END-TO-END EARLY CLASSIFICATION OF TIME SERIES IN NON-STATIONARY ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Early Classification of Time Series (ECTS) requires making accurate decisions as early as possible in inherently online and evolving environments. Yet, most existing methods assume stationarity and rely on separable designs, where classification and triggering are optimized independently, an assumption that fundamentally limits their adaptability under drift. In this work, we challenge this paradigm and study ECTS under non-stationary conditions. We provide the first controlled empirical comparison between separable and end-to-end approaches under multiple drift regimes. Building on Reinforcement Learning, we introduce DQeND, a unified architecture that jointly learns representation, classification, and triggering decisions while remaining directly comparable to state-of-the-art separable baselines. Across a wide range of drifts, DQeND demonstrates strong robustness across various non-stationary scenarios. Overall, our results indicate that end-to-end learning can offer improved adaptation capabilities for ECTS in dynamic environments and motivate further investigation of alternatives to separable designs.

1 INTRODUCTION

Early Classification of Time Series (ECTS) has emerged as a critical paradigm for applications where timely decision-making is as important as predictive accuracy (Gupta et al., 2020; Akasiadis et al., 2024; Renault et al., 2025). By allowing a model to issue a classification before observing the full time series, ECTS systems inherently operate under a trade-off between *earliness* and *accuracy*. While substantial progress has been made in designing accurate early classifiers, most existing approaches are developed and evaluated under the implicit assumption of a stationary data-generating process. However, this assumption rarely holds in practical deployments, where the process generating successive time series can be subject to distribution shifts and concept drifts. Despite the growing maturity of the ECTS literature, the impact of non-stationarity remains largely unexplored.

For instance, detecting toxic behavior on social media is subject to strong temporal constraints, as harmful content should be identified and filtered as early as possible. Moreover, the nature of such content can evolve with current events, e.g. during election periods, requiring early detection systems to continuously adapt to shifting definitions of what should be flagged.

One specificity of ECTS is that it can be viewed as a dual decision process: (*i*) the *classification* task, which aims to predict the class label based on partial observations, and (*ii*) the *trigger* (or stopping) task, which determines the optimal time at which a prediction should be emitted. The vast majority of existing methods adopt a separable design, where these two components are implemented independently and optimized one after another (Gupta et al., 2020; Renault et al., 2025). Such a decomposition offers practical advantages, including flexibility, interpretability, and ease of optimization. However, it also implicitly assumes that classification and stopping decisions can be optimized independently. This assumption becomes particularly limiting when the underlying time series distribution evolves over time. In such conditions, both the classification boundaries and the optimal stopping time may evolve in a coupled manner. A separable framework, by design, may struggle to capture this joint adaptation, as each component is typically optimized in a sequential fashion, e.g., encoder $\rightarrow$ classifier $\rightarrow$ trigger, preventing upstream modules from benefiting from feedback provided by downstream decisions. In contrast, end-to-end architectures alleviate these unidirectional optimization dependencies by enabling the learning signal, namely the incurred cost, to propagate

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

throughout the entire decision pipeline. Yet, this potential benefit of end-to-end learning for ECTS under distribution drift has yet to be investigated. Indeed, on one hand, existing work on end-to-end ECTS is limited in scope: they are often evaluated under stationary conditions and rarely benchmarked against strong separable baselines. On the other hand, existing work on non-stationarity remains within the separable scope and mainly focuses on adapting the triggering mechanism under changing decision costs while assuming a fixed classification model (Renault et al., 2026b). Consequently, a question stands out: does jointly optimizing classification and stopping decisions improve robustness and adaptability when the underlying time series distribution itself evolves? And can we identify where the differences with separable methods, if any, manifest themselves?

In this work, we provide a systematic investigation of separable and end-to-end ECTS architectures under controlled non-stationarities. Our contributions are threefold:

- We conduct an extensive empirical comparison between separable and end-to-end paradigms, analyzing the regimes in which joint optimization becomes beneficial.
- Building on recent work on Reinforcement Learning (RL)-based ECTS (Martinez et al., 2020; Renault et al., 2026a), we propose an end-to-end framework that jointly optimizes classification and triggering decisions, while being directly comparable to state-of-the-art separable baselines.
- We design controlled experimental protocols involving covariate shifts and concept drifts, demonstrating that end-to-end optimization improves robustness and adaptation capabilities across a wide range of evolving environments.

2 BACKGROUND

2.1 PROBLEM STATEMENT

In the ECTS problem, an input time series of size T is observed progressively. At time $t \leq T$, only a prefix $\mathbf{x}_t = \langle x_1, \ldots, x_t \rangle$ is available, where $\{x_i\}_{(1 \leq i \leq t)}$ denotes the possibly multivariate measurements. The full series $\mathbf{x}_T$ belongs to an unknown class $y \in \mathcal{Y}$. The task is to output a prediction $\hat{y} \in \mathcal{Y}$ for the class of the incoming time series at some time $\hat{t} \in [1, T]$. E The prediction correctness is measured by a misclassification cost $\mathrm{C}_m(\hat{y}|y)$, and time pressure is modeled by a delay cost $\mathrm{C}_d(t)$, which is assumed to be positive and typically non-decreasing over time. Formally, we consider: $\mathrm{C}_m : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}, \quad \mathrm{C}_d : \mathbb{R}^+ \to \mathbb{R}$.

Let $s \in \mathcal{S}$ be some ECTS function belonging to a class of functions $\mathcal{S}$, whose output at time t when receiving $\mathbf{x}_t$ is:

$$s(\mathbf{x}_t) = \begin{cases} \emptyset & \text{if extra measures are queried;} \\ (\hat{y}_t,\ t) & \text{if prediction is triggered at } t < T; \end{cases} \tag{1}$$

Specifically, when the deadline is reached, i.e. $t = T$, the model is forced to return a prediction: $s(\mathbf{x}_T) = (\hat{y}_T, T)$. In practice, ECTS is an online optimization problem: at each time t, the function $s(\mathbf{x}_t)$ must decide whether to predict or wait for more data, based only on the prefix $\mathbf{x}_t$. The s function induces a stopping time $\hat{t}$ on each incoming series that is the first time step for which s returns a non-null prediction. It can be defined as:

$$\hat{t} = \inf\{t \mid s(x_t) \neq \emptyset\} \tag{2}$$

Once triggered, the remaining observations $\{\mathbf{x}_t\}_{(\hat{t} < t \leq T)}$ are not considered and prediction process is stopped. From a machine learning perspective, the goal is to find a function $s \in \mathcal{S}$ that best optimizes the time-dependent loss function $\mathcal{L}(s(\mathbf{x}_{\hat{t}}), y) = \mathrm{C}_m(\hat{y}_{\hat{t}}|y) + \mathrm{C}_d(\hat{t})$, minimizing the true risk over the data-generating process $\mathbb{P}(\mathcal{X} \times \mathcal{Y})$:

$$\arg\min_{s \in \mathcal{S}} \mathbb{E}_{(\mathbf{x}_T, y) \sim \mathbb{P}(\mathcal{X} \times \mathcal{Y})} \left[\mathcal{L}(s(\mathbf{x}_{\hat{t}}), y) \right] \tag{3}$$

In order to solve the ECTS problem, a training set is made available that is composed of N labeled time series, denoted by $(\mathbf{x}_T^i, y^i)_{i \in [0,N]} \in (\mathcal{X} \times \mathcal{Y})$, where each series $\mathbf{x}_T = \langle x_1, \ldots, x_T \rangle$ is complete and of the same size T, and associated with its label $y \in \mathcal{Y}$.

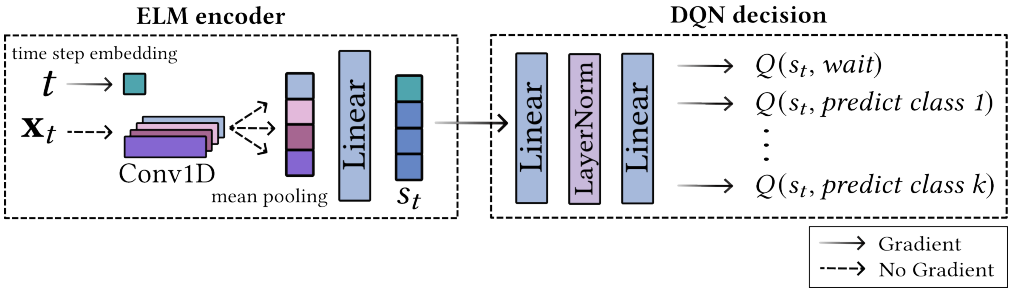


Figure 1: DQeND pipeline

2.2 Online ECTS with drift

The standard ECTS formulation typically assumes that the data-generating process observed at training time remains unchanged at deployment. This implicitly entails that both the input distribution and the underlying predictive relationship are stationary. In practice, this assumption is often violated: the *actual* conditions encountered in production may differ from those seen during training and can evolve over time. Such variations may affect the input distribution (covariate shift), or the conditional relationship between inputs and labels (concept drift). Those may occur either within the duration of a single time series (i.e., along the time index $t \in [0, T]$) or across longer time scales, for instance due to seasonal or operational changes.

To explicitly account for these different temporal dimensions, we distinguish between the time index of a given time series and the deployment timeline. Let $u \in \mathbb{N}^+$ index successive incoming labeled time series $\{(\mathbf{x}_T, y)^u\}_{(1 \leq u \leq U)}$ observed during deployment, $(\mathbf{x}_T, y)^u$ is generated by some distribution $\mathcal{D}_u$ over $\mathcal{X} \times \mathcal{Y}$. At deployment, an ECTS system repeatedly processes incoming time series drawn from the evolving sequence of distributions $\{\mathcal{D}_u\}_{u \geq 1}$. The deployment performance of some ECTS function $s \in \mathcal{S}$ is then measured by:

$$R_U(s) = \sum_{u=1}^{U} \mathbb{E}_{(\mathbf{x}_T, y) \sim \mathcal{D}_u} \left[\mathcal{L}(s(\mathbf{x}_{\hat{t}}), y) \right], \tag{4}$$

where the expectation is taken with respect to the current (drifting) data-generating process, as well as any additional sources of randomness.

3 DQeND: End-to-end RL for ECTS

In this section, the design choices for the proposed end-to-end ECTS pipeline, called DQeND, are described. In contrast to traditional separable approaches, where the different components are optimized independently, our architecture integrates them into a single decision process that can be trained and adapted holistically. The pipeline consists of two main modules: (*i*) an *encoder* that transforms raw time series prefixes into informative latent representations, and (*ii*) a RL–based *decision* module that simultaneously determines when to stop and which class to predict.

3.1 Architecture

To enable fair comparisons with state-of-the-art separable methods (see Appendix A) and to isolate the impact of end-to-end learning from architectural differences, we build our pipeline upon both ROCKET-type and Alert frameworks (Dempster et al., 2019; 2021; Renault et al., 2026a).

3.1.1 ELM Encoder

The design of the encoder module is guided by the objective of combining strong representational capacity with computational efficiency in an online setting. To this end, we draw inspiration from the principles underlying the ROCKET family of methods (Dempster et al., 2019; 2021), whose features are computed via a large number of random convolutions. More broadly, the idea of generating a

large hidden representation of the input data using random weights originates from the Extreme Learning Machine (ELM) literature (Huang et al., 2006; Wang et al., 2022).

At a high level, ELM-based approaches are single-layer neural networks. They rely on projecting input data into a high-dimensional feature space using a large set of randomly initialized and fixed parameters, and subsequently learning a lightweight mapping on top of this representation. Formally, the output of some ELM network with L hidden nodes can be defined as $f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i h_i(\mathbf{x})$, where $\beta = [\beta_1, \cdots, \beta_L]$ is the learned weight matrix, and $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \cdots, h_L(\mathbf{x})]$ the fixed random features for the i-th hidden node. This paradigm has been shown to yield competitive performance across a wide range of time series classification tasks (Liu et al., 2016; Park & Kim, 2017), while maintaining a favorable computational profile due to the limited number of trainable parameters. Such properties are particularly desirable in online learning scenarios, where frequent updates must be performed under efficiency constraints.

Combining the ELM principles with the ROCKET-type of time series transformations, the proposed encoder implements a randomized feature extractor based on a single one-dimensional convolutional layer with frozen weights. Multiple kernel sizes k are employed in parallel in order to capture temporal patterns at different scales. The convolutional outputs are then aggregated through a mean pooling operation, producing a fixed-size embedding regardless of the input length. In particular, feature h_i can be obtained as: $h_i(\mathbf{x}_t) = \text{AvgPool}(\text{Conv1D}_k(\mathbf{x}_t))$.

The convolutional filters are initialized by sampling their weights from a standard normal distribution and remain unchanged throughout training and deployment. The trainable representation parameters consist of the temporal embedding and the projection layer applied to the random convolutional features. To complement this representation, we additionally learn an explicit embedding of the time step, which is concatenated to the ELM-based features. This allows the downstream decision modules to incorporate temporal position information alongside the extracted signal patterns, a key aspect for early classification tasks where the decision process is inherently time-dependent.

3.1.2 DQN DECISION

The decision-making module is designed to remain directly comparable to strong separable baselines while enabling full end-to-end optimization. To this end, we adopt the same underlying architecture as the Alert framework (Renault et al., 2026a), namely a Deep Q-Network (DQN)-based decision model (Dayan & Watkins, 1992; Sutton & Barto, 2018). This choice ensures that any observed differences in performance can be attributed to the representation and training paradigm, rather than to architectural discrepancies in the decision process itself.

In contrast to separable approaches, Alert in particular, that typically rely on handcrafted features or classifier-derived confidence scores, the proposed module operates directly on the learned representation produced by the encoder. More precisely, the input state s_t is given by the embedding output at time t, allowing the decision module to leverage rich, task-adaptive features without additional engineering.

The action space is extended so as to jointly model the stopping decision and the class prediction. Specifically, we define $\mathcal{A} = \{\text{“}\textit{wait}\text{”}, \text{“}\textit{predict class 1}\text{”}, \cdots, \text{“}\textit{predict class } K\text{”}\}$, thereby unifying the triggering and classification tasks within a single decision process. At each time step, the selected action follows the standard greedy policy induced by the Q-function $a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a)$.

The prediction mechanism is derived directly from the chosen action. If the model selects a prediction action, the corresponding class is immediately output. A specific handling is introduced for the terminal time step T: if the selected action remains “*wait*”, the model is forced to emit a prediction, corresponding to the class associated with the highest Q-value among the non-waiting actions. Thus, the predicted label is defined as:

$$\hat{y}_t(a_t) = \begin{cases} \emptyset & \text{if } a_t = \text{“}\textit{wait}\text{” and } t < T \\ \underset{a \in \mathcal{A} \setminus \{\text{“}\textit{wait}\text{”}\}}{\arg\max} \; Q(s_t, a) & \text{if } a_t = \text{“}\textit{wait}\text{” and } t = T \\ a_t & \text{if } a_t \neq \text{“}\textit{wait}\text{”} \end{cases} \tag{5}$$

3.2 TRAINING (OFFLINE)

For stability reasons, learning is performed offline in a batch setting, where the model is trained on a fixed collection of fully labeled time series without requiring active exploration during training. In particular, the overall training procedure is kept close to that of Alert (Renault et al., 2026a). At each time step, the reward is expressed as the temporal variation of the delay cost when the agent decides to wait, and additionally incorporates the misclassification cost when a prediction is made. Formally, the reward is defined as:

$$r(t) = \begin{cases} -\Delta \mathrm{C}_d(t) & \text{if } a_t = \text{“}\textit{wait}\text{”} \\ -C_m(\hat{y}_t|y) - \Delta \mathrm{C}_d(t) & \text{if } a_t \neq \text{“}\textit{wait}\text{”} \end{cases} \tag{6}$$

where $-\Delta \mathrm{C}_d(t) = \mathrm{C}_d(t) - \mathrm{C}_d(t-1)$, with $\mathrm{C}_d(0) = 0$.
The gradient of the DQN objective, corresponding to the Bellman error, is propagated through the entire DQeND pipeline, updating all parameters associated with the shaded arrows in Figure 1. Finally, the model selection procedure is kept identical to that of Renault et al. (2026a).

3.3 DEPLOYMENT (ONLINE)

At deployment time, each observed interaction is stored in a replay buffer. The pipeline is then updated by sampling mini-batches of transitions from this buffer, following standard DQN optimization procedures. When moving from offline training to online adaptation, exploration is explicitly reintroduced to account for potential distribution shifts. Concretely, an ϵ-greedy strategy is adopted, where with probability ϵ a random action is selected, and with probability $1-\epsilon$ the greedy action, i.e. the one with maximum Q-value, is chosen. In the exploration phase, actions are sampled uniformly from the action space, ensuring equal probability across all possible decisions (e.g., $1/3$ for each action in the binary classification case). In all experiments, the exploration rate is fixed at $\epsilon = 5\%$.

4 EXPERIMENTAL PROTOCOL

This section describes the experimental protocol used to evaluate the proposed online ECTS methods under non-stationarities. We detail the evaluation metrics, the drift scenarios, and the evaluated methods. The data generation process is described in Appendix B.3.

4.1 EVALUATION METRICS

All online methods are evaluated using an *eval-then-update* (prequential) protocol (Gama et al., 2009). At each deployment step u, a time series and its associated deployment-time costs are first used to evaluate the current model. The observed loss is then used to update the models according to its update strategy. To study the impact of different trade-offs between misclassification and delay costs, we use the standard weighted formulation of the average cost, along with the cost functions definitions used by Renault et al. (2025):

$$\textit{AvgCost}_\alpha = \frac{1}{N}\sum_{i=1}^{N} \mathrm{C}_m(\hat{y}_i|y_i) + \mathrm{C}_d(\hat{t}_i) = \frac{1}{N}\sum_{i=1}^{N} \alpha\, \mathbb{1}(\hat{y}_i \neq y_i) + (1-\alpha)\,\frac{\hat{t}_i}{T}$$

where $\alpha \in [0,1]$ controls the relative importance of accuracy and earliness. Throughout this work, as done by Mori et al. (2017); Lv et al. (2023), we fix $\alpha = 0.8$.

The evaluation relies on a set of three metrics derived from the standard *AvgCost*:

First, we report the ***cumulative cost***, defined as the total sum of incurred costs over all deployment examples.

Second, we consider the ***hold-out AvgCost drift***, computed on a disjoint evaluation set that is subject to the same drift conditions as the current deployment step. Unlike the raw cumulative deployment cost, which may exhibit high variance due to transient effects or small batches, this metric provides a more stable estimate of performance under the current time series distribution.

Finally, we report the ***hold-out AvgCost original***, evaluated on a separate hold-out set drawn from the original (training-time) distribution. This complementary metric assesses the extent to which

a method preserves performance on the base concept, thereby offering insight into its ability to balance adaptation and retention; often referred to as the stability–plasticity trade-off (Kim & Han, 2023; Serra-Perello & Ortiz, 2025).

4.2 Drift Scenarios

We consider controlled variations of the input distribution and class structure. Non-stationarity occurs across deployment steps u, whereas $t \in [1, T]$ denotes progression within an individual time series. We study incremental drifts, where a variable changes smoothly during deployment, and abrupt drifts, where it switches discontinuously to a new regime. The resulting scenarios are summarized in Table 1.

Table 1: Controlled non-stationary deployment scenarios.

Scenario	*Drift type*	*Modified variable*	*Deployment schedule*
I_loc	Incremental	Template location	Pattern shifts smoothly from the beginning to the end of the series.
I_noise	Incremental	Gaussian noise	Noise scale increases log-linearly from 0.01 to 1.
A_loc	Abrupt	Template location	Pattern moves from the first to the last quarter after five steps.
A_class	Abrupt / recurrent	Meta-class composition	Sub-classes are resampled with probability 5% at each step.

4.3 Evaluated Methods

Separable baselines As highlighted in Section 3, Alert constitutes the comparison reference for the proposed end-to-end method. Thus, all the separable models described in this section are based on: (*i*) a *classifier* that shares the same architecture as the ELM encoder described in Section 3.1.1, followed by a linear classification layer, and (*ii*) a *trigger* which is exactly Alert (Renault et al., 2026a), including the input space. In what follows, several ways of adapting separable baselines to online environments, depending on the tested drift type, are presented.

- ***Common baselines***: (*i*) *bronze*: both the classifier and the trigger model remain frozen during deployment. (*ii*) *silver*: both the classifier and the trigger model are trained on the concept on which they will be evaluated. Act as an upper bound, not realistic in practice.
- ***Incremental drifts***: (*i*) *online/freeze*: the classifier is updated online, after each deployment step, the trigger model remains frozen. (*ii*) *online/online*: both the classifier and the trigger model are updated online after each deployment step. The data batch is split into two so that each module is updated based on one half of the batch.
- ***Abrupt drifts***: (*i*) *retrain/freeze*: the classifier is retrained from scratch at a given frequency; the trigger model remains frozen. (*ii*) *retrain/retrain*: both the classifier and the trigger model are retrained from scratch. The data batch is split into two as above.

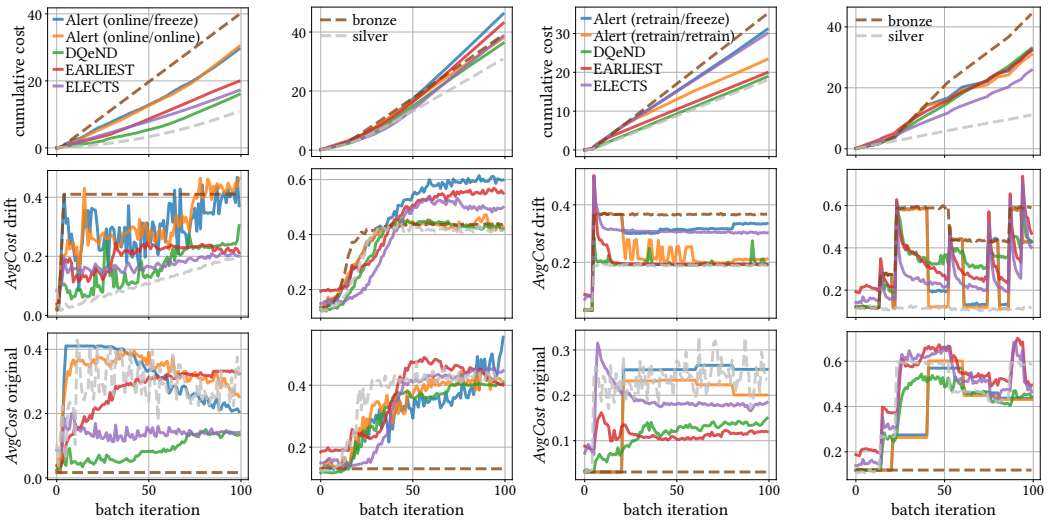
End-to-end The following end-to-end methods have been adapted so that they may be updated online: ELECTS (Rußwurm et al., 2023) and EARLIEST (Hartvigsen et al., 2019). More implementation details, including feedback availability, are discussed in Appendix B.

5 Results

5.1 Incremental drifts

Incremental Location (I_*loc*) This first experiment investigates a gradual shift of discriminative patterns toward later positions in the deployment time series. As shown in Figure 2a, end-to-end approaches generally outperform separable baselines, consistently achieving the lowest cumulative costs (top panel). In particular, DQeND ranks first overall and, together with ELECTS, best preserves performance on the original concept (bottom panel). In addition, the qualitative analysis in Figure 3 (top panel) shows that end-to-end models effectively adapt their triggering policies by progressively delaying predictions, illustrated by the increasing light-blue earliness curve of each box, as the discriminative patterns shift later in the series. In contrast, both Alert-based separable methods degrade significantly as informative patterns move later in the sequences (middle panel, right),

highlighting the limitations of decoupled classification and triggering under temporal drift. Freezing the Alert trigger further leads to unstable behavior throughout deployment, whereas updating both classifier and trigger slightly improves stability but still fails to improve global performance.



(a) ***I_loc*** (b) ***I_noise*** (c) ***A_loc*** (d) ***A_class***

Figure 2: Incremental drifts, deployment's metrics, averaged over five seeds. From top to bottom: the raw cumulative cost; the hold-out *AvgCost*, obtained on unseen drifted data; the hold-out *AvgCost* original, obtained using the fixed test set from training phase. Uppercase approaches refer to end-to-end methods while lowercase indicates separable ones.

Incremental Noise (I_*noise*) This second experiment evaluates robustness under a gradual increase of additive Gaussian noise, progressively degrading the signal-to-noise ratio and the reliability of discriminative patterns. As displayed in Figure 2b, end-to-end approaches again achieve the best cumulative costs, with DQeND ranking first, closely followed by ELECTS. Despite similar overall performance, analysing Figure 3, bottom panel, indicates that the two methods exhibit distinct adaptation behaviors. ELECTS follows a conservative strategy, initially delaying predictions to preserve accuracy while the signal remains informative, before progressively shifting toward earlier decisions as noise becomes dominant. In contrast, DQeND rapidly converges toward an early-prediction regime once accuracy approaches random performance levels , effectively minimizing delay costs when waiting no longer provides useful information. Regarding separable approaches, the same trends as in the incremental location drift experiment can be observed. In particular, the frozen Alert trigger again produces unstable behavior, whereas updating both classifier and trigger significantly improves robustness. The *online/online* variant achieves performance close to DQeND under low to moderate noise levels (Figure 2b, middle panel, left), although the end-to-end model remains slightly more robust, suggesting that unified architectures better exploit weak residual signal under mild perturbations.

5.2 Abrupt Drifts

Abrupt Location (*A_loc*) This experiment evaluates an abrupt drift scenario in which the discriminative pattern is suddenly shifted from the beginning to the end of the time series. As shown in Figure 2c, middle panel, the RL-based approaches adapt most effectively to this sudden change. Both DQeND and EARLIEST rapidly shift from early to late predictions, successfully adjusting their triggering policies to the new pattern location. RL-based methods also best preserve performance on the original concept (Figure 2c, bottom).In contrast, ELECTS struggles to substantially delay its decisions at deployment, with trigger points rarely extending beyond half of the sequence. This limited shift highlights a lack of plasticity, as the model remains biased toward its initial regime

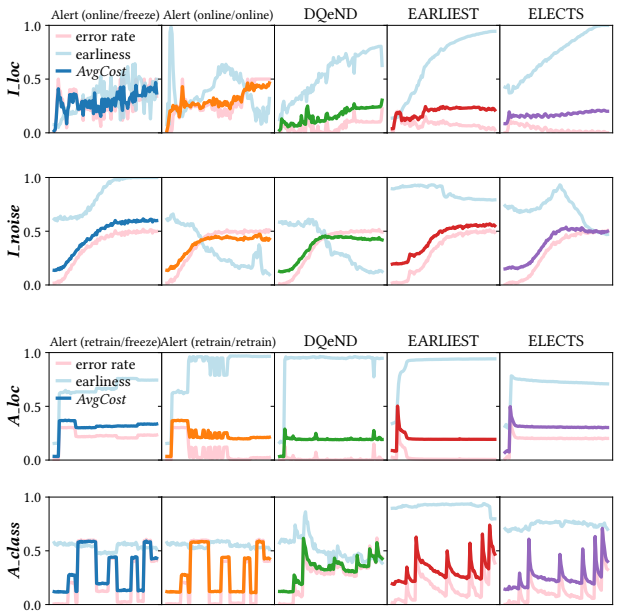


Figure 3: Hold-out *AvgCost* drift: detailed performances. The figures display the decomposition of the hold-out *AvgCost* drift into both earliness, i.e. the average proportion of the time series used, and error rate, during the deployment phase. Uppercase approaches refer to end-to-end methods while lowercase indicates separable ones.

and fails to fully adapt to the new optimal decision timing. For the Alert-based separable baselines, the importance of online adaptation becomes even more evident. Freezing the trigger model can lead to catastrophic failures, as the decision policy becomes completely misaligned with the shifted distribution. Updating the trigger improves stability and eventually recovers appropriate triggering behavior, although adaptation remains slower than with end-to-end approaches.

Sub-classes shuffling (*A_class*) This final experiment considers a highly non-stationary setting where the composition of meta-classes is randomly reshuffled at irregular intervals, directly altering the underlying class structure rather than the time series distribution. In Figure 2d, ELECTS achieves the best overall performance, consistently recovering after each drift event. The RL-based approaches, DQeND and EARLIEST, closely follow and remain competitive. Noticeably, RL-based approaches suffer from a stronger constraint: unlike ELECTS and separable baselines, they only observe the data up to the estimated trigger time rather than the full time series. This makes adaptation to abrupt-type drifts substantially more challenging, a strong exploration rate being thus critical to efficiently recover. A more detailed analysis via Figure 3, bottom panel, nevertheless reveals a limitation of DQeND in this regime. Under repeated abrupt drifts, its triggering policy progressively collapses toward systematically early predictions. This highlights the instability of the fully RL-based pipeline, which likely results from the accumulation of policy updates across successive distribution shifts. In contrast, ELECTS and EARLIEST display more stable earliness profiles (almost flat earliness curve) throughout deployment, reflecting a more conservative adaptation strategy. Within this setting, separable baselines achieve performance comparable to the RL-based approaches, although they are retrained at fixed, predetermined intervals.

5.3 ARXIV DATASET

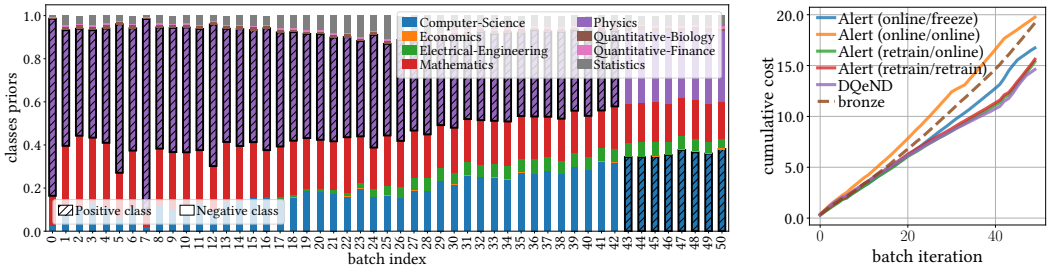
We further apply our experimental protocol to a real-world temporally ordered corpus exhibiting natural prior and covariate shifts, on which we construct an additional changing-semantics binary classification task. We use the arXiv dataset (Clement et al., 2019), which consists of categorizing arXiv papers based on their abstracts. For ECTS, the goal is to classify the paper category using as few sentences as possible. The abstracts are turned into multivariate time series of length 20, using *word2vec* (Mikolov et al., 2013) over successive text chunks. We first gather the 175 possible categories into six main classes and then turn this into a binary classification task, defining the positive class as the majority category and the negative class as all others. In particular, several types of drift appear simultaneously in the data:

- *Classes priors drift* (Abrupt & Incremental): As displayed in Figure 4a, the class balance evolves considerably. Indeed, during training-time, the positive class, i.e., Physics papers, represents nearly 80% of the dataset. It progressively shrinks and reaches only around 30% at the end of deployment.
- *Covariate shift* (Incremental): To check for the presence of covariate shift, a simple classifier is trained to discriminate the examples from the same class at different deployment stages. Table 2 shows that the classifier performances for the three most populated classes of the dataset are generally better than random, effectively revealing some distributional changes within the data.

Table 2: Test accuracy of a linear Ridge classifier.

Class	*Train vs. Depl. start*	*Depl. start vs half*	*Depl. start vs. end*
Physics	84.98%	55.44%	89.49%
Mathematics	78.45%	56.41%	59.62%
Computer Science	80.61%	63.59%	69.36%

- *Concept drift* (Abrupt): Some form of meta-class shuffling is also introduced by defining our task as the majority class vs. the rest. Indeed, at some point in deployment, the majority class switches from Physics to Computer Science, and the positive class thus changes semantics. To quantitatively check for concept drift, a linear Ridge classifier is trained over batch no. 42 and then evaluated over batch no. 43. While test accuracy, taken over batch no. 42, reaches 88%, it drops to 36% when moving to batch no. 43; thus demonstrating a strong concept drift.



(a) Classes priors evolution over deployment.

(b) Cumulative cost.

Figure 4: Arxiv experiment details.

The arXiv experiment provides an external validation of the relative robustness of DQeND against the evaluated Alert-based separable variants. As displayed in Figure 4b, DQeND is again the most robust approach in this setting, closely followed by Alert-based methods that perform full retraining of the classifier periodically. Updating both the classifier and the trigger online leads to highly unstable behavior, resulting in a higher cost than the non-adaptive baseline; emphasizing the technical challenge of updating two interdependent models online.

6 Conclusion

In this paper, we investigated the relevance of end-to-end learning for Early Classification of Time Series under non-stationarity. To address this question, we conducted extensive experiments within a controlled synthetic drift framework, comparing end-to-end approaches against the separable paradigm. Across a broad range of gradual and abrupt non-stationarities, end-to-end approaches generally demonstrate stronger robustness and adaptation capabilities than separable competitors, whose behavior can often be unstable and limited under drift. Future work could focus on feedback availability, delayed rewards, and settings where the time series may not be observed after the triggering time.

Reproducibility Statement

Experiments in this paper are fully reproducible, and the source code can be found at `https://anonymous.4open.science/r/ects_concept_drift-8177`

References

Youssef Achenchabe, Alexis Bondu, Antoine Cornuéjols, and Asma Dachraoui. Early classification of time series: Cost-based optimization criterion and algorithms. *Machine Learning*, 110(6): 1481–1504, 2021.

Charilaos Akasiadis, Evgenios Kladis, Petro-Foti Kamberi, Evangelos Michelioudakis, Elias Alevizos, and Alexander Artikis. A framework to evaluate early time-series classification algorithms. In *EDBT*, pp. 623–635, 2024.

Jakub Michal Bilski and Agnieszka Jastrzebska. Calimera: A new early time series classification method. *Information Processing & Management*, 60(5):103465, 2023.

Colin B Clement, Matthew Bierbaum, Kevin P O'Keeffe, and Alexander A Alemi. On the use of arxiv as a dataset. *arXiv preprint arXiv:1905.00075*, 2019.

Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols. Early classification of time series as a non myopic sequential decision making problem. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pp. 433–447. Springer, 2015.

Peter Dayan and CJCH Watkins. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *arXiv preprint arXiv:1910.13051*, 2019.

Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 248–257, 2021.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 329–338. Association for Computing Machinery, 2009. ISBN 9781605584959.

Sam Greydanus and Dmitry Kobak. Scaling down deep learning with mnist-1d. *arXiv preprint arXiv:2011.14439*, 2020.

Ashish Gupta, Hari Prabhat Gupta, Bhaskar Biswas, and Tanima Dutta. Approaches and applications of early classification of time series: A review. *IEEE Transactions on Artificial Intelligence*, 1(1): 47–61, 2020.

Thomas Hartvigsen, Cansu Sen, Xiangnan Kong, and Elke Rundensteiner. Adaptive-halting policy network for early classification. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 101–110, 2019.

Thomas Hartvigsen, Cansu Sen, Xiangnan Kong, and Elke Rundensteiner. Recurrent halting chain for early multi-label classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1382–1392, 2020.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

Dongwan Kim and Bohyung Han. On the stability-plasticity dilemma of class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20196–20204, 2023.

Huaping Liu, Lianzhi Yu, Wen Wang, and Fuchun Sun. Extreme learning machine for time sequence classification. *Neurocomputing*, 174:322–330, 2016.

Junwei Lv, Yuqi Chu, Jun Hu, Peipei Li, and Xuegang Hu. Second-order confidence network for early classification of time series. *ACM Transactions on Intelligent Systems and Technology*, 2023.

Coralie Martinez, Guillaume Perrin, Emmanuel Ramasso, and Michèle Rombaut. A deep reinforcement learning approach for early classification of time series. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 2030–2034. IEEE, 2018.

Coralie Martinez, Emmanuel Ramasso, Guillaume Perrin, and Michèle Rombaut. Adaptive early classification of temporal sequences using deep reinforcement learning. *Knowledge-Based Systems*, 190:105290, 2020.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Usue Mori, Alexander Mendiburu, Eamonn Keogh, and Jose A Lozano. Reliable early classification of time series based on discriminating the classes over time. *Data mining and knowledge discovery*, 31:233–263, 2017.

Jin-Man Park and Jong-Hwan Kim. Online recurrent extreme learning machine and its application to time-series prediction. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1983–1990. IEEE, 2017.

Aurélien Renault, Alexis Bondu, Antoine Cornuéjols, and Vincent Lemaire. Early classification of time series: A survey and benchmark. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=bcNDYmBicK. Reproducibility Certification.

Aurélien Renault, Alexis Bondu, Antoine Cornuéjols, and Vincent Lemaire. Alert: Learning trigger functions for early classification of time series using deep-RL. *Transactions on Machine Learning Research*, 2026a. ISSN 2835-8856. URL https://openreview.net/forum?id=J7EIFs4sjx.

Aurélien Renault, Alexis Bondu, Antoine Cornuéjols, and Vincent Lemaire. Early classification of time series in non-stationary cost regimes. *arXiv preprint arXiv:2602.00918*, 2026b.

Marc Rußwurm, Nicolas Courty, Rémi Emonet, Sébastien Lefèvre, Devis Tuia, and Romain Tavenard. End-to-end learned early classification of time series for in-season crop type mapping. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:445–456, 2023.

Miquel Serra-Perello and Alberto Ortiz. Incremental learning methodologies for addressing catastrophic forgetting: Analysis and experimental evaluation. *Journal of Artificial Intelligence Research*, 83, 2025.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Jian Wang, Siyuan Lu, Shui-Hua Wang, and Yu-Dong Zhang. A review on extreme learning machine. *Multimedia Tools and Applications*, 81(29):41611–41660, 2022.

A Related works

The ECTS literature can be broadly divided into *end-to-end* and *separable* approaches. End-to-end methods jointly learn a classifier and a triggering mechanism. While flexible, they are classifier-specific and often difficult to compare consistently across studies. In contrast, *separable* approaches decouple classification and decision triggering. This modular design dominates the literature and has been extensively evaluated (Gupta et al., 2020; Akasiadis et al., 2024; Renault et al., 2025).

A.1 Separable ECTS

Classification module A common design choice in separable ECTS systems consists of discretizing the temporal axis and learning a collection of classifiers, each specialized for a given timestamp or subset of timestamps. This strategy enables the direct reuse of state-of-the-art Time Series Classification algorithms, while ensuring that predictions can be produced at different stages of the sequence. Among recent advances, the MiniROCKET pipeline Dempster et al. (2021) has emerged as a particularly strong choice, combining high predictive performance with excellent scalability. Its effectiveness stems from the use of a large number of fixed random convolutional kernels, coupled with simple yet informative aggregation functions, yielding rich representations at a very low computational cost. As such, MiniROCKET-based classifiers are frequently adopted as backbone models in modern separable ECTS pipelines (Bilski & Jastrzebska, 2023; Renault et al., 2026a;b).

Trigger module On the decision side, a widely used baseline is the Proba Threshold strategy, which triggers a prediction as soon as the maximum posterior class probability exceeds a predefined threshold. Despite its simplicity, this approach has proven to be competitive across a variety of datasets and cost configurations, and is therefore often used as a reference method (Renault et al., 2025).

Beyond such myopic strategies, most high-performing trigger models fall into the class of *non-myopic* approaches, which explicitly account for the potential evolution of future observations when making decisions. The Economy family of methods (Achenchabe et al., 2021; Dachraoui et al., 2015) follows this principle by triggering a decision when the expected cost at the current time step is minimal among all anticipated future costs. Similarly, Calimera (Bilski & Jastrzebska, 2023) adopts an anticipation-based framework, leveraging regression models learned via backward induction to estimate future costs and guide the triggering decision.

Recently, RL has been introduced as a flexible alternative for modeling the triggering process. In particular, Alert (Renault et al., 2026a) learns a triggering policy based on a state space composed of handcrafted features obtained by probing the ensemble of calibrated MiniROCKET estimators. In addition to strong empirical performance, this approach is inherently well-suited to online adaptation and can naturally incorporate feedback from a deployment phase. For those reasons, the present work primarily builds upon this trigger model and investigates its integration within an end-to-end architecture, where the classifier is directly informed by the triggering signal.

A.2 End-to-end ECTS

Recent advances in deep learning have enabled tighter integration between representation learning, classification, and decision timing. In particular, ELECTS (Rußwurm et al., 2023) proposes an LSTM-based architecture, augmented by a stopping head that determines the probability of triggering. It is trained with a dedicated loss function that explicitly balances classification accuracy and earliness.

In parallel, the growing popularity of RL has led to a number of approaches that formulate ECTS as a sequential decision-making problem. Earlier work by Martinez et al. (2018; 2020) introduced the use of Deep Q-Networks (DQN) for ECTS, modeling the triggering task as a value-based decision process. EARLIEST (Hartvigsen et al., 2019; 2020) relies on a policy gradient method to learn a stopping policy jointly with a LSTM-based encoder and classifier. These RL-based methods offer a natural framework to capture the trade-off between earliness and accuracy, while enabling flexible adaptation to complex and potentially non-stationary environments.

B IMPLEMENTATION DETAILS

B.1 ONLINE FEEDBACK PROTOCOL

At deployment, each method first receives only the prefix observed up to its triggering time. After a prediction is emitted, the ground-truth label is revealed. Unless otherwise stated, separable and ELECTS methods may use the complete labeled sequence for their prescribed update procedures, whereas RL-based methods update from transitions collected along the observed prefix. This reflects the native learning mechanisms of the compared methods, and further investigation about the impact of the feedback availability is left for future work.

B.2 HYPERPARAMETERS

The deployment phase consists of $U = 100$ steps, with batches containing 500 examples each. For the *retrain* baselines, the retraining frequency has been set to 20. For both ELECTS and EARLIEST algorithms, the training phase is conducted over 500 epochs; the deployment batch size has been set to 64, and the learning rate to 5e-3. All experiments have been run on CPU, with up to five threads to perform cross-validation when needed. Results that follow consist of the mean taken over five different random seeds. For all other details, we make the code available at: `https://anonymous.4open.science/r/ects_concept_drift-8177`. Details on all DQeND hyperparameters' values can be found in Table 3.

Table 3: DQeND hyperparameters

Encoder	kernel sizes	$\{3, 6, 9, 12\}$
	nb. of kernels	$\{128, 64, 32, 16\}$
	embedding dim.	64
	time embedding dim.	2
Decision	nb. hidden layers	1
	hidden dim.	64
	activation function	tanh
Training	learning rate	1e-3
	nb. epochs	100
	batch size	256
Deployment	learning rate	1e-4
	exploration rate ϵ	0.05
	batch size	64

B.3 DATA GENERATION

Studying non-stationarity in ECTS requires benchmarks where the difficulty of the task and the nature of the drift can be precisely controlled. In particular, the ability to independently manipulate factors such as noise intensity, temporal location of discriminative patterns, or class structure is essential. The MNIST-1D (Greydanus & Kobak, 2020) dataset naturally fulfills these requirements through its synthetic generation process, making it particularly well-suited for the controlled drift scenarios considered in this work. The MNIST-1D dataset is a ten-class classification problem, which has been binarized in this work, based on univariate time series of fixed length $T = 100$. Each generated time series contains a class-specific pattern representing the shape of a digit, embedded at a random position. The training phase is conducted using 1,000 examples (700 for training and the 300 left to compute the hold-out *AvgCost* original); at each deployment step u, 1,500 new (drifted) examples are generated, of which 500 are used for updates and 1,000 to compute the hold-out *AvgCost* drift.

B.4 ARXIV EXPERIMENT

For the arxiv experiment, the number of training examples is set to 10,000; the number of deployment steps is set to $U = 50$ with batches of 500 examples each. The *retrain* baselines retraining frequency has been set to 10. ELECTS and EARLIEST are not included in this experiment, as they

would need further adaptation to handle multivariate time series. All other hyperparameters are the same as in the synthetic experiment.

C Ablation studies

C.1 Freezing subparts of DQeND

To further assess the benefits of the proposed end-to-end design, we conduct an ablation study in which the encoder and decision modules of DQeND are alternatively frozen during adaptation. This allows for isolating the respective contributions of representation learning and decision-making in the presented non-stationary settings.

The results, reported in Table 4, show that jointly updating both modules consistently yields the best performance across most scenarios. In contrast, partially frozen variants lead to noticeable degradations, confirming the importance of coordinated adaptation. Among the two, freezing the decision module results in the most significant performance drop. This observation is further supported by the fact that the decision-making module accounts for most of the learnable parameters in the proposed architecture.

Interestingly, freezing the encoder has a more limited impact in the abrupt drift settings considered in our experiments. This suggests that a well-trained representation can remain sufficiently informative, provided that the decision module is allowed to adapt. Such behavior is intuitive in the ***A_class*** scenario, where the input distribution itself remains unchanged, despite shifts in class composition. In the ***A_loc*** setting, although the temporal position of the discriminative pattern changes, the nature and scale of the signal remain consistent, and the drift range remains limited, enabling the decision module to adjust its policy without requiring an update of the underlying representation.

Table 4: (*a*): The mean percentage difference between the cumulative cost of full updated DQeND and the partially frozen version, over five random seeds. (*b*): The mean of the median percentage difference of the hold-out *AvgCost* drift of the full updated DQeND and partially frozen version, over five random seeds. Negative figures indicate degraded performance compared to original DQeND.

	freeze **encoder**		*freeze* **decision**	
	(*a*)	(*b*)	(*a*)	(*b*)
I_loc	-32.42%	-36.44%	-36.61%	-31.87%
I_noise	-11.29%	-4.04%	-19.28%	-6.63%
A_loc	1.38%	-0.34%	-43.52%	-45.30%
A_class	1.32%	2.14%	-18.18%	-4.39%

C.2 Exploration rate ϵ

We further test the sensitivity of DQeND to the exploration rate ϵ in two of the four synthetic scenarios, i.e. ***I_noise*** and ***A_class***. Figures reported in Table 5 show that the best exploration rate depends on the scenario: while less exploration seems to work better for the ***A_class*** scenario, more exploration may be beneficial in the ***I_noise*** one.

Table 5: DQeND sensitivity to ϵ.

		0%	1%	5%	10%	20%
I_noise	*Cumulative cost*	37.52 ± 1.54	36.91 ± 2.22	36.2 ± 1.04	$\mathbf{35.99} \pm 0.5$	36.72 ± 0.73
	AvgCost drift	0.38 ± 0.01	0.37 ± 0.02	0.36 ± 0.01	$\mathbf{0.36} \pm 0.01$	0.37 ± 0.01
	AvgCost original	0.33 ± 0.03	0.35 ± 0.04	$\mathbf{0.31} \pm 0.01$	0.32 ± 0.02	0.35 ± 0.02
A_class	*Cumulative cost*	31.72 ± 3.30	$\mathbf{29.47} \pm 3.74$	32.98 ± 3.88	34.07 ± 2.95	33.67 ± 3.4
	AvgCost drift	0.32 ± 0.03	$\mathbf{0.29} \pm 0.04$	0.33 ± 0.04	0.34 ± 0.03	0.34 ± 0.03
	AvgCost original	0.39 ± 0.04	0.39 ± 0.04	0.4 ± 0.04	0.37 ± 0.03	$\mathbf{0.36} \pm 0.03$

D VARIABILITY REPORT

In this section, the evaluation metrics displayed in Section 5 are detailed in Tables 6, 7, and 8, and variability is reported.

Table 6: The mean cumulative cost and standard deviation, over five random seeds. **Bold** value indicates the best average performing method.

	I_loc	***I_noise***	***A_loc***	***A_class***
Alert(*online/freeze*)	29.02 ± 4.08	46.14 ± 0.3	31.06 ± 11	32.15 ± 1.27
Alert(*online/online*)	31.87 ± 5.49	37.92 ± 0.83	23.32 ± 1.88	30.86 ± 0.2
DQeND	**16.18** ± 3.40	**36.2** ± 1.04	**18.86** ± 0.55	32.98 ± 3.88
EARLIEST	20.53 ± 6.54	43 ± 3.37	19.90 ± 0.79	32.44 ± 5.88
ELECTS	17.33 ± 1.81	38.63 ± 1.98	29.98 ± 12.06	**25.79** ± 3.52

Table 7: The mean hold-out *AvgCost* drift and standard deviation, over all the deployment steps and five random seeds. **Bold** value indicates the best average performing method.

	I_loc	***I_noise***	***A_loc***	***A_class***
Alert(*online/freeze*)	0.3 ± 0.04	0.46 ± 0.003	0.31 ± 0.11	0.32 ± 0.01
Alert(*online/online*)	0.31 ± 0.07	0.38 ± 0.009	0.23 ± 0.02	0.31 ± 0.002
DQeND	**0.16** ± 0.03	**0.36** ± 0.01	**0.19** ± 0.01	0.33 ± 0.04
EARLIEST	0.2 ± 0.06	0.43 ± 0.03	0.2 ± 0.01	0.32 ± 0.06
ELECTS	0.17 ± 0.02	0.39 ± 0.02	0.3 ± 0.12	**0.26** ± 0.04

Table 8: The mean hold-out *AvgCost* original and standard deviation, over all the deployment steps and five random seeds. **Bold** value indicates the best average performing method.

	I_loc	***I_noise***	***A_loc***	***A_class***
Alert(*online/freeze*)	0.31 ± 0.02	0.31 ± 0.04	0.21 ± 0.04	**0.37** ± 0.03
Alert(*online/online*)	0.33 ± 0.06	0.33 ± 0.02	0.18 ± 0.02	0.37 ± 0.003
DQeND	**0.09** ± 0.04	**0.31** ± 0.01	**0.11** ± 0.02	0.4 ± 0.04
EARLIEST	0.27 ± 0.07	0.36 ± 0.01	0.11 ± 0.07	0.51 ± 0.01
ELECTS	0.14 ± 0.07	0.33 ± 0.01	0.19 ± 0.1	0.48 ± 0.02

E RUNTIME ANALYSIS

Computation times are reported in Table 9. Overall, ELECTS and EARLIEST are the fastest approaches, both during training and deployment. Unlike Alert and DQeND, which rely on convolution-based encoders, these methods are LSTM-based and therefore benefit from the CPU-only experimental setup used in this work. In practice, convolutional approaches would likely reduce this gap when leveraging GPU acceleration. For Alert-based methods indeed, most of the training time is spent fitting the ELM-based classifier rather than the trigger module itself, confirming the relatively low computational overhead of the RL trigger. In contrast, DQeND exhibits the highest training and update times.

More broadly, the deployment times of DQeND remain significantly longer. This is consistent with the known sample inefficiency of value-based RL algorithms, whose repeated updates and replay-based optimization can constitute computational limitations in online settings.

Table 9: Computation times (in *s.*). The deployment times are reported on a *per batch* basis. The total time accounts for the all 100 deployment steps. The fit time of the Alert-based methods are decomposed over (*classifier* fit time, *trigger* fit time).

	Training	**Deployment** (*per batch*)		**Total** ↓
	Fit	Inference	Update	
ELECTS	31.89	0.026	0.05	39.22
EARLIEST	119.25	0.23	0.09	138.98
Alert (*online*/*online*)	(78.02, 46.55)	2.39	2.45	612.75
Alert (*retrain*/*retrain*)	(78.02, 46.55)	2.71	3.59	759.86
DQeND	248.74	2.90	5.75	1124.31