Incremental Sequence Classification with Temporal Consistency

Lucas Maystre[*] UiPath London, UK
Gabriel Barello UiPath Bellevue, WA, USA
Tudor Berariu UiPath London, UK
Aleix Cambray UiPath London, UK
Rares Dolga UiPath & UCL London, UK
Alvaro Ortega Gonzalez UiPath London, UK
Andrei Nica UiPath London, UK
David Barber UiPath & UCL London, UK

Abstract

We address the problem of incremental sequence classification, where predictions are updated as new elements in the sequence are revealed. Drawing on temporal-difference learning from reinforcement learning, we identify a temporal-consistency condition that successive predictions should satisfy. We leverage this condition to develop a novel loss function for training incremental sequence classifiers. Through a concrete example, we demonstrate that optimizing this loss can offer substantial gains in data efficiency. We apply our method to text classification tasks and show that it improves predictive accuracy over competing approaches on several benchmark datasets. We further evaluate our approach on the task of verifying large language model generations for correctness in grade-school math problems. Our results show that models trained with our method are better able to distinguish promising generations from unpromising ones after observing only a few tokens.

1 Introduction

Learning to classify a sequence $\boldsymbol{x} = (x_1, \ldots, x_T)$ into one of K classes is a fundamental problem in machine learning [44, 16]. In this work, we focus on an incremental variant in which the sequence is revealed element by element, and a predictive model must provide predictions at every timestep. As a concrete example, consider a movie review, represented as a sequence of tokens.

$$\underbrace{\texttt{A}}_{x_1}\ \underbrace{\texttt{touching}}_{x_2}\ \underbrace{\texttt{movie}}_{x_3}\underbrace{\texttt{.}}_{x_4}\ \underbrace{\texttt{It}}_{x_5}\ \underbrace{\texttt{is}}_{x_6}\ \underbrace{\texttt{full}}_{x_7}\ \underbrace{\texttt{of}}_{x_8}\ \underbrace{\texttt{emotions}}_{x_9}\ \underbrace{\texttt{and}}_{x_{10}}\ \underbrace{\texttt{wonderful}}_{x_{11}}\ \underbrace{\texttt{acting}}_{x_{12}}\underbrace{\texttt{.}}_{x_{13}}$$

We ask the question: Can we predict the sentiment of this review using only the first two or three tokens? More generally, can we learn a predictive model that accurately classifies any prefix $\boldsymbol{x}_{\leq t}$, consisting of the first $t \leq T$ elements of a sequence? This problem naturally arises in domains where there is a cost associated to waiting for the full sequence; In healthcare or finance, for example, the cost might be time or opportunity [45, 34]. Recently, sequence classifiers have also been deployed as verifiers to improve applications of large language models (LLMs), where generating full sequences incurs a non-trivial computational cost [8, 39, 31].

A key property of the incremental classification problem is that any calibrated predictive model should be *temporally-consistent*. That is, the predictive class distribution given a prefix $\boldsymbol{x}_{\leq t}$ should equal the expected class distribution given the extended prefix $\boldsymbol{x}_{\leq t+1}$, where the expectation is taken over x_{t+1}. We take advantage of this temporal-consistency property to develop a novel loss function for training

[*]Corresponding author, e-mail: lucas@maystre.ch.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

incremental classifiers (Section 2). Our approach shares many parallels with temporal-difference (TD) learning [37], an important class of methods in reinforcement learning (RL) [38]. Exploiting these parallels, we study a simple sequence model and demonstrate that optimizing our loss function can lead to substantial data-efficiency gains over the standard maximum-likelihood approach (Section 3).

We apply our method to text classification using decoder-only transformers (Section 4). We first evaluate it on four benchmark datasets and find that models trained with our temporal-consistency loss achieve higher predictive accuracy—both on prefixes and on full sequences. When fine-tuning models of the OPT family [49], the improvement in predictive performance is comparable to increasing the size of the base model by a factor 10. We then explore the emerging application of verifying LLM generations. On GSM8K math word problems [8], our method yields verifiers that accurately distinguish correct from incorrect generations after observing only a few tokens. We illustrate how this enables a more favorable trade-off between answer accuracy and computational cost.

Our approach is inspired by celebrated methods in RL and is rigorously grounded in theory. It is simple to implement, adds negligible computational overhead during training and inference, and improves predictive performance across all the datasets we evaluated. As such, we believe it will be valuable to machine-learning practitioners.

1.1 Related work

Early sequence classification [44, 45, 27] addresses a problem that is distinct but complementary to ours: determining *when* enough information has been observed to make a reliable prediction. In contrast, we focus exclusively on *what* to predict at each prefix. Our goal is to maximize classification accuracy across all timesteps, without addressing the decision-making aspect. We note that incremental classifiers trained with our method could serve as components in early classification architectures [34, 4].

TD learning [37, 38] leverages a temporal-consistency condition, closely related to ours, to learn value functions in RL. Classical TD learning is concerned with modeling the reward-to-go, a scalar quantity. Our work can be understood as extending the core idea underpinning TD learning to the multiclass classification setting. Distributional RL [29, 2, 9] shares some algorithmic features, but does not address classification. Cheikhi and Russo [7] recently analyzed the data-efficiency benefits of TD learning, and we build on their results in Section 3. Beyond scalar values, ideas from TD learning have also been applied to survival analysis [26, 40] and early event prediction [48].

Incremental classifiers have recently been applied to verify LLM generations, either during post-training [39, 21] or at inference time [8, 47, 31], where they are referred to as token-level verifiers [8] or outcome-supervised reward models [39, 21]. Verification is typically framed as a binary classification problem, with verifiers trained using a cross-entropy loss against the final observed outcome, a baseline we refer to as *direct cross-entropy* (DCE). Some prior work use soft targets, as we do, but these targets are still derived solely from observed outcomes [41, 20]. A notable exception is Mudgal et al. [31], who frame verification as a regression problem and learn a token-level value function using TD learning. However, they rely on a squared loss, which is ill-suited to binary outcomes [18]. Our work bridges this gap by developing a TD-style approach tailored to classification.

2 Direct and temporally-consistent estimators

In order to introduce our approach to learning incremental sequence classifiers, we shift our perspective slightly. Instead of a sequence of arbitrary elements, which we have denoted by $\boldsymbol{x} = (x_1, \ldots, x_T)$ in the introduction, we now consider a sequence of Markov states $\boldsymbol{s} = (s_1, \ldots, s_T)$. That is, we assume that the ground-truth distribution $p(\boldsymbol{s}, y)$ of states and class label satisfies

$$p(s_{t+1} \mid s_t, \ldots, s_1) = p(s_{t+1} \mid s_t), \qquad p(y \mid s_t, \ldots, s_1) = p(y \mid s_t), \tag{1}$$

for any $t < T$, and for any $t \leq T$ and any y, respectively. This is not a restrictive assumption: By slight abuse of notation, one can write $s_t = \boldsymbol{x}_{\leq t}$ and $p(s_{t+1} \mid s_t) = p(x_{t+1} \mid \boldsymbol{x}_{\leq t})$, and trivially satisfy these two Markov properties. The Markov-chain perspective will be useful to relate our developments to temporal-difference learning, and to provide intuition into the statistical benefits of our method in Section 3.

Notation and problem setup Let $[M]$ denote the set of consecutive integers $1, \ldots, M$. We are given a dataset of N labelled sequences $\mathcal{D} = \{(\boldsymbol{s}^n, y^n) : n \in [N]\}$, where $y^n \in [K]$, and where different sequences might be of different lengths. We seek to learn a probabilistic classifier $p_{\boldsymbol{\theta}}(y \mid s_t)$, parametrized by $\boldsymbol{\theta}$, that approximates the ground-truth distribution $p(y \mid s_t)$. We denote by $\boldsymbol{p}_{\boldsymbol{\theta}}(\cdot \mid s_t)$ the K-dimensional probability vector $[p_{\boldsymbol{\theta}}(1 \mid s_t) \quad \cdots \quad p_{\boldsymbol{\theta}}(K \mid s_t)]$, and by $\boldsymbol{\delta}_y$ the K-dimensional one-hot vector with a 1 in position y.

Direct cross-entropy A natural approach for learning $p_{\boldsymbol{\theta}}(y \mid s_t)$ is to maximize the likelihood of the observed samples from $p(y \mid s_t)$ in the dataset $\mathcal{D}$, or equivalently, to minimize the cross-entropy relative to these samples. Given a labeled sequence $(\boldsymbol{s}, y)$, we define the following loss function, which penalizes deviations from the label y simultaneously across all $t \leq T$.

$$\ell_{\text{DCE}}(\boldsymbol{\theta}; \boldsymbol{s}, y) = -\sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(y \mid s_t) = \sum_{t=1}^{T} H\big[\boldsymbol{\delta}_y \,\big\|\, \boldsymbol{p}_{\boldsymbol{\theta}}(\cdot \mid s_t)\big], \tag{2}$$

where $H[\boldsymbol{p} \| \boldsymbol{q}] = -\sum_{k=1}^{K} p_k \log q_k$ is the cross-entropy function. Given the full dataset $\mathcal{D}$, we find

$$\boldsymbol{\theta}^{\star}_{\text{DCE}} \leftarrow \arg\min_{\boldsymbol{\theta}} \textstyle\sum_n \ell_{\text{DCE}}(\boldsymbol{\theta}; \boldsymbol{s}_n, y_n). \tag{3}$$

We call this approach *direct cross-entropy* (DCE), because the target distribution is the observed one-hot class label $\boldsymbol{\delta}_y$. In LLM verification applications, this is the predominant approach to training token-level verifiers [8, 39, 21, 41, 20].

2.1 A family of temporally-consistent estimators

Intuitively, a drawback of the direct approach is that, for early states s_t (with $t \ll T$), the training signal provided by observed label y is noisy. Indeed, the prediction $p_{\boldsymbol{\theta}}(y \mid s_t)$ needs to account for two sources of uncertainty, *a*) the uncertainty about how the remainder of the sequence $s_{t+1}, \ldots, s_T$ will unfold, and *b*) the uncertainty about the label y given the last state s_T. We make progress by observing that

$$p(y \mid s_t) = \mathbf{E}_{p(s_{t+1} \mid s_t)}[p(y \mid s_{t+1})], \tag{4}$$

for all y and all $t < T$, an identity that follows from (1). This identity captures a notion of *temporal consistency*: It states that the class distribution at step t is equal to the class distribution at step $t+1$ on average. Driven by this observation, we propose the following loss function, which for $t < T$ penalizes the temporal inconsistency relative to a reference model parametrized by $\boldsymbol{\theta}'$:

$$\ell_{\text{TC}}(\boldsymbol{\theta}; \boldsymbol{\theta}', \boldsymbol{s}, y) = H\big[\boldsymbol{\delta}_y \,\big\|\, \boldsymbol{p}_{\boldsymbol{\theta}}(\cdot \mid s_T)\big] + \sum_{t=1}^{T-1} H\big[\boldsymbol{p}_{\boldsymbol{\theta}'}(\cdot \mid s_{t+1}) \,\big\|\, \boldsymbol{p}_{\boldsymbol{\theta}}(\cdot \mid s_t)\big]. \tag{5}$$

Comparing (2) and (5) carefully, we can think of this temporal-consistency (TC) loss as replacing the hard targets $\boldsymbol{\delta}_y$ by soft targets $\boldsymbol{p}_{\boldsymbol{\theta}'}(\cdot \mid s_{t+1})$ capturing the predictive distribution at the next state. Given the full dataset $\mathcal{D}$, we start with random parameters $\boldsymbol{\theta}^{(0)}$, and iteratively solve

$$\boldsymbol{\theta}^{(i+1)} \leftarrow \arg\min_{\boldsymbol{\theta}} \textstyle\sum_n \ell_{\text{TC}}(\boldsymbol{\theta}; \boldsymbol{\theta}^{(i)}, \boldsymbol{s}_n, y_n). \tag{6}$$

In Section 3, we study a tractable setting and show that this iteration converges to a fixed point $\boldsymbol{\theta}^{\star}_{\text{TC}}$, and that the estimator is consistent, i.e., that $p_{\boldsymbol{\theta}^{\star}_{\text{TC}}}(y \mid s_t) \to p(y \mid s_t)$ as the dataset size $N \to \infty$.

We can extend the identity (4) to multiple steps, capturing the temporal consistency of class distributions across longer time spans (c.f. Appendix A.1). We use this to formulate a generalized temporal-consistency loss,

$$\begin{aligned} \ell_{\text{TC-}\lambda}(\boldsymbol{\theta}; \boldsymbol{\theta}', \boldsymbol{s}, y) &= \textstyle\sum_{t=1}^{T} H\big[\boldsymbol{z}_t \,\big\|\, \boldsymbol{p}_{\boldsymbol{\theta}}(\cdot \mid s_t)\big], \\ \boldsymbol{z}_t &= \lambda^{T-t}\boldsymbol{\delta}_y + (1-\lambda)\textstyle\sum_{k=1}^{T-t} \lambda^{k-1} \boldsymbol{p}_{\boldsymbol{\theta}'}\big(\cdot \mid s_{t+k}\big), \end{aligned} \tag{7}$$

where $\lambda \in [0, 1]$ is a hyperparameter. In this loss function, the target $\boldsymbol{z}_t$ providing the training signal is a weighted average of the predictive distributions k steps ahead, with exponentially decreasing weights. The larger λ is, the larger the influence of distant states is.[1] In fact, TC-λ generalizes both TC (5) and DCE (2). Setting $\lambda = 0$ recovers TC, while $\lambda = 1$ yields DCE. Throughout the paper, we refer to any model trained with the TC-λ loss with $\lambda < 1$ as *temporally consistent*.

[1] We can think of the mean of the geometric distribution $\lambda/(1-\lambda)$ as the effective lookahead, i.e., as "how far" the target looks ahead on average.

2.2 Connections to temporal-difference learning

Our approach is inspired by temporal-difference (TD) learning, a key idea in reinforcement learning [37, 38, 28]. Quoting Sutton's seminal paper, "whereas conventional prediction-learning methods assign credit by means of the difference between predicted and actual outcomes, the new methods assign credit by means of the difference between temporally successive predictions." Recasting our developments into the language of reinforcement learning, we can think of the probabilistic classifier we learn as a state-value function, capturing the eventual final outcome of a trajectory at any given intermediate state. The temporal-consistency condition (4) is a form of Bellman equation, relating the value function across successive states. Similarly to TD learning, our approach uses the temporal inconsistency of predictions across successive states as the learning signal. Our generalized loss function (7) is inspired by the TD(λ) family of algorithms.

A key difference is that we consider categorical outcomes and a cross-entropy loss, as opposed to the scalar outcomes and squared loss usually employed in TD learning algorithms. In Section 4, we compare our approach to classical value-estimation methods from RL, by treating our K-way classification problem as K separate value estimation problems.

3 Convergence, consistency and data efficiency

In this section, we analyze the DCE and TC estimators in problems with a finite number of states. We consider tabular models, where a separate probability is learned for each state-class pair. This tractable setting enables a theoretical comparison of the properties of DCE and TC. Specifically, we show that *a*) the TC optimization procedure in (6) converges and *b*) yields a consistent estimator of the true class probabilities, and that *c*) the TC estimator is more data-efficient than DCE. Complete proofs of all propositions are provided in Appendix A.2. Our perspective in this section is partly inspired by dynamic programming and its application to stochastic shortest path problems, as presented in Bertsekas and Tsitsiklis [3, Sec. 2.2].

In the finite-state case, we formalize the problem of multiclass sequence classification as that of finding the absorption probabilities of a Markov chain on M transient states and K absorbing states [32]. The transient states correspond to M distinct values each element of the sequence $\boldsymbol{s}$ can take, and the absorbing states correspond to K classes. The Markov chain is fully characterized by the pair $(\boldsymbol{Q}, \boldsymbol{R})$, where the $M \times M$ matrix $\boldsymbol{Q}$ describes the transition probabilities between every pair of transient states, and the $M \times K$ matrix $\boldsymbol{R}$ describes the transition probabilities from transient to absorbing states. We assume that it is possible to reach a terminal state from any transient state within a finite number of steps. That is, for any transient state m, there is a $t \geq 0$ such that $[\boldsymbol{Q}^t \boldsymbol{R}]_{mk} > 0$ for some k. Our goal is to estimate the absorption probabilities $p^\star_{mk} = p(y = k \mid s_t = m)$ from data, organized into the $M \times K$ matrix $\boldsymbol{P}^\star$.

Before addressing the estimation problem, note that if the ground-truth transition matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are known, $\boldsymbol{P}^\star$ can be computed by starting from an initial guess $\boldsymbol{P}_0$ and refining it iteratively as

$$\boldsymbol{P}_{i+1} = \boldsymbol{Q}\boldsymbol{P}_i + \boldsymbol{R}. \tag{8}$$

Proposition 1. *For any $M \times K$ row-stochastic $\boldsymbol{P}_0$, the fixed-point iteration* (8) *converges to $\boldsymbol{P}^\star$.*

Now consider the setting where, instead of access to $\boldsymbol{Q}$ and $\boldsymbol{R}$, we are given a dataset $\mathcal{D} = \{(\boldsymbol{s}^n, y^n) : n \in [N]\}$ of trajectories sampled from the Markov chain. Each trajectory is composed of a transient sequence $\boldsymbol{s}^n$ and terminates in an absorbing state y^n. We explore two approaches to estimating the absorption probabilities.

Direct estimation We can estimate the absorption probabilities directly, as the empirical fraction of sequences passing through m ending in k. Denoting by $T(\boldsymbol{s})$ the length of sequence $\boldsymbol{s}$ and letting $\mathcal{D}' = \cup_{(\boldsymbol{s},y)\in\mathcal{D}} \cup_{t \leq T(\boldsymbol{s})} \{(s_t, y)\}$ be the union of all pairs of transient state and eventual absorbing state in $\mathcal{D}$, we have

$$\hat{p}^{\text{dir}}_{mk} = \mathbf{E}_{(s,y)\sim\mathcal{D}'}[\mathbf{1}_{\{y=k\}} \mid s = m],$$

where $(s, y) \sim \mathcal{D}'$ denotes uniform sampling on $\mathcal{D}'$. We collect these estimates into the matrix $\hat{\boldsymbol{P}}^{\text{dir}}$.

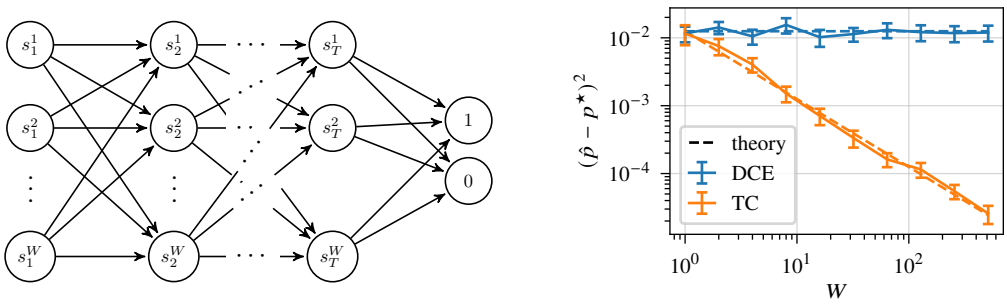

Figure 1: *Left*: Markov chain with T layers of W states each, and two absorbing states. *Right*: Mean-squared error of the direct (DCE) and indirect (TC) estimates for a state in the first layer state as a function of W. We set $N = 20W$ and report the mean and 95% confidence intervals over 100 runs.

Indirect estimation Alternatively, we can first estimate the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ by using the empirical one-hop transition counts. Letting $\mathcal{A} = \cup_{(\boldsymbol{s},y)\in\mathcal{D}} \cup_{t<T(s)} \{(s_t, s_{t+1})\}$ and $\mathcal{B} = \cup_{(\boldsymbol{s},y)\in\mathcal{D}} \{(s_{T(\boldsymbol{s})}, y)\}$ be the union of all transitions between successive transient states and between transient and absorbing states, respectively, and $\mathcal{T} = \mathcal{A} \cup \mathcal{B}$, we write

$$\hat{q}_{mm'} = \mathbf{E}_{(s,s')\sim\mathcal{T}}[\mathbf{1}_{\{s'=m'\}} \mid s = m], \qquad \hat{r}_{mk} = \mathbf{E}_{(s,s')\sim\mathcal{T}}[\mathbf{1}_{\{s'=k\}} \mid s = m]. \tag{9}$$

We can then plug the estimates $\hat{\boldsymbol{Q}}$ and $\hat{\boldsymbol{R}}$ into the fixed-point iteration (8), and iterate until convergence. We denote the resulting estimate by $\hat{\boldsymbol{P}}^{\text{ind}}$. By Proposition 1, this fixed point corresponds to the exact absorption probability matrix of the empirical Markov chain $(\hat{\boldsymbol{Q}}, \hat{\boldsymbol{R}})$, but it is only an approximation of the ground-truth the absorption probabilities.

The following proposition relates the TC loss (5) and its iterative optimization procedure (6), introduced in Section 2, to the indirect estimator $\hat{\boldsymbol{P}}^{\text{ind}}$ we have just derived.

Proposition 2. *Let $p_{\boldsymbol{\theta}}(y = k \mid s_t = m) \doteq \theta_{mk}$. Then, the TC iterative optimization procedure* (5) *is equivalent to the fixed-point iteration* (8) *on the empirical Markov chain $(\hat{\boldsymbol{Q}}, \hat{\boldsymbol{R}})$ defined by* (9).

It follows that, in the tabular case, the TC estimator (6) is guaranteed to converge. Furthermore, under mild assumptions on the data-generating distribution[2], the TC estimator is consistent, i.e.,

$$\hat{\boldsymbol{P}}^{\text{ind}} \to \boldsymbol{P}^\star \text{ as } N \to \infty.$$

Similarly, we can relate the optimization of the DCE loss (2) to the direct estimator $\hat{\boldsymbol{P}}^{\text{dir}}$ (see Proposition 4 in Appendix A).

3.1 Statistical benefits

In general, DCE and TC will not result in the same estimate, i.e., $\hat{\boldsymbol{P}}^{\text{ind}} \neq \hat{\boldsymbol{P}}^{\text{dir}}$, raising the question: Which estimator is better? In related work, Cheikhi and Russo [7] study direct and indirect estimators for the state-value function of a Markov reward process. They find that, depending on the structure of the Markov reward process, the indirect estimator can be significantly more data-efficient than (and is always at least as efficient as) the direct estimator.

In Figure 1, we revisit one of their examples and adapt it to our classification setting. We consider an absorbing Markov chain with T layers of W distinct states each. For any $t < T$, the probability of transitioning from a state at layer t to any other state at layer $t + 1$ is $1/W$. From any state at layer T, we transition to one of two absorbing states, 0 and 1, with probability $1/2$ each. Given this, it is easy to verify that $p^\star_{mk} = 1/2$ for any transient state m and any absorbing state $k \in \{0, 1\}$. We collect N trajectories of length $T + 1$, sampling the first state uniformly at random, and subsequent states as described previously. The next proposition shows that, for any state in the first layer, the expected squared error of the indirect estimator is W times smaller than that of the direct estimator.

[2]Assuming that the initial state distribution and the transition probabilities are such that, for every transient state, there is a non-zero probability of it being sampled in the sequence.

Proposition 3 (Adapted from [7]). *For any $T \geq 1$ and any state m in the first layer,*

$$\mathbf{E}\left[(\hat{p}^{\text{ind}}_{mk} - p^{\star}_{mk})^2\right] \;/\; \mathbf{E}\left[(\hat{p}^{\text{dir}}_{mk} - p^{\star}_{mk})^2\right] \xrightarrow{N\to\infty} 1/W.$$

We validate this result empirically in Figure 1 (right), where we report on numerical simulations using $N = 20W$. In this Markov chain, the indirect estimator is increasingly more data-efficient as W increases. Intuitively, the indirect approach acts as a form of regularization, by requiring the solution to satisfy the temporal-consistency property (4), which arises from the problem's Markov structure. The indirect approach benefits from a form of data pooling: By minimizing the inconsistency across successive states instead of regressing the target outcome directly, we take advantage of information from other trajectories that originated from different states and crossed paths.

Beyond the tabular setting In the remainder of this paper, we move beyond the tabular setting studied above and fine-tune parametric large language models on very large state spaces. The theoretical results developed in this section no longer strictly apply, yet we believe that the underlying intuition remains valuable for interpreting our method's performance on complex, real-world tasks. In text classification, for example, many distinct word sequences can result in a similar *semantic state* predictive of the target label. Optimizing for consistency across successive states can therefore improve data efficiency, similarly to the tabular toy example. With temporal consistency, the model implicitly exploits information from sequences that begin differently but reach similar intermediate semantic states, the same data-pooling phenomenon underpinning the gains of the indirect method in Figure 1.

4 Empirical evaluation

In this section, we apply our methodology to text classification with decoder-only transformers. First, in Section 4.1, we evaluate multiple different approaches to training incremental classifiers. We compare the predictive performance of models on four well-known text classification benchmarks. Then, in Section 4.2, we consider a concrete application to verifying LLM generations. We show that an accurate token-level correctness classifier enables solving grade-school math problems computationally more efficiently.

Model architecture & training Decoder-only (i.e., causal) transformers [23, 33] are particularly well-suited to incremental sequence classification. As the tth output of a decoder on an input sequence $\boldsymbol{x}$ depends only on the prefix $\boldsymbol{x}_{\leq t}$, we can compute predictions at every prefix efficiently, with a single forward inference pass. Throughout this section, we model class probabilities as

$$\boldsymbol{p}_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x}_{\leq t}) = \text{softmax}(\boldsymbol{A}\boldsymbol{h}_t + \boldsymbol{b}), \tag{10}$$

where $\boldsymbol{h}_t \in \mathbf{R}^D$ is the hidden vector at the last layer of a transformer at position t, and $\boldsymbol{A} \in \mathbf{R}^{K\times D}$ and $\boldsymbol{b} \in \mathbf{R}^K$ are the parameters of a classification head. We start with a pre-trained language model, and jointly optimize $(\boldsymbol{A}, \boldsymbol{b})$ as well as all the parameters of the transformer, which we collectively refer to as $\boldsymbol{\theta}$. We make two minor practical adjustments with respect to the optimization procedure outlined in Section 2. First, we update the parameters using stochastic gradient updates. Second, we average the loss over all prefixes of each sequence, instead of summing them. This means that every sequence contributes to the loss equally, irrespective of its length. Appendix B describes the precise training procedure and documents how we select hyperparameters.

4.1 Text classification datasets

We consider four text classification datasets, spanning tasks such as movie review sentiment prediction (IMDB [25]) and topic classification (OHSUMED [30], NEWSGROUPS [19], AG-NEWS [10]). The number of classes K ranges from 2 to 23. We provide summary statistics for all datasets in Appendix B.2, including the number of training and test samples and the distribution of document length. In addition to the standard setting, where the goal is to predict the class label given the full sequence $\boldsymbol{x}$, we are interested in evaluating the performance of classifiers in the *incremental* setting, where we need to make a prediction after observing only a prefix $\boldsymbol{x}_{\leq t}$ consisting of the first t tokens.

We fine-tune pre-trained language models from the OPT family [49]. Unless otherwise noted, we report results on the $125\,\text{M}$-parameter version of the family, with hidden size $D = 768$. Our primary

Table 1: Predictive performance of incremental text classifiers on four datasets. We report the classification accuracy on 4-token and 16-token prefixes, and on full sequences (all tokens). We highlight the **best** and second-best performing models.

	OHSUMED			NEWSGROUPS			IMDB			AG-NEWS		
	4	16	all	4	16	all	4	16	all	4	16	all
Most frequent	16.0	16.0	16.0	5.3	5.3	5.3	50.0	50.0	50.0	25.0	25.0	25.0
GPT 4o	31.5	54.0	57.5	7.5	11.0	80.4	58.0	67.0	94.3	77.4	87.4	88.3
Filtering	22.1	46.9	46.2	10.4	19.8	72.0	64.1	73.4	92.1	77.9	86.6	87.2
Last token	16.7	45.0	80.6	6.5	9.4	87.9	56.6	68.4	94.7	54.4	78.3	94.8
Specialist	24.5	59.8	80.6	23.6	**47.3**	87.9	59.8	73.3	94.7	77.8	91.8	94.8
Direct ℓ_2 loss	30.3	65.0	80.4	19.7	29.9	88.3	63.6	74.7	94.3	80.0	92.6	94.7
LSTD(λ)	32.7	64.9	78.0	26.2	36.7	87.8	64.6	75.4	94.7	81.1	92.8	94.9
DCE	30.5	65.5	81.1	27.7	40.1	**89.0**	63.5	74.7	94.4	80.0	92.6	94.8
TC-λ (ours)	**33.7**	**68.3**	**81.8**	**33.4**	44.7	88.5	**64.7**	**75.7**	**94.9**	**81.4**	**93.0**	**95.0**

focus is on comparing models trained by using the DCE (2) and TC-λ (7) loss functions. Both of these approaches train a single model to classify prefixes of any length (including the full sequence). In addition, we also consider the following baselines and competing methods.

Most frequent A naive baseline that always predicts the most frequent class in the training split.

GPT-4o We design a simple prompt asking GPT-4o (version 2024-08-06) to classify a text, by giving the list of classes, one example for each class, and the text itself. We access GPT-4o through OpenAI's commercial API. Details are provided in Appendix B.2.

Filtering We fine-tune a class-conditional language model $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid y)$ on the training data, by using a standard next-token prediction task. At test time, we use Bayes' rule to reverse the conditional probability as $\hat{p}(y \mid x) = Z^{-1} p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid y) p(y)$, where $p(y)$ is a prior distribution and $Z = \sum_y p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid y) p(y)$.

Last token Similarly to the DCE loss (2), we minimize the cross-entropy relative to the observed label. But unlike the DCE loss, we include only a single cross-entropy term per sequence, corresponding to the prediction based on the hidden vector $\boldsymbol{h}_T$ at the last token (i.e., capturing the full sequence). To the best of our knowledge, this is the most widespread approach to text classification with decoder-only transformers [43, 12, 17].

Specialist This approach is similar to the *last token* method, but improves upon it by training a distinct, specialized model for each prefix length. Instead of full sequences, each model is trained exclusively on prefixes of the corresponding length.

Direct ℓ_2 loss This approach is similar to DCE but removes the softmax and replaces the cross-entropy loss with a squared loss. This is analogous to standard offline Monte Carlo methods for value function estimation in RL [38, Chap. 5].

LSTD(λ) The offline temporal-difference value estimation method of Bradtke and Barto [5], recently applied to language modelling in Mudgal et al. [31]. This approach is similar to TC-λ, without the softmax and with a squared loss instead of the cross-entropy loss.

For TC-λ and LSTD(λ), we treat λ as a hyperparameter. Values for this and for all other hyperparameters are presented in Appendix B.2. The time it takes to train a TC-λ model is virtually indistinguishable (within 1%) from that used to train a DCE model, confirming that the overhead required to compute the soft targets $\{\boldsymbol{z}_t\}$ in (7) is negligible compared to the cost of LLM forward and backward passes.

In Table 1, we report the predictive accuracy of classifiers on hold-out sequences, given prefixes of 4 tokens, 16 tokens, and full sequences. For DCE and TC-λ and the corresponding squared-loss methods, we report additional metrics and more prefix lengths in the appendix. As expected, the accuracy of every classifier increases as more tokens are available. However, even with only 4 or 16 tokens (often representing a small fraction of the full sequence), some approaches reach a non-trivial accuracy.

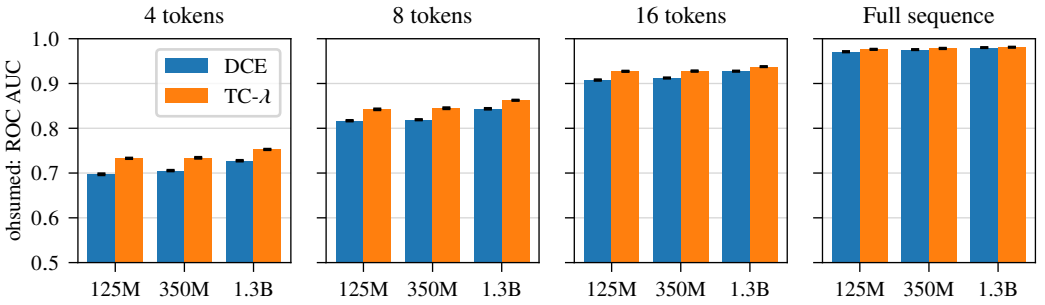

Figure 2: Predictive performance of OPT models with 125 M, 350 M, and 1.3 B parameters, respectively, on the OHSUMED dataset. We report the area under the ROC curve (mean and 95% confidence interval over 10 runs; higher is better).

We observe that TC-λ outperforms DCE and other approaches in all but two cases, where it ranks second. This supports the insights from Section 3.1: incorporating temporal-consistency into the loss function improves predictive performance, even on real-world sequences and with large parametric models. Gains are more substantial for short prefixes, but—perhaps surprisingly—optimizing for temporal-consistency also improves full-sequence classification accuracy. Relatedly, we observe that training incremental classifiers is beneficial even when the goal is only to classify full sequences. Indeed, both DCE and TC-λ outperform the *last-token* approach in this setting. This was noticed by Cobbe et al. [8], who suggest that including prefixes in the loss provides a useful auxiliary signal. Our findings reinforce this observation. Finally, we note that the *filtering* method generally underperforms approaches using a classification head, consistent with conventional wisdom that, for classification problems, discriminative models can be more effective than generative models [1].

Cross-entropy vs. squared loss When comparing DCE and TC-λ, which use a cross-entropy loss, to their squared-loss counterparts (direct ℓ_2 loss and LSTD(λ), respectively), we observe the following. Temporal consistency tends to improves performance regardless of the loss function, especially on partial sequences. Models trained with a cross-entropy loss perform significantly better than those trained with a squared loss on datasets with many classes (OHSUMED and NEWSGROUPS). For binary or three-way classification tasks (IMDB and AG-NEWS), both loss functions yield a similar accuracy. However, Figure 6 in the appendix shows that methods optimizing a squared loss produce noticeably less well-calibrated predictive probabilities.

Increasing model size Focusing on the OHSUMED dataset and the DCE and TC-λ methods, we train OPT models with 125 M, 350 M, and 1.3 B parameters. Figure 2 shows the area under the ROC curve (ROC AUC) for predictions after 4, 8, and 16 tokens, as well as for full sequences. These results offer the following perspective on the benefits of temporal consistency: DCE requires a model that is approximately $10\times$ larger to match the performance of TC-λ.

Varying the temporal-consistency parameter In Figure 3 (*left*), we show the accuracy of TC-λ models trained on OHSUMED with different values of λ. The setting $\lambda = 1$, corresponsing to DCE, is never optimal, but the best setting depends on the prefix length. When optimizing for full-sequence accuracy, we observe empirically (across the four datasets) that performance is maximized with an effective lookahead of 5–50 tokens, corresponding values of λ between 0.8 and 0.98.

Beyond predictive performance We examine the effects of optimizing for temporal consistency more closely. Given predictive distributions $\boldsymbol{p}_t$ and $\boldsymbol{p}_{t+1}$ produced after seeing t and $t+1$ tokens, respectively, we compute the divergence $\mathrm{KL}[\boldsymbol{p}_{t+1} \| \boldsymbol{p}_t]$. Figure 3 (*right*) shows this divergence, averaged of all successive prefixes of all sequences in the test set. This is the quantity that TC-λ explicitly optimizes[3], and unsurprisingly DCE models are significantly less consistent. While we view temporal consistency primarily as a means to improve predictive performance, stable and consistent predictions might be valuable in their own right, for example in decision-making systems [34].

[3] $\mathrm{KL}[\boldsymbol{p}_{t+1} \| \boldsymbol{p}_t] = H[\boldsymbol{p}_{t+1} \| \boldsymbol{p}_t] + \text{cst}$, where the constant is independent of $\boldsymbol{p}_t$.

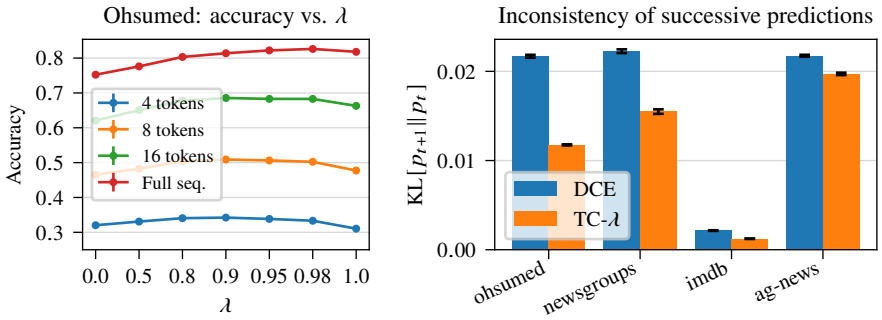

Figure 3: *Left*: Accuracy of OPT-125M classifiers on OHSUMED as a function of λ (mean and 95% CI over 5 runs). *Right*: Average KL-divergence between successive predictive distributions (mean and 95% CI over 10 runs). Lower values correspond to predictive distributions that are more similar across successive time steps.

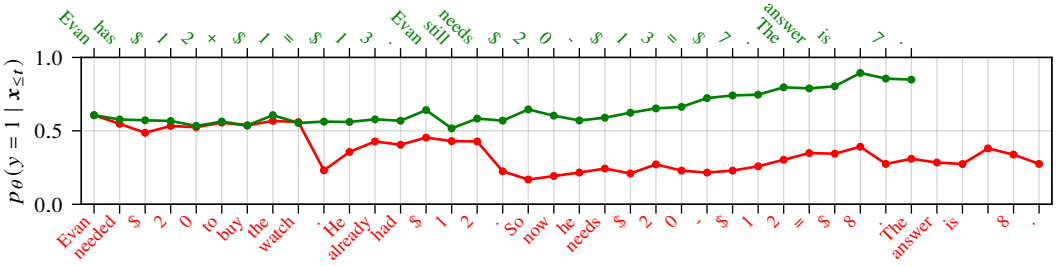

Figure 4: Predicted probability of correctness for two generations of Qwen2.5-0.5B for the prompt *David found $12 on the street. He then gave it to his friend Evan who has $1 and needed to buy a watch worth $20. How much does Evan still need?*

4.2 Language model verification on GSM8K

Next, we consider the problem of using an LLM to solve math word problems. In seminal work, Cobbe et al. [8] propose a simple two-step procedure, consisting of *a*) sampling N generations from the LLM, and *b*) scoring them with a *verifier* model, ultimately selecting the generation with the largest score. This best-of-N approach was shown to significantly increase performance over a single generation, at the expense of a larger computational cost (due to sampling N generations). In this setting, *incremental* verifiers bring substantial benefits: If the verifier is able to accurately distinguish between correct and incorrect generations early on, we can focus computational resources on extending only the most promising generations. This idea has recently received significant attention [47, 39, 21, 41, 31, 20]. In this section, we demonstrate that our TC-λ approach holds promise for learning better incremental LLM verifiers.

We study GSM8K, a dataset of grade-school math problems and their solutions [8]. For our experiments, we use Qwen2.5-0.5B, a pre-trained language model with 0.5 B parameters that is known to perform well on GSM8K for its size [46]. We proceed as follows. For each of the 7473 problems in the training set, we sample 32 generations with temperature 0.7. We label each generation based on whether or not the answer extracted from the generation matches the ground-truth solution. For each problem, we keep exactly one correct and one incorrect generation (sampled uniformly at random), resulting in a balanced dataset. We then train incremental verifiers by fine-tuning Qwen2.5-0.5B with a classification head, as in (10), using DCE and TC-λ (details provided in Appendix B.3). Figure 4 illustrates our setup: Given a problem, our model predicts the probability, token after token, that a generation will eventually produce the correct answer.

Predictive performance For this binary classification task, the ROC AUC metric is particularly relevant, as it reflects the probability that the model correctly ranks a randomly-selected correct generation higher than an incorrect one [15]. Figure 5 (*left*) shows the performance of DCE and TC-λ

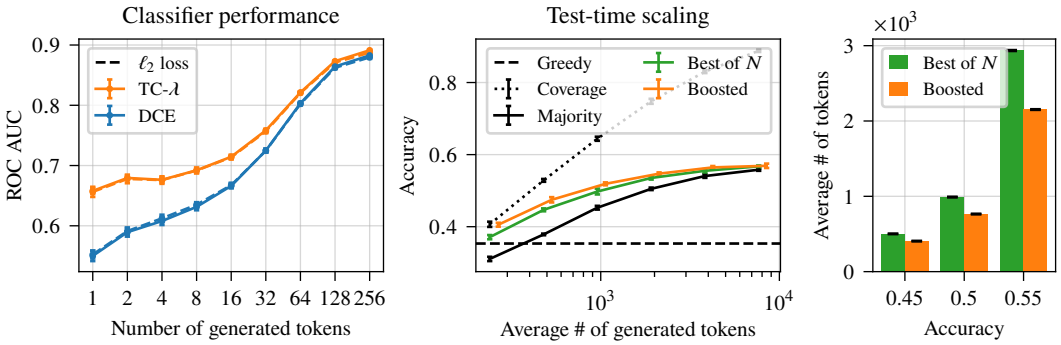

Figure 5: Incremental verification for Qwen2.5-0.5B on GSM8K. *Left*: The TC-λ verifier is better at distinguishing between correct and incorrect generations early on. *Center & right*: A better trade-off between accuracy and compute can be obtained by stopping unpromising generations early on.

as a function of the number of generated tokens. We observe that TC-λ significantly outperforms DCE when the number of tokens is small. With only 8 tokens, TC-λ is almost 70% accurate in distinguishing correct from incorrect generations. Consistent with our findings on the text classification experiments, on this binary classification task, models trained with a squared loss perform almost identically to those trained with a cross-entropy loss in terms of ROC AUC. However, as shown in Figure 8 in the appendix, they produce significantly worse predictive probabilities.

Application to test-time scaling We illustrate the benefit of accurate early correctness predictions in a basic test-time scaling scenario, where we trade off answer accuracy against computational cost, measured by the total number of generated tokens. We propose a simple modification to the best-of-N method, which we call *boosted best-of-N* and which is a simplistic version of the speculative rejection method recently proposed by Sun et al. [36]. Sample 10 tokens for each of $2N$ independent generations, rank them using the TC-λ incremental verifier, and continue sampling the remaining tokens for the top-N generations until completion. Finally, apply the verifier again to the N completed generations and select the best one. Figure 5 (*center & right*) shows that our approach compares favorably to vanilla best-of-N and to majority voting [42]. For a given level of accuracy, the boosted approach requires 23%–33% fewer tokens.

5 Limitations & future work

We have introduced TC-λ, a loss function for training incremental sequence classifiers that draws on insights from TD learning to improve predictive performance. Our empirical results focus on text classification with transformers, but our approach is architecture-agnostic and applicable to any sequence classification problem. Future work could explore its effectiveness on multimodal applications, such as predicting task success from video frames in robotics [11] and games [13].

Given a limited compute budget, we have prioritized small-scale experiments. This has enabled us to run comprehensive hyperparameter sweeps and to report performance averaged over multiple random seeds, increasing our confidence in the results. Our experiments suggest that TC-λ could benefit models at all scales (Figure 2). However, the effectiveness of temporally-consistent methods with models larger than $1.3\,\mathrm{B}$ parameters has not yet been systematically evaluated. Likewise, while our experiments on LLM verification and test-time scaling show promise, further evaluation is needed, particularly in combination with state-of-the-art approaches such as speculative rejection [36].

Finally, a promising direction for future work is applying TC-λ to multi-token prediction [35], which has been shown to improve LLM pre-training [14, 22] and accelerate inference [6]. Importantly, calibrated multi-token predictive distributions should satisfy a temporal-consistency condition that is similar to (4), making this a natural fit for our approach.

References

[1] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[2] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *Proceedings of ICML 2017*, Sydney, Australia, Aug. 2017.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[4] J. M. Bilski and A. Jastrzębska. CALIMERA: A new early time series classification method. *Information Processing & Management*, 60(5), 2023.

[5] S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996.

[6] T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of ICML 2024*, Vienna, Austria, July 2024.

[7] D. Cheikhi and D. Russo. On the statistical benefits of temporal difference learning. In *Proceedings of ICML 2023*, Honolulu, HI, USA, July 2023.

[8] K. Cobbe, V. Kosaraju, et al. Training verifiers to solve math word problems. Preprint, `arXiv:2110.14168 [cs.CL]`, Oct. 2021.

[9] W. Dabney, M. Rowland, M. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of AAAI 2018*, New Orleans, LA, USA, Feb. 2018.

[10] G. M. Del Corso, A. Gulli, and F. Romani. Ranking a stream of news. In *Proceedings of WWW 2005*, Chiba, Japan, May 2005.

[11] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi. Vision-language models as success detectors. In *Proceedings of CoLLAs 2023*, Montreal, QC, Canada, Aug. 2023.

[12] D. Dukić and J. Šnajder. Looking right is sometimes right: Investigating the capabilities of decoder-only LLMs for sequence labeling. In *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand, Aug. 2024.

[13] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar. MineDojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems 35*, New Orleans, LA, USA, Dec. 2022.

[14] F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of ICML 2024*, Vienna, Austria, July 2024.

[15] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

[16] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[17] S. Kadavath, T. Conerly, et al. Language models (mostly) know what they know. Preprint, `arXiv:2207.05221 [cs.CL]`, July 2022.

[18] D. M. Kline and V. L. Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14:310–318, 2005.

[19] K. Lang. NewsWeeder: Learning to filter netnews. In *Proceedings of ICML 1995*, Tahoe City, CA, USA, July 1995.

[20] J. H. Lee, J. Y. Yang, B. Heo, D. Han, and K. M. Yoo. Token-supervised value models for enhancing mathematical reasoning capabilities of large language models. In *Proceedings of ICLR 2025*, Singapore, Apr. 2025.

[21] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. In *Proceedings of ICLR 2024*, Vienna, Austria, May 2024.

[22] A. Liu, B. Feng, et al. DeepSeek-V3 technical report. Preprint, `arXiv:2412.19437 [cs.CL]`, Dec. 2024.

[23] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating Wikipedia by summarizing long sequences. In *Proceedings of ICLR 2018*, Vancouver, BC, Canada, Apr. 2018.

[24] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proceedings of ICLR 2019*, New Orleans, LA, USA, May 2019.

[25] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT 2011*, Portland, Oregon, USA, June 2011.

[26] L. Maystre and D. Russo. Temporally-consistent survival analysis. In *Advances in Neural Information Processing Systems 35*, New Orleans, LA, USA, Dec. 2022.

[27] Y. Meier, J. Xu, O. Atan, and M. Van der Schaar. Predicting grades. *IEEE Transactions on Signal Processing*, 64(4):959–972, 2015.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. Preprint, `arXiv:1312.5602 [cs.LG]`, Dec. 2013.

[29] T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka. Parametric return density estimation for reinforcement learning. In *Proceedings of UAI 2010*, Catalina Island, CA, USA, July 2010.

[30] A. Moschitti and R. Basili. Complex linguistic features for text classification: A comprehensive study. In *Proceedings of ECIR 2004*, Sunderland, UK, Apr. 2004.

[31] S. Mudgal, J. Lee, et al. Controlled decoding from language models. In *Proceedings of ICML 2024*, Vienna, Austria, July 2024.

[32] J. R. Norris. *Markov Chains*. Cambridge University Press, 1997.

[33] A. Radford, J. Wu, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

[34] P. Schäfer and U. Leser. TEASER: Early and accurate time series classification. *Data Mining and Knowledge Discovery*, 34(5):1336–1362, 2020.

[35] M. Stern, N. Shazeer, and J. Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems 31*, Montreal, QC, Canada, Dec. 2018.

[36] H. Sun, M. Haider, R. Zhang, H. Yang, J. Qiu, M. Yin, M. Wang, P. Bartlett, and A. Zanette. Fast best-of-N decoding via speculative rejection. In *Advances in Neural Information Processing Systems 36*, Vancouver, BC, Canada, Dec. 2024.

[37] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3: 9–44, 1988.

[38] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.

[39] J. Uesato, N. Kushman, R. Kumar, F. Song, N. Siegel, L. Wang, A. Creswell, G. Irving, and I. Higgins. Solving math word problems with process- and outcome-based feedback. Preprint, `arXiv:2211.14275 [cs.LG]`, Nov. 2022.

[40] M. Vargas Vieyra and P. Frossard. Deep end-to-end survival analysis with temporal consistency. Preprint, `arXiv:2410.06786 [cs.LG]`, Oct. 2024.

[41] P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. Preprint, arXiv:2312.08935 [cs.AI], Dec. 2023.

[42] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of ICLR 2023*, Kigali, Rwanda, July 2023.

[43] T. Wolf, L. Debut, et al. HuggingFace's transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP 2020: System Demonstrations*, virtual event, Oct. 2020.

[44] Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1):40–48, 2010.

[45] Z. Xing, J. Pei, and P. S. Yu. Early classification on time series. *Knowledge and Information Systems*, 31:105–127, 2012.

[46] A. Yang, B. Yang, et al. Qwen2.5 technical report. Preprint, arXiv:2412.15115 [cs.CL], Dec. 2024.

[47] K. Yang and D. Klein. FUDGE: Controlled text generation with future discriminators. In *Proceedings of NAACL 2021*, virtual event, June 2021.

[48] H. Yèche, A. Pace, G. Rätsch, and R. Kuznetsova. Temporal label smoothing for early event prediction. In *Proceedings of ICML 2023*, Honolulu, HI, USA, July 2023.

[49] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. OPT: Open pre-trained transformer language models. Preprint, arXiv:2205.01068 [cs.CL], May 2022.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and in the introduction are supported by the remainder of the paper, and they respect the guidelines outlined below.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section 3, when developing a theory for our approach, we call out specific assumptions that are likely not satisfied in real-world applications. However, the insights appear to be robust, as supported by the empirical results presented in Section 4. Section 5 discusses additional limitations of our work.

   Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Complete proofs for each of the three propositions presented in Section 3 are provided in Appendix A.2. For space reasons, we were unable to provide proof sketches in the main body, but we made an effort to introduce the theoretical results gradually, in such a way that the reader should have an intuitive understanding of why each result holds.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have attempted to document our experiments thoroughly, providing in the appendix many additional details that we deemed out of scope for the main text, such as specific dataset versions, hyperparameter configurations, the model selection procedure, and more. We intend to complement this information with a comprehensive code release upon publication.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All datasets used in our experiments are publicly available, as are the pretrained language models we build upon (OPT [49] and Qwen2.5 [46]). We intend to release code with instructions to support the reproduction of our main experiments upon publication.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All these details are provided in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All plots in the paper include error bars reflecting 95% confidence intervals across multiple random initializations, computed as $1.96 \times$ standard error. Table 1 omits error measures for clarity, but the results corresponding to the DCE and TC-λ rows are also shown with error bars in Figure 6.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide this information in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS code of ethics, and we believe our work conforms with it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: Our paper proposes a general-purpose training method for improving incremental sequence classification, without targeting any specific application domain. While we acknowledge that downstream applications of such models may carry societal implications, we do not identify any impacts (positive or negative) that are specific to the contributions of this work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This work does not involve the release of models or datasets that present a high risk of misuse. As such, no specific safeguards have been put in place.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use publicly available pretrained models and benchmark datasets in our experiments. We have cited the original sources in the paper and have made a reasonable effort to ensure that all assets are used in accordance with their licenses and terms of use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not introduce any new assets directly. We plan to release code to support reproducibility and will ensure it is adequately documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

A Additional methodological details & proofs

In this appendix, we provide additional details relating to the material presented in Sections 2 and 3.

A.1 Generalized temporal-consistency condition

The temporal-consistency condition (4) can be extended to k-step transitions as follows. Slightly abusing notation and setting $s_{T+1} \equiv \cdots \equiv s_{T+k} \equiv y$, and $p(y \mid s_t) = \mathbf{1}_{\{y=s_t\}}$ for $t > T$, we have

$$p(y \mid s_t) = \mathbf{E}_{p(s_{t+k}|s_t)}[p(y \mid s_{t+k})],$$

for any y, any t and any k. This follows from the Markov properties (1).

A.2 Theoretical results

This sections provides proofs for Propositions 1, 2, and 3. It also introduces an additional result, Proposition 4, that is the analogue of Proposition 2 for DCE.

For an N-dimensional vector $\boldsymbol{x}$, we denote the L_1-norm as $\|\boldsymbol{x}\|_1 = \sum_{n=1}^{N} |x_i|$. Similarly, for an $N \times M$ matrix $\boldsymbol{X}$, we denote the induced ∞-norm as $\|\boldsymbol{X}\|_\infty = \max_{n=1}^{N} \|\boldsymbol{x}_n\|_1$, where $\boldsymbol{x}_n$ is the nth row of $\boldsymbol{X}$.

Proof of Proposition 1. Denote by $\boldsymbol{a}_m^t$ the mth row of the matrix $\boldsymbol{Q}^t\boldsymbol{R}$. By assumption, we know that there is a $\tau \in \mathbf{N}_{\geq 0}$ and a $\varepsilon > 0$ such that $\|\boldsymbol{a}_m^\tau\|_1 \geq \varepsilon$ for all m. Let $F : \mathbf{R}^{M\times K} \to \mathbf{R}^{M\times K}$ be the linear operator defined by $F(\boldsymbol{P}) = \boldsymbol{Q}\boldsymbol{P} + \boldsymbol{R}$. We will show that $F^{\tau+1}$ is a contraction mapping in the induced ∞-norm, that is,

$$\|F^{\tau+1}(\boldsymbol{P}) - F^{\tau+1}(\boldsymbol{P'})\|_\infty \leq (1-\varepsilon)\|\boldsymbol{P} - \boldsymbol{P'}\|_\infty,$$

for any two $M \times K$ row-stochastic matrices $\boldsymbol{P}, \boldsymbol{P'}$. By the Banach fixed-point theorem, it follows that F admits a unique fixed point $\boldsymbol{P}^\star = \lim_{t\to\infty} F^t(\boldsymbol{P}_0)$ for any initial $\boldsymbol{P}_0$.

By construction, the matrix $\boldsymbol{U} \doteq [\boldsymbol{Q} \quad \boldsymbol{R}]$ is row-stochastic, and thus $\|\boldsymbol{U}\|_\infty = 1$ and $\|\boldsymbol{Q}\|_\infty \leq 1$. It follows that $\boldsymbol{Q}^\tau\boldsymbol{U} = \begin{bmatrix}\boldsymbol{Q}^{\tau+1} & \boldsymbol{Q}^\tau\boldsymbol{R}\end{bmatrix}$ is such that $\|\boldsymbol{Q}^\tau\boldsymbol{U}\| \leq \|\boldsymbol{Q}\|_\infty^\tau \|\boldsymbol{U}\|_\infty \leq 1$. Since every row of $\boldsymbol{Q}^\tau\boldsymbol{R}$ has L_1-norm at least ε, it must be that $\|\boldsymbol{Q}^{\tau+1}\|_\infty \leq 1 - \varepsilon$. It follows that

$$\begin{aligned}\|F^{\tau+1}(\boldsymbol{P}) - F^{\tau+1}(\boldsymbol{P'})\|_\infty = \|\boldsymbol{Q}^{\tau+1}(\boldsymbol{P} - \boldsymbol{P'})\|_\infty &\leq \|\boldsymbol{Q}^{\tau+1}\|_\infty \|\boldsymbol{P} - \boldsymbol{P'}\|_\infty \\ &\leq (1-\varepsilon)\|\boldsymbol{P} - \boldsymbol{P'}\|_\infty. \qquad \square\end{aligned}$$

Proof of Proposition 2. Consider one step of the TC optimization problem (6), where we denote the tabular model by using the $M \times K$ matrix $\boldsymbol{\Theta} = [\boldsymbol{\theta}_m]$. Letting $c_m = \sum_{(s,s')\in\mathcal{T}} \mathbf{1}_{\{s=m\}}$, we have

$$\begin{aligned}\boldsymbol{\Theta}^{(i+1)} &\in \arg\min_{\boldsymbol{\Theta}} \sum_n \ell_{\text{TC}}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{(i)}, \boldsymbol{s}_n, y_n) \\ &= \arg\min_{\boldsymbol{\Theta}} \sum_{(s,y)\in\mathcal{B}} H[\boldsymbol{\delta}_y \| \boldsymbol{\theta}_s] + \sum_{(s,s')\in\mathcal{A}} H[\boldsymbol{\theta}_{s'}^{(i)} \| \boldsymbol{\theta}_s] \\ &= \arg\min_{\boldsymbol{\Theta}} \sum_m c_m \left\{ \sum_k \hat{r}_{mk} H[\boldsymbol{\delta}_k \| \boldsymbol{\theta}_m] + \sum_{m'} \hat{q}_{mm'} H[\boldsymbol{\theta}_{m'}^{(i)} \| \boldsymbol{\theta}_m] \right\} \\ &= \arg\min_{\boldsymbol{\Theta}} \sum_m c_m H[\hat{\boldsymbol{r}}_m + \hat{\boldsymbol{q}}_m^\top \boldsymbol{\Theta}^{(i)} \| \boldsymbol{\theta}_m],\end{aligned}$$

where $\hat{\boldsymbol{r}}_m = [\hat{r}_{mk}] \in \mathbf{R}^K$, and $\hat{\boldsymbol{q}}_m = [\hat{q}_{mm'}] \in \mathbf{R}^M$, and where the last equality uses the linearity of the cross-entropy with respect to the target distribution. It follows that the cross-entropy is minimized if $\boldsymbol{\Theta} = \hat{\boldsymbol{Q}}\boldsymbol{\Theta}^{(i)} + \hat{\boldsymbol{R}}$, which corresponds exactly to (8). $\square$

Proof of Proposition 3. The Markov chain of Figure 1 has exactly two absorbing states, and the absorption probabilities sum up to 1. As such, we can focus on the variance of the estimator for $p_{m1}^\star$. We will construct a Markov reward process whose state-value function $V(m)$ is equivalent to the

absorption probability $p^\star_{m1}$. To this end, we collapse the two absorbing states 0 and 1 into a single terminal state denoted by $\varnothing$. We instantiate the reward distribution as follows. For any transition between two transient states, the reward is always 0. For any transition between a state at layer T and the absorbing state $\varnothing$, the reward is 1 with probability $1/2$ and zero otherwise.

We can now recast our problem by using the terminology of Cheikhi and Russo [7, Sec. 7]. For every transient state m we have

$$V(m) = p^\star_{m1}, \qquad V^{\text{MC}}(m) = \hat{p}^{\text{dir}}_{m1}, \qquad V^{\text{TD}}(m) = \hat{p}^{\text{ind}}_{m1}.$$

We focus on states in the first and second layer. Given the sampling process we have defined in the main text, any given state in the second layer will appear in N/W trajectories in expectation, and the transition between any pair of first and second layer states will appear in N/W^2 trajectories in expectation. It follows that, for any state m in the first layer and any state m' in the second layer, the coupling coefficient and the inverse trajectory pooling coefficient are given by

$$C(m, m') = 1/W, \qquad C(m) = 1/W,$$

respectively. We can then apply Theorem 7.2 in Cheikhi and Russo [7] to obtain the desired result. □

The next proposition relates the DCE loss (2) and the optimization problem (3), introduced in Section 2, to the direct estimator $\hat{\boldsymbol{P}}^{\text{dir}}$ presented in Section 3.

Proposition 4. *Let $p_{\boldsymbol{\theta}}(y = k \mid s_t = m) \doteq \theta_{mk}$. Then, $\hat{\boldsymbol{P}}^{\text{dir}}$ is a solution of the DCE optimization problem* (3)*.*

Proof. Denote the tabular model by using the $M \times K$ matrix $\boldsymbol{\Theta} = [\boldsymbol{\theta}_m]$. Letting $c_m = \sum_{(s,y')\in\mathcal{D}'} \mathbf{1}_{\{s=m\}}$, we have

$$\begin{aligned}
\boldsymbol{\Theta}^\star_{\text{DCE}} &\in \arg\min_{\boldsymbol{\Theta}} \sum_n \ell_{\text{DCE}}(\boldsymbol{\Theta}, \boldsymbol{s}_n, y_n) \\
&= \arg\min_{\boldsymbol{\Theta}} \sum_{(s,y)\in\mathcal{D}'} H[\boldsymbol{\delta}_y \| \boldsymbol{\theta}_s] \\
&= \arg\min_{\boldsymbol{\Theta}} \sum_m c_m \left\{ \sum_k \hat{p}^{\text{dir}}_{mk} H[\boldsymbol{\delta}_k \| \boldsymbol{\theta}_m] \right\} \\
&= \arg\min_{\boldsymbol{\Theta}} \sum_m c_m H[\hat{\boldsymbol{p}}^{\text{dir}}_m \| \boldsymbol{\theta}_m],
\end{aligned}$$

where $\hat{\boldsymbol{p}}^{\text{dir}}_m = [\hat{p}^{\text{dir}}_{mk}] \in \mathbf{R}^K$, and where the last equality uses the linearity of the cross-entropy with respect to the target distribution. It follows that the cross-entropy is minimized if $\boldsymbol{\Theta} = \hat{\boldsymbol{P}}^{\text{dir}}$. □

B Additional details on experimental evaluation

This section provides additional details on the experiments presented in the main paper. In Section B.1, we describe the precise procedure we employ to train and select models. In Section B.2, we provide more details on the text classificiation experiments of Section 4.1. In Section B.3, we provide more details on the verification experiments of Section 4.2.

B.1 Training, model selection & metrics

Algorithm 1 presents one step of the training loop for the TC-λ and DCE approaches. Note that DCE is obtained simply by setting $\lambda = 1$, as explained in Section 2.1. With respect to the iterative optimization procedure (6), we make two minor practical adjustments. First, we update the parameters using stochastic gradient updates. Second, we average the loss over all prefixes of each sequence, instead of summing them. This means that every sequence contributes to the loss equally, irrespective of its length.

We run our experiments on an `a3-highgpu-8g` instance on Google Cloud, with 208 vCPUs, 1872 GB of memory, and 8 NVIDIA H100 GPUs. We ensure that every experiment runs on a single

Algorithm 1 TC-λ incremental classifier: single training step

Require: minibatch $\mathcal{B}$, parameters $\boldsymbol{\theta}$, temporal-consistency parameter λ, learning rate η

$\ell(\boldsymbol{\theta}) \leftarrow 0$ ▷ Initialize the loss function.
for $(\boldsymbol{x}, y) \in \mathcal{B}$ **do** ▷ Iterate over the minibatch.
  $T \leftarrow \text{length}(\boldsymbol{x})$
  $\boldsymbol{z}_T \leftarrow \boldsymbol{\delta}_y$
  for $t = T-1, \ldots, 1$ **do**
    $\boldsymbol{z}_t \leftarrow \lambda \boldsymbol{z}_{t+1} + (1-\lambda)\boldsymbol{p}_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x}_{\leq t+1})$
  $\ell(\boldsymbol{\theta}) \leftarrow \ell(\boldsymbol{\theta}) + \frac{1}{T}\sum_{t=1}^{T} H[\text{stopgrad}(\boldsymbol{z}_t) \| \boldsymbol{p}_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x}_{\leq t})]$
return $\boldsymbol{\theta} - \frac{\eta}{|\mathcal{B}|}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta})$ ▷ Update the parameters.

GPU. For every dataset, we set the batch size to maximize GPU utilization. We use the AdamW optimizer [24] with a dynamic learning rate. The learning rate starts at zero, increases linearly over a warmup period, then decreases linearly to zero at the end of the optimization process. We vary the following hyperparameters:

- the number of training epochs,
- the maximum learning rate,
- the warmup period (as a ratio of the total number of training steps),
- the weight decay, and
- the temporal-consistency parameter λ (for TC-λ only).

We run experiments on a grid of hyperparameter configurations, and we select the configuration that maximizes the full-sequence predictive accuracy on a small dataset of held-out sequences. Finally, throughout section 4, we report the mean and standard deviation of the performance of the winning hyperparameter configuration on the full test set, across 10 training runs with different random seeds.

We consider three metrics to measure the quality of a model $p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ on held-out data. The average accuracy is the fraction of examples where $\arg\max_y p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ matches the ground-truth label $y^\star$ (higher is better). The average negative log-likelihood (NLL) is the empirical average of $-\log p_{\boldsymbol{\theta}}(y^\star \mid \boldsymbol{x})$ (lower is better). The area under the ROC curve (ROC AUC) captures the model's ability to discriminate between a random positive and negative example; In the multiclass setting, we use the one-vs-rest macro-average version. A higher ROC AUC is better.

B.2 Text classification datasets

We evaluate models on the following four well-known text classification benchmarks. Summary statistics and licensing terms for each dataset are provided in Table 2.

OHSUMED The dataset contains abstracts of publications in medical journals [30]. The task is to categorize each abstract into one of 23 themes, corresponding to sub-categories of cardiovascular diseases. We use the archive `ohsumed-all-docs.tar.gz` available at `https://disi.unitn.it/moschitti/corpora.htm`.

NEWSGROUPS The dataset contains newsgroup documents from 20 different newsgroups [19]. The task is to identify which newsgroup the document comes from. We use the version of the dataset hosted at `https://huggingface.co/datasets/google-research-datasets/newsgroup`.

IMDB The dataset contains movie reviews from the IMDb website [25]. The task consists of identifying the sentiment of the review (positive or negative). We use the version of the dataset hosted at `https://huggingface.co/datasets/stanfordnlp/imdb`.

AG-NEWS The dataset contains short news articles [10]. The task is to identify the topic of each article. We use the version of the dataset hosted at `https://huggingface.co/datasets/fancyzhx/ag_news`.

Table 2: Summary statistics for the text classification datasets. Statistics on the sequence length assume that the text is tokenized with the GPT-2 tokenizer [33].

					Seq. length percentiles		
Dataset	License	N_{train}	N_{test}	K	50th	90th	99th
OHSUMED	CC BY-NC 4.0	11 520	6782	23	270	430	604
NEWSGROUPS	unknown	11 314	7532	20	373	935	4226
IMDB	unknown	25 000	25 000	2	221	584	1159
AG-NEWS	unknown	120 000	7600	4	51	70	122

The NEWSGROUPS, IMDB and AG-NEWS datasets are provided with separate train and test splits, which we reuse as-is. For OHSUMED, we create our own train and test splits, by partitioning the data uniformly at random.

We fine-tune pre-trained models from the OPT family [49], which are made publicly available under the OPT-175B license[4]. Table 3 provides the hyperparameter configurations for the fine-tuned OPT-125M models whose performance we report in Table 1 in the main text. These hyperparameters are found using the process outlined in Section B.1.

Figure 6 presents detailed results for DCE, TC-λ, and the corresponding squared-loss variants, Direct ℓ_2 loss and LSTD(λ), for prefixes of length $2^i, i = 0, \ldots, 9$, across three metrics. We report the mean and the 95% confidence interval of 10 independent seeds, but the standard error is too small to be visible on the plot. On all datasets, the TC-λ models outperform the DCE models on almost all metrics at almost all prefix lengths.

Compute resources Each experiment, consisting of training an evaluating a model, takes 5–90 minutes on a single GPU, depending on the dataset, the size of the model, and the number of epochs. We estimate that the total compute used for all the experiments performed in the paper, including the hyperparameter sweeps, amounts to approximately 2000 GPU hours.

B.2.1 GPT-4o baseline

We present the templates used for prompting GPT-4o in Figure 7. We use the structured outputs API to ensure that the model always returns exactly one valid class label. For cost reasons, we sample of a subset of 200 examples uniformly at random from the test set for each dataset and each prefix length (4, 16, and all tokens), and restrict our evaluation to this subset. Approximately 0.7% of requests trigger the ChatGPT filters (mostly in the IMDB and NEWSGROUPS datasets). We simply omit the corresponding examples from the evaluation.

[4] See: `https://github.com/facebookresearch/metaseq/blob/main/projects/OPT/MODEL_LICENSE.md`.

Table 3: Hyperparameters used to fine-tune the OPT-125M models reported in the paper.

Dataset	Model	Batch size	Epochs	Max. LR	Warmup	Weight decay	λ
ohsumed	Filtering	32	3	2×10^{-4}	0.03	1×10^{-3}	—
	Specialist, 4	64	2	5×10^{-4}	0.10	1×10^{-4}	—
	Specialist, 16	64	2	5×10^{-4}	0.10	1×10^{-3}	—
	Last token	64	4	1×10^{-4}	0.10	1×10^{-4}	—
	DCE	128	2	2×10^{-4}	0.10	1×10^{-4}	—
	TC-λ	128	4	1×10^{-4}	0.10	1×10^{-3}	0.95
	ℓ_2 loss, direct	128	2	2×10^{-4}	0.10	1×10^{-4}	—
	ℓ_2 loss, TD-λ	128	4	1×10^{-4}	0.10	1×10^{-3}	0.95
newsgroups	Filtering	4	4	5×10^{-5}	0.03	0	—
	Specialist, 4	64	2	2×10^{-5}	0.10	1×10^{-4}	—
	Specialist, 16	64	4	5×10^{-5}	0.10	1×10^{-2}	—
	Last token	64	4	5×10^{-5}	0.10	1×10^{-4}	—
	DCE	64	4	1×10^{-4}	0.03	1×10^{-5}	—
	TC-λ	64	4	1×10^{-4}	0.10	1×10^{-3}	0.98
	ℓ_2 loss, direct	64	4	1×10^{-4}	0.03	1×10^{-5}	—
	ℓ_2 loss, TD-λ	64	4	1×10^{-4}	0.10	1×10^{-3}	0.98
imdb	Filtering	8	2	5×10^{-4}	0.03	1×10^{-3}	—
	Specialist, 4	8	2	5×10^{-5}	0.10	1×10^{-4}	—
	Specialist, 16	8	2	2×10^{-5}	0.10	1×10^{-2}	—
	Last token	8	2	2×10^{-5}	0.10	1×10^{-4}	—
	DCE	8	1	2×10^{-5}	0.10	1×10^{-3}	—
	TC-λ	8	2	2×10^{-5}	0.10	1×10^{-3}	0.80
	ℓ_2 loss, direct	8	1	2×10^{-5}	0.10	1×10^{-3}	—
	ℓ_2 loss, TD-λ	8	2	2×10^{-5}	0.10	1×10^{-3}	0.80
ag-news	Filtering	64	2	5×10^{-5}	0.03	0	—
	Specialist, 4	64	2	1×10^{-4}	0.10	1×10^{-2}	—
	Specialist, 16	64	2	1×10^{-4}	0.10	1×10^{-3}	—
	Last token	64	2	1×10^{-4}	0.10	1×10^{-4}	—
	DCE	128	2	1×10^{-4}	0.03	1×10^{-3}	—
	TC-λ	128	2	1×10^{-4}	0.10	1×10^{-2}	0.90
	ℓ_2 loss, direct	128	2	1×10^{-4}	0.03	1×10^{-3}	—
	ℓ_2 loss, TD-λ	128	2	1×10^{-4}	0.10	1×10^{-2}	0.90

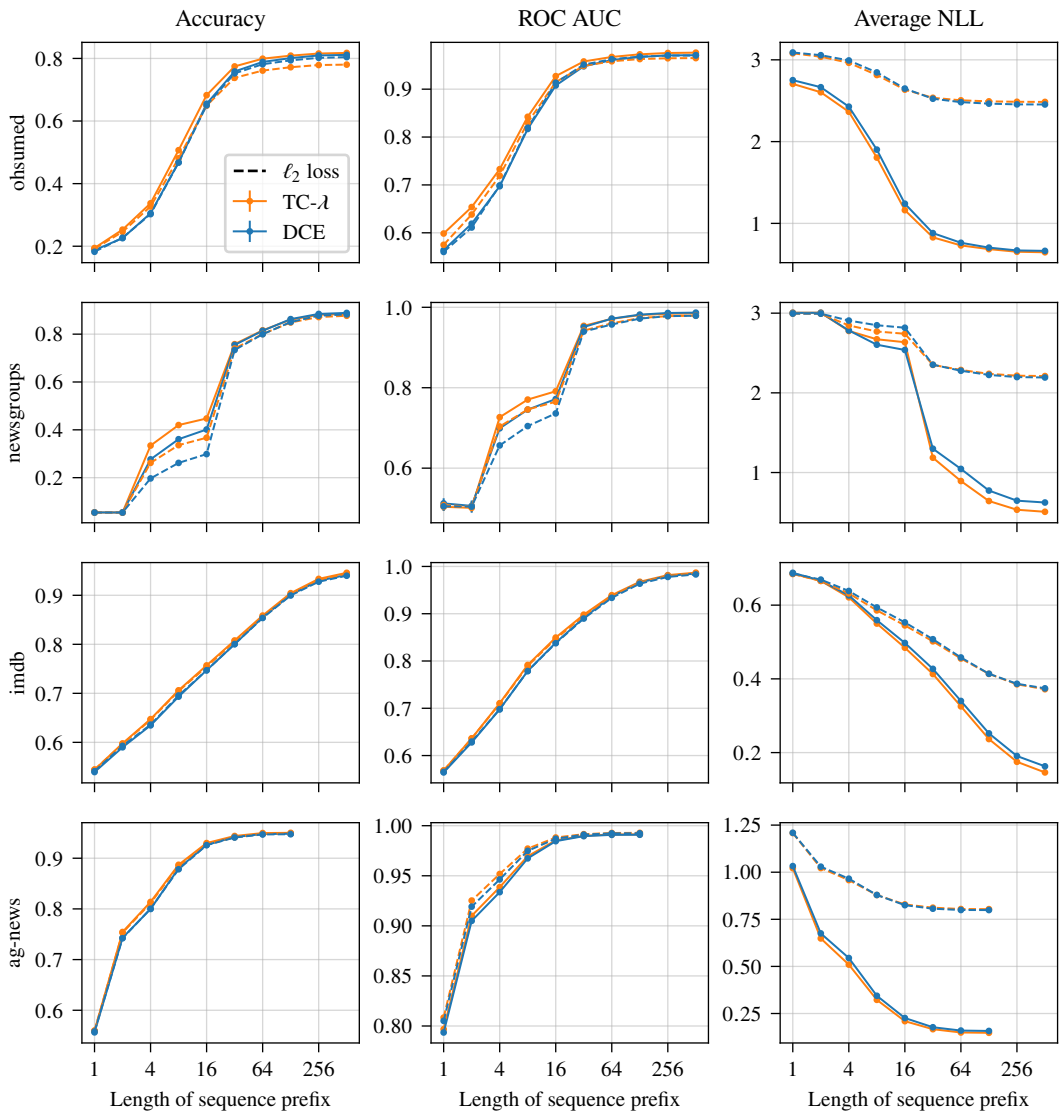

Figure 6: Detailed results for OPT-125M models trained with DCE, TC-λ, and the corresponding squared-loss variants on the text classification datasets. Accuracy and ROC AUC: higher is better. Average NLL: lower is better. We report means and 95% confidence intervals over 10 runs (too small to be visible).

Listing 1: System prompt

```
You are a helpful AI
assistant
specializing in
classifying text. The
 possible class
labels are:

{class_names}

Here are some
examples of text with
 their corresponding
class labels:

{examples}
```

Listing 2: Single example

```
### BEGIN TEXT ###
{text}
### END TEXT ###
label: {label}
```

Listing 3: User prompt

```
What is the label of
the following text?

### BEGIN TEXT ###
{text}
### END TEXT ###
```

Figure 7: Templates used for prompting GPT-4o.

Table 4: Hyperparameters used to fine-tune the Qwen2.5-0.5B models reported in the paper.

Dataset	Model	Batch size	Epochs	Max. LR	Warmup	Weight decay	λ
GSM8K	DCE	48	2	5×10^{-6}	0.03	1×10^{-2}	—
	TC-λ	48	2	5×10^{-6}	0.10	1×10^{-5}	0.95
	Direct ℓ_2 loss	48	2	5×10^{-6}	0.03	1×10^{-2}	—
	LSTD(λ)	48	2	5×10^{-6}	0.10	1×10^{-5}	0.95

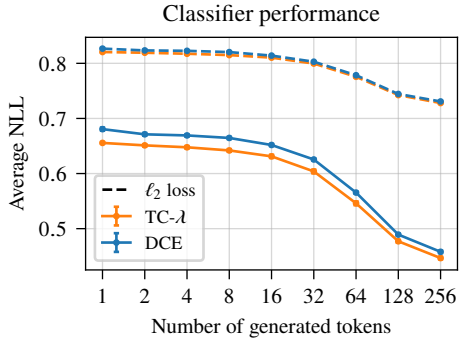

Figure 8: Predictive negative log-likelihood of incremental Qwen2.5-0.5B verifiers on GSM8K (mean and 95% CI over 10 runs).

B.3 Language model verification on GSM8K

For the language model verification experiments, we use the Qwen2.5-0.5B pre-trained language model [46], which is publicly available under the Apache 2.0 license. The GSM8K dataset [8] is publicly available under the MIT license. Hyperparameter selection follows the procedure outlined for the text classification experiments in Section B.1, with one small change: instead of selecting the best hyperparameter configuration based on the full-sequence accuracy, we select it based on the full-sequence ROC AUC. Table 4 provides the hyperparameter configurations for the fine-tuned Qwen2.5-0.5B models whose performance we report in Figure 5 in the main text.

In these experiments, there is one important conceptual difference with respect to the text classification experiments of Section 4.1. The sequence we seek to classify consists of the *generated* tokens only, but the verifier also needs to access the prompt (i.e., the problem statement). We implement this by concatenating the generated response to the prompt. However, when computing the loss, we mask out the terms that correspond to tokens in the prompt.

Predictive NLL Figure 5 (left) in the main text shows that models trained with a squared loss achieve essentially the same ROC AUC as those trained with cross-entropy. Figure 8 further shows that models trained with a cross-entropy loss achieve substantially lower (i.e., better) negative log-likelihood. Thus, when accurate & calibrated predictive uncertainties are important, optimizing the cross-entropy objective performs better in practice.