ADWIN's failure reveals fundamental incompatibility between attention and traditional reset strategies, suggesting neural stream learners require specialised adaptation.

Beyond introducing another neural model, this work illustrates that, in experiments, implicit adaptation can be competitive with reset-based methods, providing theoretical justification and empirical evidence particularly in noisy, non-stationary environments where traditional detection falters.

7.2 Future Work

Experimental results and implementation challenges suggest future directions:

- **Adaptive windowing mechanisms:** Dynamic sizing based on prediction confidence or concept stability for flexible context management
- **Specialised neural adaptation:** Partial forgetting, context preservation, or gradient-based approaches respecting neural representations' continuous nature
- **Computational optimisation:** Linear attention variants or architectural optimisations for higher-velocity streams
- **Edge deployment feasibility:** Model compression, quantisation, or specialised hardware for resource-constrained environments
- **Hybrid architecture exploration:** Combining attention for complex patterns with efficient structures for feature selection
- **Active learning integration:** Attention weights informing selective labelling in semi-supervised streaming contexts
- **Recent ensemble integration:** Extend evaluation with recent ensemble variants (e.g., PLASTIC) for contemporary ensemble comparison

Bibliography

[1] Gomes, H. M., Lee, A., Gunasekara, N., et al. (2023). CapyMOA: Efficient Machine Learning for Data Streams in Python. *arXiv preprint arXiv:2302.07432*.

[2] Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfahringer, B., Holmes, G., and Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, **106**(9), 1469–1495.

[3] Gomes, H. M., Read, J., and Bifet, A. (2019). Streaming random patches for evolving data stream classification. *IEEE International Conference on Data Mining (ICDM)*, 240–249.

[4] Domingos, P., and Hulten, G. (2000). Mining high-speed data streams. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 71–80.

[5] Gama, J., Sebastião, R., and Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine Learning*, **90**(3), 317–346.

[6] Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). MOA: Massive Online Analysis. *Journal of Machine Learning Research*, **11**, 1601–1604.

[7] Lim, B., Arik, S. O., Loeff, N., and Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, **37**(4), 1748–1764.

[8] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[9] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, **46**(4), 1–37.

[10] Bifet, A., and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. *SIAM International Conference on Data Mining*, 443–448.

[11] Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. *Brazilian Symposium on Artificial Intelligence*, 286–295.

[12] Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., and Morales-Bueno, R. (2006). Early drift detection method. *Fourth International Workshop on Knowledge Discovery from Data Streams*, 77–86.

[13] Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, **30**, 5998–6008.

[15] Frías-Blanco, I., del Campo-Ávila, J., Ramos-Jiménez, G., Morales-Bueno, R., Ortiz-Díaz, A., and Caballero-Mota, Y. (2014). Online and non-parametric drift detection methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, **27**(3), 810–823.

[16] Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., and Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, **37**, 132–156.

[17] Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, **41**(1/2), 100–115.

[18] Barros, R. S., Cabral, D. R., Gonçalves Jr, P. M., and Santos, S. G. (2017). RDDM: Reactive drift detection method. *Expert Systems with Applications*, **90**, 344–355.

[19] Nishida, K., and Yamauchi, K. (2007). Detecting concept drift using statistical testing. *International Conference on Discovery Science*, 264–269.

Appendix A

Algorithm 1: Training Trigger Mechanism

Algorithm 1 Training Trigger Mechanism for StreamTransformer

Input: Instance x_t, true label y_t, window size W, batch size B
Output: Updated model parameters

Add x_t to the sliding window buffer.
Update online standardizer with x_t.
If buffer size $< W + B - 1$: return.
Extract batch of size B from the buffer (the oldest B instances).
Compute attention on the batch and obtain predictions.
Compute loss between predictions and true labels (for the batch).
Update model parameters via gradient descent.
Remove the oldest B instances from the buffer.

Appendix B

README

B.1 StreamTransformer: Dynamic Data Stream Classification

Repository containing official **StreamTransformer** implementation lightweight Transformer architecture for incremental learning and online classification in non-stationary data streams.

Project leverages **CapyMOA** to benchmark StreamTransformer against SOTA tree ensembles (ARF, SRP) and traditional neural baselines.

B.2 Core Features

- **Implicit Adaptation:** Continuous via sliding window attention vs. reactive binary resets
- **Relational Feature Modelling:** Captures complex interactions better than axis-aligned trees
- **Drift Resilience:** Mitigates "drift chattering" in high noise
- **Efficiency:** Competitive accuracy with ensembles while significantly faster

B.3 Architecture Overview

Encoder-only Transformer design tailored for tabular streams:

- **Sliding Window Attention:** Fixed FIFO buffer ($W = 50$) for $O(1)$ memory with context
- **Causal Masking:** Triangular mask for temporal causality
- **Online Scaling:** Welford's Algorithm for $O(1)$ feature standardisation
- **Sinusoidal Encodings:** Fixed positional encodings without learnable parameters

B.4 Repository Structure

- `StreamTransformerCapyMOA_V4.ipynb`: Primary research notebook with pipeline/visualisations
- `StreamTransformer`: Core Python class
- `StreamTransformer_ADWIN`: Wrapper with ADWIN drift detection
- `OnlineStandardScaler`: Welford's algorithm implementation

B.5 Requirements & Installation

Python 3.8+ and NVIDIA GPU (Tesla T4 or better) required.

```
pip install capymoa torch numpy pandas matplotlib seaborn tqdm scipy
```

B.6 Usage Example

```
from stream_transformer import StreamTransformer
from capymoa.stream import Electricity
```

```
from capymoa.evaluation import prequential_evaluation

stream = Electricity()
learner = StreamTransformer(schema=stream.schema)

results = prequential_evaluation
(stream=stream, learner=learner, max_instances=100000)

print(f"Accuracy: {results.accuracy():.2f}%")
print(f"Kappa: {results.kappa():.4f}")
```

B.7 Experimental Results

Evaluated across 10 datasets, showing specialised dominance in rule-based and high-noise streams:

Table B.1: StreamTransformer Performance Highlights

Dataset	StreamTransformer Accuracy	Top Competitor (ARF/SRP)
Agrawal_a	**86.06%**	85.87%
Airlines	**69.60%**	68.92%
Electricity	**89.86%**	89.33%

Key Performance Insight

While tree ensembles dominate very high-dimensional spaces, StreamTransformer occupies optimal position on **accuracy–latency frontier** for mid-dimensional complex streams.

B.8 Repository Access

Available at:

`https://github.com/raycmarange/StreamTransformer`

B.9 Source Code Structure

B.9.1 OnlineStandardScaler

Implements Welford's online algorithm:

- `partial_fit(x)`: Updates running statistics
- `transform(x)`: Standardises features using current statistics

B.9.2 SinusoidalPositionalEncoding

Fixed sinusoidal encodings for temporal awareness without learnable parameters.

B.9.3 StreamTransformer

Core classifier with sliding window attention and causal masking:

- `train(instance)`: Updates scaler, buffers instance, triggers training
- `predict_proba(instance)`: Generates predictions via attention
- `reset()`: Clears buffers and statistics

B.9.4 StreamTransformer_ADWIN

Wrapper combining StreamTransformer with ADWIN drift detector.

B.10 Citation

If using this work, cite:

> Marange, Ray Chakanetsa (2026). *StreamTransformer: A Transformer-Based Model for Classification in Dynamic Data Streams*. Master's Thesis, Victoria University of Wellington.