CONTINUOUS LATENT CONTEXTS ENABLE EFFICIENT ONLINE LEARNING IN TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) exhibit a strong capacity for in-context learning: Given labeled examples, they can generate good predictions without parameter updates. However, many interactive settings go beyond static prediction to online decision-making, in which effective behavior demands adaptation over long multi-turn horizons in response to feedback, and efficient algorithms in these domains must use compact representations of what they have learned. Recently, continuous transformer architectures with latent chain of thought have shown promise for offline iterative tasks such as directed graph-reachability. Motivated by this, we study whether continuous latent context tokens equip transformers to more effectively realize online learning. We give explicit constructions of constant-depth transformers that implement two foundational online decision-making procedures – the weighted majority algorithm and Q-learning – by storing their algorithmic state as linear combinations of feature embeddings, using a small number of latent context tokens. We further train a small GPT-2-style transformer with latent contexts using a multi-curriculum objective that does not directly supervise the latent states. On long synthetic online prediction sequences, this model outperforms comparable discrete-state baselines, while comparisons with prompted LLMs show that textual state summaries can also improve adaptation. Our results suggest that continuous latent contexts provide a simple and effective persistent state for transformers to implement online learning algorithms.

1 INTRODUCTION

Large language models (LLMs), built on the transformer architecture (Vaswani et al., 2017), have demonstrated a striking ability to perform in-context learning: Given a sequence of examples, they can infer a useful predictor without updating their parameters. This ability is especially relevant in interactive settings, where a model receives (often, implicit) feedback over multiple turns and must adapt its future behavior accordingly (Lin et al., 2024b). For example, in a multi-turn conversation, the user's responses provide implicit feedback on which interpretations, plans, or styles are effective. In turn, a helpful conversational agent might use this feedback to improve user satisfaction.

This perspective raises a fundamental question: *Can transformers implement long-horizon online learning algorithms within their context?* In online decision-making, the learner acts sequentially, receives feedback, and records some information to inform future decisions (often conceived as maintaining an algorithmic state). Decisions are often irrevocable, and performance is measured against adaptive or hindsight targets such as the best expert, the best policy, or the optimal value function (Hazan, 2023; Kalai and Vempala, 2005; Cohen and Hazan, 2015). In the case of a conversational agent, online decision-making can support reasoning, whether incorporating user feedback, directing lines of reasoning, or allocating resources, in its broader goal of improving user satisfaction. Therefore, a good online reasoning model should be a good online decision making model (Akyürek et al., 2023), and good online decisions should follow from effective reasoning.

In standard chain-of-thought prompting intermediate reasoning is represented by discrete tokens, which can become very long (Wei et al., 2023). A recent development in latent reasoning is the transformer architecture with continuous chain-of-thought (COCONUT) (Hao et al., 2025), where the continuous states can carry information in "superpositions" of vocabulary tokens that might be lost by discretization. It was recently shown that COCONUT admits a constant-depth transformer to

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

solve the directed graph-reachability problem on an n-vertex graph in $O(D)$ steps (where $D < n$ is the diameter of the graph) (Zhu et al., 2025c), whereas discrete chain-of-thought needs $O(n^2)$ autoregressive steps (Merrill and Sabharwal, 2024)). This is a clear example of how continuous latent representations can serve as a compact representation of context, such as algorithmic state.

Online learning is a natural setting in which to explore similar ideas. For online decision-making, the ability to extract and maintain compactly the information necessary for future decisions is central (Anand and Karmarkar, 2026; Yang et al., 2021; Anand et al., 2025). Classical online algorithms such as weighted majority, online gradient methods, and tabular reinforcement-learning methods all can be abstracted as compressing history into a low-dimensional state and updating that state incrementally as new feedback arrives (Hazan et al., 2017; Kalai and Vempala, 2005; Zinkevich, 2003; Simchowitz et al., 2020; Flaxman et al., 2004; Lin et al., 2023; 2024a). In the weighted majority algorithm this state is a set of expert weights which are updated via multiplicative weights, while in tabular reinforcement learning the state is a value estimate for each state which are updated by various online strategies for dynamic programming, including SARSA and Q-learning. This notion of an online algorithmic state matches the computational role of latent context: to summarize the relevant history, be read by later attention heads, and be updated to incorporate new information.

1.1 Contributions

In this paper, we show that a small number of continuous context tokens suffices to allow transformers to perform well on fundamental online decision-making problems. Specifically:

- We give explicit transformer constructions for multiplicative weights (Littlestone and Warmuth (1994)) in the experts prediction setting, using a continuous latent token to store expert log-weights as a superposition, and tabular Q-learning (Watkins and Dayan (1992)), using continuous context tokens which encode action-indexed columns of the Q-table as a superposition,
- We train transformers with continuous latent contexts using a curriculum strategy and a loss function that is agnostic to the latent contexts, showing that they store the algorithmic state in their latent contexts and track the corresponding classical online-learning algorithms,
- We compare against models that carry their state in discrete space: otherwise identical transformers whose context is replaced by the embedding of a vocabulary token, and LLMs that keep a natural-language note between rounds, Qwen-3-14B (40 layers, 14.8B parameters) (Yang et al., 2025) and DeepSeek-V3 (61 layers, 671B parameters) DeepSeek-AI et al. (2025). The continuous transformers achieve lower regret than both on expert prediction, and track tabular Q-learning more closely than their discrete counterparts.

Taken together, our results support an algorithmic interpretation of continuous latent reasoning: that transformers with continuous latent contexts can use their context as a persistent online-learning state, enabling them to implement online-learning algorithms.

2 Preliminaries

Basic notations. For any integer $N > 0$, we use $[N]$ to denote the set $\{1, 2, \ldots, N\}$. For any finite set $\mathcal{X}$, let $|\mathcal{X}|$ denote the cardinality of $\mathcal{X}$. Let $\mathbb{R}$ be the set of real numbers. We use $\mathbb{1}\{\cdot\}$ to denote the indicator function. We use lower-case bold letters (e.g., $\mathbf{x}, \theta$) and upper-case bold letters (e.g., $\mathbf{W}, \mathbf{U}$) to denote vectors and matrices, respectively. In particular, we use $\mathbf{I}_d$ to denote a $d \times d$ identity matrix, use $\mathbf{0}_{m\times n}$ (or $\mathbf{0}_m$) to denote an $m \times n$ zero matrix (or an m-dimensional zero vector), and use $\mathbf{e}_i$ to denote a one-hot vector of which the i-th entry is one and other entries are all zero, where the dimension of $\mathbf{e}_i$ can be inferred from the context. We also use $\|\cdot\|_\infty$ and $\|\cdot\|_2$ to represent ℓ_∞ and ℓ_2 norm, respectively. For vectors $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$, let $\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^\top \in \mathbb{R}^{m\times n}$ denote their outer product. Finally, for any vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we define the softmax function $\mathsf{SoftMax} : \mathbb{R}^d \to \mathbb{R}^d$ as $\mathsf{SoftMax}(\mathbf{x})_i = \exp(x_i)/(\sum_{j=1}^d \exp(x_j))$.

Tokens and Embeddings. Fix a vocabulary $\mathcal{V} = [V]$. Each token $v \in \mathcal{V}$ has an embedding $\mathbf{u}_v \in \mathbb{R}^d$, whose dimension $d = k\, d_{\text{TE}} + d_{\text{PE}}$ decomposes as k token-embedding blocks and one positional block. For a vector $\mathbf{x} = (x_1, x_2 \cdots, x_d)^\top \in \mathbb{R}^d$, we define $\mathrm{id}(\mathbf{x}) = (x_1, \cdots, x_{d_{\text{TE}}})^\top$, $\mathrm{buf}_j(\mathbf{x}) =$

$(x_{j\cdot d_{\text{TE}}+1}, \cdots, x_{(j+1)\cdot d_{\text{TE}}})^\top$ for $j \in [k]$, and $\text{pos}(\mathbf{x}) = (x_{k\cdot d_{\text{TE}}+1}, \cdots, x_d)$. Let $\tilde{\mathbf{u}}_v = \text{id}(\mathbf{u}_v) \in \mathbb{R}^{d_{\text{TE}}}$ for $v \in \mathcal{V}$. Next, let $\mathbf{U} = [\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \cdots, \tilde{\mathbf{u}}_v] \in \mathbb{R}^{d_{\text{TE}} \times V}$ and assume token orthonormality, $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_V$.

Algorithm 1 Decoder-only Causal Transformer (TF) with Persistent Context Tokens

Require: input embeddings $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_t)$ containing latent slots $\mathbf{c}_1, \ldots, \mathbf{c}_K$ at fixed positions, learned absolute positional encoding parameter θ_{PE}, write map $\mathbf{W}_{\text{ctx}} \in \mathbb{R}^{d \times d}$

$\mathbf{h}_i^{(0)} \leftarrow \mathbf{x}_i + \text{PosEncode}_{\theta_{\mathsf{PE}}}(i)$, for all $i \in [t]$

for $\ell = 0$ to $L-1$ **do**

$h^{\ell+0.5} \leftarrow h^\ell + \sum_{m=1}^{H_\ell} \mathsf{Attn}_{\theta_{\mathsf{Attn}}^{(\ell,m)}}(\mathsf{LN}(h^\ell))$

$h^{\ell+1} \leftarrow h^{\ell+0.5} + \mathsf{MLP}_{\theta_{\mathsf{MLP}}^\ell}(\mathsf{LN}(h^{\ell+0.5}))$

Output: Embedding of the last layer at the last position $\mathsf{TF}_\theta(\mathbf{x}) := \mathsf{LN}(h_t^{(L)})$.

Latent write (at the end of a step): $\mathbf{c}_k \leftarrow \mathbf{c}_k + \mathbf{W}_{\text{ctx}}\, \mathsf{TF}_\theta(\mathbf{x})$ for the slot k selected by the step.

Transformer architectures. An L-layer autoregressive transformer receives a sequence of input embeddings $\mathbf{h} = \mathbf{h}_{[t]} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_t) \in \mathbb{R}^{d \times t}$ and outputs $\mathsf{TF}_\theta(h) \in \mathbb{R}^d$ where $\mathsf{TF}_\theta(\cdot)$ is defined in Algorithm 1. Let $\mathbf{W}_O \in \mathbb{R}^{V \times d}$ be the decoding matrix. A traditional decoder samples the next token $v_{t+1} \sim \mathsf{SoftMax}(\mathbf{W}_O \mathsf{TF}_\theta(h))$ and appends its token embedding in position $t+1$, i.e. $h_{t+1} = u_{v_{t+1}}$, to autoregressively generate subsequent outputs. In our setup, the output at the last token of a step may also be written into a latent slot of the next step's input, $\mathbf{c}_k \leftarrow \mathbf{c}_k + \mathbf{W}_{\text{ctx}} \mathsf{TF}_\theta(h)$. θ_{PE} denotes the learned absolute positional encoding, and each attention layer $\ell \in [L]$ has H_ℓ heads $\{\theta_{\mathsf{Attn}}^{(\ell,m)}\}_{\ell=0,m=0}^{L-1,H_\ell}$.

Algorithm 2 Causal self-attention (Attn)

Require: Input $h = (h_1, \ldots, h_t)$ and $\theta_{\mathsf{Attn}} = (Q, K, V, O)$

Compute the queries $q_i \leftarrow Qh_i$, keys $k_i \leftarrow Kh_i$ and values $v_i \leftarrow Vh_i$, for all $i \in [t]$

for $i = 1, \ldots, t$ **do**

$s_i \leftarrow \mathsf{SoftMax}(\langle q_i, k_1 \rangle, \ldots, \langle q_i, k_i \rangle)$ and $h_i^{\mathsf{Attn}} \leftarrow O \sum_{j=1}^i s_{i,j} v_j$

return $\mathsf{Attn}_{\theta_{\mathsf{Attn}}}(h) = (h_1^{\mathsf{Attn}}, \cdots, h_t^{\mathsf{Attn}})$

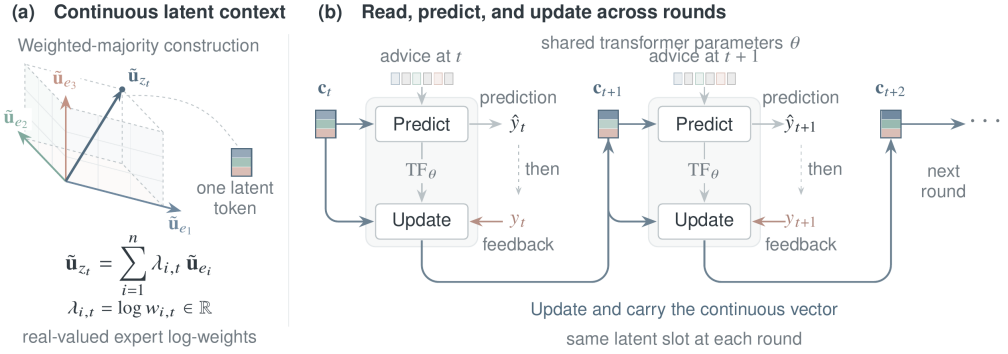


Figure 1: **Continuous latent context as persistent online-learning state. (a)** In the weighted-majority construction, a single latent token represents expert log-weights $\lambda_{i,t} = \log w_{i,t}$ as coefficients in the expert-embedding basis. **(b)** At each round, the transformer reads the current latent state c_t and expert advice to produce a prediction $\hat{y}_t$. After observing the outcome y_t, it updates the state to c_{t+1}, which is carried forward in the same latent slot for the next round. Transformer parameters θ are shared across prediction and update phases and remain fixed throughout the interaction; adaptation occurs through changes to the continuous latent state.

Latent context as online algorithmic state. An online learner can be described by a prediction map G and a state-update map U such that $\widehat{y}_t = G(z_t, x_t)$ and $z_{t+1} = U(z_t, x_t, \xi_t)$ where x_t is the current input, $\widehat{y}_t$ is a prediction or action distribution, and ξ_t contains the subsequently revealed feedback and any realized action needed for the update. The state z_t is sufficient for the algorithm

if its dependence on previous rounds passes entirely through z_t: histories producing the same state induce the same predictions and updates under identical subsequent inputs and feedback.

A transformer with persistent latent context has this interface. Its context tokens $C_t = (c_{1,t}, \ldots, c_{K,t})$ represent the state; the prediction phase defines a map G_θ, and the projected additive latent write defines a map U_θ. Both use parameters θ that remain fixed across rounds. For an encoding Φ of algorithmic states into context tokens, exact implementation means that $G_\theta(\Phi(z), x) = G(z, x)$ and $U_\theta(\Phi(z), x, \xi) = \Phi\big(U(z, x, \xi)\big)$. So, starting from $C_1 = \Phi(z_1)$, these identities ensure that the transformer reproduces the learner's predictions and encoded states at every round. Our constructions establish this correspondence in the attention limits we specify: WMA encodes expert log-weights in one latent token, while tabular Q-learning encodes the columns of its Q-table across $|\mathcal{A}|$ latent tokens.

Storing T interactions requires $\Theta(T)$ memory, and scanning the accumulated history once per round requires $\Theta(T^2)$ cumulative work. If the state size and the costs of evaluating G and U are independent of the round index, the persistent storage is independent of T and total work is $O(T)$. So, under a horizon-independent budget, information retained from earlier rounds must be represented by a bounded-size state. *Online learning* provides concrete algorithms for which such states and their recursive updates are explicit, allowing us to study how continuous latent contexts implement them.

3 Recovering the Weighted Majority Algorithm

We give a constructive implementation of the Weighted Majority algorithm (WMA) (Littlestone and Warmuth, 1994) using a constant-depth causal transformer equipped with latent tokens. Our construction essentially maintains the state of WMA in a single context vector, which consists in a "superposition", or linear combination, of the embeddings of the experts, where the weight of each expert's embedding is precisely the logarithm of its weight. Under softmax attention, this readily allows predictions to be generated as the result of a context vector query against expert embedding keys.

3.1 Problem Formulation

Problem setting. We consider the standard online full-information experts protocol with n experts which all make a binary prediction $p_{i,t} \in \{0, 1\}$ at each round. The learner sees these before making a prediction of her own, at which point the ground truth $y_t \in \{0, 1\}$ is revealed. The learner's objective is to minimize the number of mistakes she makes, often compared against the most reliable expert.

The Weighted Majority Algorithm. The WMA maintains a positive weight for each expert $w_{i,t}$. The prediction of the algorithm at time t depends on the quantity

$$\hat{p}_t = \frac{\sum_{i:\, p_{i,t}=1} w_{i,t}}{\sum_i w_{i,t}} \tag{1}$$

In the randomized version of WMA, the algorithm predicts outcome 1 with probability $\hat{p}_t$; in the deterministic version, by binarizing at a suitable threshold (typically $1/2$). The multiplicative weights update method incorporates feedback: all experts who predicted correctly on that round have their weight increased by a factor of $\gamma = e^\eta > 1$. Crucially, we observe that $\log w_{i,t} = \eta \sum_{s<t} \mathbb{1}\{p_{i,s} = y_s\}$ is proportional to the number of times that expert i was correct up to time t.

Tokens and embeddings. We use the vocabulary $\mathcal{V} = \{e_1, \ldots, e_n, q_0, q_1, \ell_0, \ell_1, p?, w?\}$, for the experts $\{\langle e_1\rangle, \cdots, \langle e_n\rangle\}$, the binary possibilities $\{\langle q_0\rangle, \langle q_1\rangle\}$, the two loss values $\{\langle \ell_0\rangle, \langle \ell_1\rangle\}$ and two special tokens $\{\langle p?\rangle, \langle w?\rangle\}$, with orthonormal identity embeddings $\{\tilde{\mathbf{u}}_v\}_{v\in\mathcal{V}} \subset \mathbb{R}^{d_{\mathrm{TE}}}$. We use a 2-block token embedding with positional extension, $d = 2d_{\mathrm{TE}} + d_{\mathrm{PE}}$, with $\mathbf{u}_v = \big[\mathrm{id}(v) \big| \mathrm{buf}_1(v) \big| \mathrm{pos}(v)\big]^\top$, which contain identity, buffer, and positional subspaces. We write $\langle p_{i,t}\rangle = \langle q_{p_{i,t}}\rangle$, $\langle y_t\rangle = \langle q_{y_t}\rangle$ and $\langle \ell_{i,t}\rangle$ for the token of the loss $\ell_{i,t} = \mathbb{1}\{p_{i,t} \neq y_t\}$.

Input structure. Each round is encoded as a length-$(4n + 4)$ sequence

$$\underbrace{\langle z_t\rangle}_{\text{state slot}} \underbrace{\langle e_1\rangle \langle p_{1,t}\rangle \ldots \langle e_n\rangle \langle p_{n,t}\rangle}_{\text{advice dictionary}} \underbrace{\langle p?\rangle}_{\text{query}} \underbrace{\langle y_t\rangle \langle e_1\rangle \langle \ell_{1,t}\rangle \ldots \langle e_n\rangle \langle \ell_{n,t}\rangle \langle w?\rangle}_{\text{after prediction}}, \tag{2}$$

where $\langle p?\rangle$ and $\langle w?\rangle$ prompt the model for its prediction and updated state. Expert i's tokens always occupy the same positions, so a position identifies an expert. The latent token $\langle z_t\rangle$, initially $\mathbf{0}$, holds $\mathrm{buf}_1(z_t) = \sum_{i=1}^n \log w_{i,t}\, \tilde{\mathbf{u}}_{e_i}$.

3.2 Transformer Construction for the Weighted Majority Algorithm

There are two tasks that the algorithm needs to perform: Given current expert weights and predictions, generate a prediction; then, once the ground truth is revealed, update the expert weights. The weights are encoded in the superposition $\mathbf{z}_t = \mathrm{buf}_1(z_t) = \sum_{i=1}^{n} \lambda_{i,t} \tilde{\mathbf{u}}_{e_i}$, where $e^{\lambda_{i,t}} = w_{i,t}$. Since the expert embeddings are orthonormal, $\mathbf{z}_t \cdot \tilde{\mathbf{u}}_{e_i} = \log w_{i,t}$. To generate a prediction, consider a soft attention head with query $\mathbf{z}_t$, keys $\tilde{\mathbf{u}}_{e_1}, \ldots, \tilde{\mathbf{u}}_{e_n}$, and values $\tilde{\mathbf{u}}_{p_{1,t}}, \ldots, \tilde{\mathbf{u}}_{p_{n,t}}$. The attention head's output is

$$\hat{y}_t = \frac{\sum_{i=1}^{n} e^{\mathbf{z}_t \cdot \tilde{\mathbf{u}}_{e_i}} \tilde{\mathbf{u}}_{p_{i,t}}}{\sum_{i=1}^{n} e^{\mathbf{z}_t \cdot \tilde{\mathbf{u}}_{e_i}}} = \frac{\sum_{i=1}^{n} w_{i,t} \tilde{\mathbf{u}}_{p_{i,t}}}{\sum_{i=1}^{n} w_{i,t}} \Rightarrow \hat{y}_t \cdot \tilde{\mathbf{u}}_{q_1} = \frac{\sum_{i:\, p_{i,t}=1} w_{i,t}}{\sum_{i=1}^{n} w_{i,t}},$$

which is precisely the quantity (1) upon which WMA's prediction is indexed. The update $\lambda_{i,t+1} = \lambda_{i,t} + \eta\, \mathbb{1}\{p_{i,t} = y_t\}$ is linear in each component of the superposition:

$$\mathbf{z}_{t+1} = \mathbf{z}_t + \eta \sum_{i=1}^{n} (1 - \ell_{i,t})\, \tilde{\mathbf{u}}_{e_i}.$$

Theorem 3.1 (Weighted Majority)**.** *For any number of experts n, horizon T, and learning rate $\eta > 0$, there exists a two-layer causal transformer TF_θ (in the hard attention limit $\beta \to \infty$, without LayerNorm) with $O(1)$ attention heads and one ReLU MLP, embedding dimension $d = O(n)$, and a fixed parameter θ independent of T, such that running TF_θ autoregressively on the input structure above outputs the WMA quantity $\hat{p}_t$ at $\langle p? \rangle$ in every round $t \le T$.*

Proof sketch. All heads are causal softmax heads. The routing scores carry a large inverse temperature β, while the exponential weights are computed at unit temperature. The MLP computes exact ReLU conjunctions of $0/1$ inputs (Lemma B.2). The full proof is in Appendix C.2.

Layer 1 attention: Reading the latent. Head 1.1 lets $\langle p? \rangle$ attend to $\langle z_t \rangle$ and receive $\mathbf{z}_t$ in buf_1.

Layer 1 MLP: Correctness indicators. An MLP adds $(1 - \ell_{i,t})\, \tilde{\mathbf{u}}_{e_i}$ into buf_1 of $\langle \ell_{i,t} \rangle$, as the ReLU conjunction of the position of $\langle \ell_{i,t} \rangle$ and the identity ℓ_0.

Layer 2 attention: Prediction and update. Head 2.1 lets $\langle p? \rangle$ attend to the prediction tokens $\langle p_{i,t} \rangle$ with scores $\mathbf{z}_t \cdot \tilde{\mathbf{u}}_{e_i} = \lambda_{i,t}$, where the key maps the position of $\langle p_{i,t} \rangle$ to $\tilde{\mathbf{u}}_{e_i}$, plus a fixed bonus on prediction tokens that excludes all other earlier tokens ($\langle y_t \rangle$ comes later and is masked). As computed above, $\langle p? \rangle$ receives $\hat{p}_t$ along $\tilde{\mathbf{u}}_{q_1}$ in buf_1. Head 2.2 lets $\langle w? \rangle$ attend uniformly to the n loss tokens and receive $\eta \sum_i (1 - \ell_{i,t})\, \tilde{\mathbf{u}}_{e_i}$ in buf_1.

We choose $\mathbf{W}_{\mathrm{ctx}}$ to keep only the expert directions of buf_1, which at $\langle w? \rangle$ hold this increment, so the latent write of Algorithm 1 gives $\mathrm{buf}_1(z_{t+1}) = \mathbf{z}_{t+1}$. The claim follows by induction on t.

Discussion 3.2 (Randomized and deterministic WMA)**.** Thresholding $\hat{p}_t$ at $1/2$ recovers the deterministic WMA, and predicting outcome 1 with probability $\hat{p}_t$ recovers the randomized WMA.

The preceding construction implements WMA exactly in the hard-attention limit. At finite routing inverse temperature β, attention assigns some mass to unintended positions, introducing errors that can propagate through the recurrent latent updates. We now show that, over any fixed horizon, these errors can be made arbitrarily small by choosing a sufficiently large finite β, without changing the architecture or its causal attention masks.

Lemma 3.3 (Informal: finite-temperature approximation)**.** *For every fixed number of experts n, horizon T, learning rate $\eta > 0$, and tolerance $\varepsilon > 0$, there exists a finite routing inverse temperature β_0 such that the prescribed WMA construction satisfies*

$$\sup_{t \le T} \sup_{h_t} \left| f_{\beta,t}(h_t) - p_t^{\mathrm{WMA}}(h_t) \right| \le \varepsilon$$

for all $\beta \ge \beta_0$, where the supremum ranges over all binary histories available at round t. The construction uses its original causal attention masks.

4 Recovering Online Tabular Q-learning

We prove that a constant-depth transformer equipped with latent tokens can execute online Tabular Q-learning, a foundational off-policy reinforcement learning algorithm for solving Markov Decision Processes (MDPs) (Watkins and Dayan, 1992).

Problem setting. We model the environment as an infinite-horizon discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where $\mathcal{S}/\mathcal{A}$ are finite state/action spaces, $\mathcal{P}(s'|s,a)$ is the transition kernel, and $r_t \in [-R_{\max}, R_{\max}]$ is a bounded, possibly stochastic reward. Let $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denote the action-value function, with discount factor $\gamma \in (0,1)$ and learning rate $\alpha \in (0,1]$. Classical online tabular Q-learning algorithm maintains a tabular estimate Q_t and iteratively applies the Bellman update

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a \in \mathcal{A}} Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right). \tag{3}$$

Tokens, embeddings, and input structure. We specialize the general embedding framework above to the vocabulary $\mathcal{V} = \mathcal{S} \cup \mathcal{A} \cup \{\langle \mathsf{BOS} \rangle, \langle r \rangle, \langle \mathsf{Select} \rangle, \langle Q_{\text{curr}} \rangle, \langle Q_{\text{next}} \rangle, \langle \text{Update} \rangle\}$, with orthonormal identity embeddings $\{\tilde{\mathbf{u}}_v\}_{v \in \mathcal{V}} \subset \mathbb{R}^{d_{\text{TE}}}$. We use a 3-block token embedding with positional extension, $d = 3d_{\text{TE}} + d_{\text{PE}}$, with $\mathbf{u}_v = \left[\text{id}(v) \middle| \text{buf}_1(v) \middle| \text{buf}_2(v) \middle| \text{pos}(v)\right]^\top$, containing identity, buffer, and positional subspaces. Each Q-learning step is encoded as a length-$(3|\mathcal{A}|+9)$ sequence consisting of a context segment (encoding the current Q-table) followed by evaluation and update phases:

$$\langle \mathsf{BOS} \rangle \;\; c_1, \ldots, c_{|\mathcal{A}|} \| \, \langle Q_{\text{curr}} \rangle \, \langle s_t \rangle \, \langle a_t \rangle \, \langle r_t \rangle \, \langle Q_{\text{next}} \rangle \, (\langle s_{t+1} \rangle \, \langle a_i \rangle)_{i=1}^{|\mathcal{A}|} \, \langle \mathsf{Select} \rangle \, \| \, \langle a^* \rangle \, \langle \text{Update} \rangle$$

Each context token c_a, initially $\mathbf{0}$, encodes the a-th column of Q_t via $\text{buf}_2(c_a) = \sum_{s \in \mathcal{S}} Q_t(s,a)\, \tilde{\mathbf{u}}_s$. The reward token carries its value as a continuous input, $\text{buf}_2(r_t) = r_t \tilde{\mathbf{u}}_r$.

4.1 Transformer Construction for the Tabular Q-learning Algorithm

At each step, the model must perform two tasks: evaluate the current Q-values to select an action, and update the Q-table using the observed transition. Since all quantities are stored along orthogonal directions, the Bellman update (4.1) is implemented by writing each term into disjoint subspaces and combining them in the residual stream. Here, we show how a small number of attention heads route and assemble these quantities to realize the update within a fixed-depth transformer.

Theorem 4.1 (Tabular Q-learning)**.** *For any finite MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, horizon T, learning rate $\alpha \in (0,1]$, and discount $\gamma \in (0,1)$, there exists a two-layer causal transformer TF_θ (in the hard attention limit $\beta \to \infty$, without LayerNorm) with $O(1)$ attention heads and one ReLU MLP per layer, embedding dimension $d = O(|\mathcal{S}| + |\mathcal{A}|)$, and a fixed parameter θ independent of T, such that running TF_θ autoregressively on the input structure above produces continuous thought tokens that implement the Q-learning update* (4.1) *at each step t.*

Proof sketch. We use a generalized fixed-offset attention mechanism inspired by Zhu et al. (2025c): $\text{FO}(\mathcal{T}, -\ell)$ routes any token in a set $\mathcal{T}$ to a fixed position ℓ steps earlier (Lemma B.1). Under orthonormal embeddings, this acts as an exact indicator-based router and is used to deterministically focus attention from token to token. Also, we use ReLU MLPs which can multiply a bounded value $|x| \le 2R_{\max}/(1-\gamma)$ by a selector $e \in \{0,1\}$ exactly (Lemma B.2). The full proof is in Appendix D.

Layer 1 attention: Routing and context lookup. This layer stages all operands. Heads 1.1 and 1.2 apply $\text{FO}(\mathcal{A}, -1)$ and $\text{FO}(\mathcal{A}, -2)$, copying into buf_1 of each action token the identities of the two tokens before it. Each action token thereby receives the state before it (s_t for $\langle a_t \rangle$, s_{t+1} for candidates $\langle a_i \rangle$) and its role ($\langle a_t \rangle$ and $\langle a^* \rangle$ are after $\langle Q_{\text{curr}} \rangle$ and $\langle \mathsf{Select} \rangle$, respectively). In Head 1.3, each action token attends to its corresponding context slot c_{a_i} and retrieves $\sum_s Q_t(s,a)\tilde{\mathbf{u}}_s$ into buf_2.

Layer 1 MLP: Q-value evaluation. An MLP computes the inner product between buf_1 and buf_2 of each action token, adding $Q_t(s_t,a_t)\tilde{\mathbf{u}}_{Q_{\text{next}}}$ at $\langle a_t \rangle$ and $Q_t(s_{t+1},a_i)\tilde{\mathbf{u}}_{Q_{\text{next}}}$ at each $\langle a_i \rangle$ into buf_2.

Layer 2 attention: Selection and maximization. This layer selects the greedy action and collects the terms of the Bellman update at $\langle \text{Update} \rangle$. Head 2.1 lets $\langle \mathsf{Select} \rangle$ and $\langle \text{Update} \rangle$ attend to the candidates $\langle a_i \rangle$ with scores $\beta_{\max} Q_t(s_{t+1}, a_i)$. As $\beta_{\max} \to \infty$, it attends only to the greedy action a^*, so $\langle \mathsf{Select} \rangle$ receives $\tilde{\mathbf{u}}_{a^*}$ in buf_1 and $\langle \text{Update} \rangle$ receives $\max_a Q_t(s_{t+1}, a)$ along $\tilde{\mathbf{u}}_{Q_{\text{next}}}$ in buf_2. Head 2.2 lets $\langle \text{Update} \rangle$ attend to $\langle a_t \rangle$ and receive $\tilde{\mathbf{u}}_{s_t}$ in buf_1 and $Q_t(s_t,a_t)$ along $\tilde{\mathbf{u}}_{Q_{\text{curr}}}$ in buf_2. Head 2.3 lets $\langle \text{Update} \rangle$ attend to $\langle r_t \rangle$ and receive r_t along $\tilde{\mathbf{u}}_r$ in buf_2.

Layer 2 MLP: Bellman update. This MLP assembles the temporal-difference update at $\langle \text{Update} \rangle$ with the same gate, writing $\alpha\big(r_t + \gamma \max_a Q_t(s_{t+1},a) - Q_t(s_t,a_t)\big)\tilde{\mathbf{u}}_{s_t}$ into buf_2.

We choose $\mathbf{W}_{\text{ctx}}$ to only keep the state directions of buf_2, which at $\langle \text{Update} \rangle$ hold the Bellman update, so the context slot of c_{a_t} becomes the a_t-th column of Q_{t+1}. The claim follows by induction on t.

Behavior policy and Bellman target. The construction implements the Q-learning update for any supplied behavior actions a_t. In particular, a_t may be sampled from an ϵ-greedy or Boltzmann behavior policy. Exploration changes how a_t is chosen, while the target in retains the hard maximum $\max_a Q_t(s_{t+1}, a)$. These two operations must be distinguished: softening Head 2.1 also replaces its value output by a Boltzmann-weighted average, and therefore changes the Bellman update. If both operations are implemented by attention, separate selection and target heads can use different temperatures. A Boltzmann policy requires sampling from its action probabilities, while taking their argmax remains greedy.

5 Experiments

Our theoretical results show that continuous latent contexts allow constant-depth transformers to implement online learning algorithms by storing and updating superposition states. We complement the theory with experiments that test whether such algorithms can be learned by small transformers, whether the learned latent contexts encode the intended algorithmic state, and whether this depends on the latent being continuous.

In each setting, we compare two transformers with the same architecture and data that differ only in the nature of their persistent contexts: the *continuous transformer* adds its encoded output to a latent context as in Algorithm 1, while the *discrete transformer* has context consisting of tokens from the vocabulary, as in traditional chain-of-thought. Section 5.1 studies expert prediction, Section 5.2 online tabular Q-learning, and Section 5.3 frontier LLMs. Details and additional results are in Appendix F[1].

5.1 Unsupervised Learning From Expert Data

We train a GPT-2 style decoder (Radford et al., 2019) (2 layers, $d_{\text{model}} = 64$) that reads one round at a time, with a single latent token $\langle z_t \rangle$ as in Theorem 3.1. Each sequence has four experts whose qualities are drawn uniformly from $[0.3, 0.9]$, and each expert predicts a random binary label correctly with probability equal to its quality. The model is trained, unsupervised on the context, to predict the label ($\mathcal{L} := \mathrm{BCE}(\text{predicted decision}, \text{true label})$) under a curriculum over sequence lengths of up to 200 rounds.

The continuous transformer learns to make online decisions in this setting. Its performance generalizes well beyond the training horizon, and its cumulative regret is below that of multiplicative weights (MW) (Figure 2). Additionally, the continuous transformer effectively uses its context by storing the MW algorithm's state as a superposition. We use a linear probe to decode each expert's cumulative loss from the latent context token. The decoded state accumulates each round's losses without forgetting (retention $\rho = 1.000$ when fitted as $s_t = \rho\, s_{t-1} + k\, \ell_t$), as the residual write of Theorem 3.1 does. This is consistent with the MW algorithm which depends on every past loss. The discrete transformer, whose latent is a single vocabulary token, is not able to store such a state and falls back to a majority vote, which has suboptimal regret.

Note that the continuous transformer is more aligned with the Bayes learner than MW. These algorithms utilize the same state/statistic, the number of mistakes $L_{i,t}$ of each expert, but vote with it differently – MW weights expert i by $e^{-\eta L_{i,t}}$ whereas the Bayes learner weights it by the log-odds of its estimated accuracy $\hat{q}_{i,t} = (t - L_{i,t})/(t+1)$, which allows it to invert experts that are worse than chance. The Bayes vote minimizes the training loss, and the model has $> 85\%$ vote agreement across seeds. When the data are generated so that MW is Bayes-optimal (with labels given by one hidden expert and flipped with a fixed probability), the continuous transformer reproduces the predictions of MW (Table 1).

Table 1: Data on which MW with $\eta = \ln 4$ is Bayes-optimal: distance of the predicted probability to that of MW with $\eta = \ln 4$ (rounds 51–200), and log-loss regret at $T = 500$.

	$\lvert p - p^{\mathrm{MW}}_{\ln 4} \rvert$	Regret
Continuous transformer	0.0015	1.51
MW, $\eta = \ln 4$	0	1.39
MW, $\eta = \sqrt{\ln n / T}$	0.066	8.52
Follow the leader	–	2.27

[1]We provide our code here.

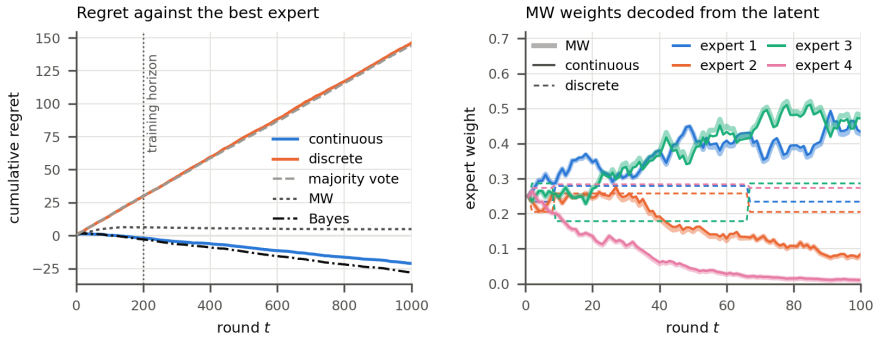


Figure 2: **Expert prediction.** (Left) Long-horizon cumulative regret against the best expert (100 sequences, two seeds per transformer). At $T = 1000$ the regret is -21.1 (continuous), 146.1 (discrete), 4.9 (MW with horizon-tuned η) and -27.9 (Bayes). (Right) MW's weights on a held-out sequence (thick) and the weights computed from the cumulative losses that a linear probe decodes from the contexts of the continuous (thin) and discrete (dashed) transformers. The probe has $R^2 = 1.00$ for the continuous and 0.00 for the discrete latent.

5.2 Imitation Learning of Tabular Q-learning

We train a GPT-2 style decoder (4 layers, 8 heads, $d_{\text{model}} = 256$) on the input structure of Theorem 4.1: one context token per action is prepended to each step, and the output at $\langle \text{Update} \rangle$ is written into the context token of the executed action.

The model is trained to predict the teacher's greedy action, $\mathcal{L} = (1/T) \sum_{t=1}^{T} \text{CE}(\hat{p}_t, a_t^*)$, on 22,000 tabular Q-learning trajectories of 100–200 steps on random MDPs with up to 8 states and 4 actions.

The continuous transformer learns to track Q-learning. To check that the model uses the same discount as the teacher, we compare it to tabular Q-learners with different values of γ run on the same trajectories. The continuous transformer agrees most with $\gamma = 0.9$, the teacher's discount, and least with the myopic $\gamma = 0$ (Figure 4, middle). The discrete transformer agrees poorly with all of them. When given the teacher's trajectory, its chosen action matches the teacher's greedy action on 87% of steps, compared to 46% for the discrete transformer, and this agreement improves over the course of an episode (Figure 4, right). We can also linearly decode the teacher's Q-table from its context tokens (Figure 3).

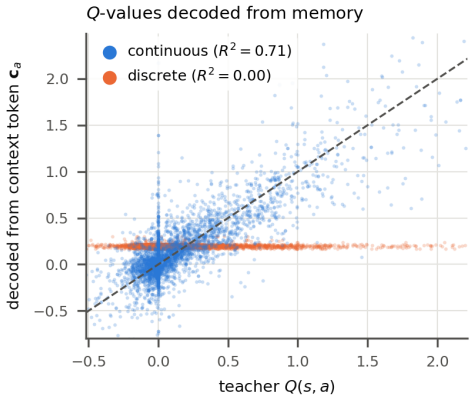


Figure 3: The teacher's Q-values decoded from the context tokens by a linear probe, on held-out episodes. Each context token $\mathbf{c}_a$ is mapped to the column $Q_t(\cdot, a)$.

We also evaluate the models when they act autoregressively. At each step, the model's predicted action is executed (with probability $\epsilon = 0.2$ a random action is taken instead) and fed back as $\langle a^* \rangle$. The continuous transformer reaches 44% of the optimal return, compared to 34% for the discrete transformer, 29% for a random policy and 55% for tabular Q-learning, and it keeps improving long after the training horizon (Figure 4, left). Without exploration, all learners, including tabular Q-learning, perform close to a random policy (27–34%).

5.3 Frontier LLMs using Discrete Contexts

Frontier LLMs are pretrained on large amounts of text that describes online learning, so they may already carry out some form of it in context. We place Qwen-3-14B (Yang et al., 2025) and DeepSeek-V3 (DeepSeek-AI et al., 2025) in the same expert-prediction task: at each round, four experts make a binary prediction, the model makes its own, and the true outcome is then revealed.

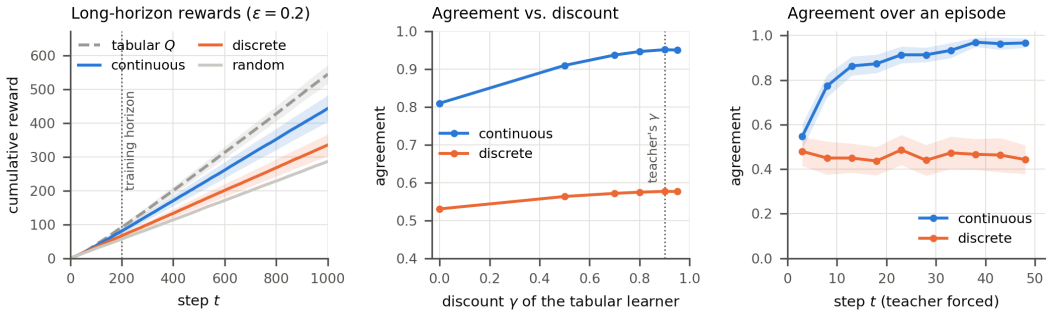


Figure 4: **Tabular Q-learning.** (Left) Cumulative reward when acting in 1000-step MDPs with ϵ-greedy exploration (50 MDPs, mean and 95% interval); the dotted line marks the longest training episode. (Middle) Agreement with tabular Q-learners of discount γ ($\alpha = 0.2$) on the same trajectories. (Right) Agreement with tabular Q-learning's greedy action over an episode under teacher forcing, in windows of 5 steps.

The model is not told to run any particular algorithm. We compare two protocols. In the first, the model only sees the growing interaction history. In the second, it may also write a short, unsupervised note after each round, which is passed back to it in the next round. The note is a visible state that the model chooses for itself, a natural-language analogue of our latent context.

Without a note, the regret of DeepSeek-V3 is close to that of MAJORITY VOTE and FOLLOW PREVIOUS WINNER, which suggests that it relies on the last few rounds rather than on a running estimate of each expert's reliability (Figure 5). With a note, its regret improves substantially, and the notes it writes track how often each expert has been right, much like the state of MW. These notes, however, grow with the number of experts, while our construction keeps the same state in a constant number of continuous latent tokens. Prompts, the Qwen-3-14B results and an analysis of the notes are in Appendix G.

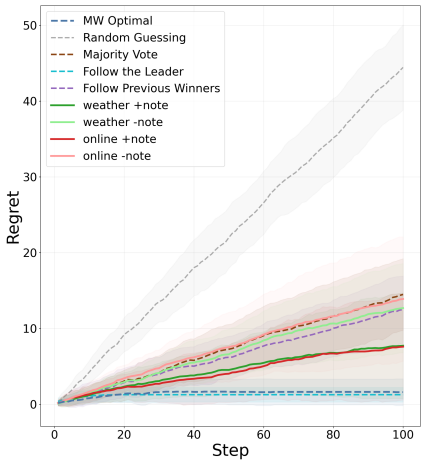


Figure 5: Cumulative regret of DeepSeek-V3 against the best fixed expert on each prefix, with and without a note.

6 CONCLUSION

We study how transformers make online decisions using small continuous latent contexts to maintain algorithmic state. For learning from experts and tabular RL, we construct transformers that implement Weighted Majority and Q-learning by storing and updating expert log-weights and value estimates in latent tokens. Gradient-based training with prediction-based (experts) and imitation-based (RL) losses, without direct latent-state supervision, yields performance comparable to the target algorithms, suggesting that effective recurrent state representations can emerge through training. Experiments with two open-source LLMs, including variants that maintain textual notes, reveal a performance gap to Weighted Majority and motivate further investigation of continuous latent memory for online decision-making. Together, these results connect recurrent latent contexts to the persistent states of online learning algorithms. As persistent adaptation is central to agentic systems, this connection opens a principled direction for studying how transformers acquire, maintain, and update algorithmic state over extended interactions.

AI USE STATEMENT

In this work, we used generative AI tools for literature searches, coding implementation, and to aid in the presentation of our experimental results. We also used AI assistance to edit the manuscript, help design scientific figures, and validate our mathematical claims. The authors take full responsibility for verifying the correctness and originality of all the material in the manuscript, including the theoretical claims, experimental results, code, and figures.

section*Ethics Statement This work studies how transformers can store and update the state of online learning algorithms in continuous latent tokens. All experiments use synthetic data: expert-prediction sequences and randomly generated finite MDPs. No human subjects or personal data are involved. The LLM experiments send the same synthetic expert sequences to a commercial API (DeepSeek-V3) and to a locally served open-weight model (Qwen-3-14B); the prompts contain no personal information. Continuous latent states are not human-readable, so models that carry their state in latent tokens may be harder to monitor than models that reason in text. Linear probes and interventions such as those in Appendix F.1 are one way to inspect such states. Performance on these controlled tasks does not establish safe or reliable adaptation in deployed systems.

REPRODUCIBILITY STATEMENT

Appendices B to D give every weight matrix of the constructions in Theorems 3.1 and 4.1 together with their proofs, including the finite-temperature approximation of Lemma C.8. Appendix E describes the numerical verification of the constructions and the robustness sweeps. Appendix F describes the data generators, model configurations, optimization settings, curricula, evaluation protocols, and hardware for the trained transformers, and Appendix G gives the full LLM prompts, state protocols, decoding settings, and baselines. The anonymized repository linked in Section 5 contains the code for the constructions, training, evaluation, and figures, with instructions for running each experiment.

REFERENCES

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models, 2023.

Ido Amos, Avi Caciularu, Mor Geva, Amir Globerson, Jonathan Herzig, Lior Shani, and Idan Szpektor. Latent Reasoning with Supervised Thinking States. *arXiv preprint arXiv:2602.08332*, 2026. URL https://arxiv.org/abs/2602.08332.

Emile Anand and Ishani Karmarkar. Learning Approximate Nash Equilibria in Cooperative Multi-Agent Reinforcement Learning via Mean-Field Subsampling, 2026.

Emile Anand and Guannan Qu. Efficient Reinforcement Learning for Global Decision Making in the Presence of Local Agents at Scale, 2024.

Emile Timothy Anand and Sarah Liaw. Feel-Good Thompson Sampling for Contextual Bandits: a Markov Chain Monte Carlo Showdown. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Emile Timothy Anand, Ishani Karmarkar, and Guannan Qu. Mean-Field Sampling for Cooperative Multi-Agent Reinforcement Learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Yi Cao, Weijie Shi, Wei-Jie Xu, Yucheng Shen, Yue Cui, Hanghui Guo, Shimin Di, Ziyi Liu, Jiaming Li, Alexander Zhou, Jia Zhu, and Jiajie Xu. TIV: Thought Injection via Vectors for Efficient Reasoning in Large Reasoning Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(36):30148–30155, Mar. 2026. doi: 10.1609/aaai.v40i36.40264.

Bryan Chan, Xinyi Chen, András György, and Dale Schuurmans. Toward Understanding In-context vs. In-weight Learning, 2025. URL https://arxiv.org/abs/2410.23042.

Lijie Chen, Binghui Peng, and Hongxun Wu. Theoretical limitations of multi-layer Transformer, 2024. URL https://arxiv.org/abs/2412.02975.

David Chiang, Peter Cholak, and Anand Pillay. Tighter bounds on the expressivity of transformer encoders. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5544–5562. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/chiang23a.html.

Alon Cohen and Tamir Hazan. Following the perturbed leader for online structured learning. In *International Conference on Machine Learning*, pages 1034–1042. PMLR, 2015.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong

Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. DeepSeek-V3 Technical Report, 2025. URL https://arxiv.org/abs/2412.19437.

Hanwen Du, Yuxin Dong, and Xia Ning. Latent Thinking Optimization: Your Latent Reasoning Language Model Secretly Encodes Reward Signals in Its Latent Thoughts, 2026.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798, 2023.

Abraham D. Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient, 2004.

Deqing Fu, Tianqi CHEN, Robin Jia, and Vatsal Sharan. Transformers Learn Higher-Order Optimization Methods for In-Context Learning: A Study with Linear Models, 2024.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can Looped Transformers Learn to Implement Multi-step Gradient Descent for In-context Learning?, 2024. URL https://arxiv.org/abs/2410.08292.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training Language Models With Pause Tokens, 2024.

Wes Gurnee and Max Tegmark. Language models represent space and time. *arXiv preprint arXiv:2310.02207*, 2023. URL https://arxiv.org/abs/2310.02207.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training Large Language Model to Reason in a Continuous Latent Space, 2025.

Elad Hazan. Introduction to Online Convex Optimization, 2023.

Elad Hazan, Karan Singh, and Cyril Zhang. Learning Linear Dynamical Systems via Spectral Filtering, 2017. URL https://arxiv.org/abs/1711.00946.

Jianhao Huang, Zixuan Wang, and Jason D Lee. Transformers learn to implement multi-step gradient descent with chain of thought. *arXiv preprint arXiv:2502.21212*, 2025.

Francesco Innocenti and El Mehdi Achour. A Simple Generalisation of the Implicit Dynamics of In-Context Learning, 2025. URL https://arxiv.org/abs/2512.11255.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

Jungtaek Kim, Thomas Zeng, Ziqian Lin, Minjae Lee, Chungpa Lee, Jy yong Sohn, Hyung Il Koo, and Kangwook Lee. Transformers in the Dark: Navigating Unknown Search Spaces via Bandit Feedback. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=Jij7zCjVfc. J2C Certification.

Deqian Kong, Dehong Xu, Minglu Zhao, Bo Pang, Jianwen Xie, Andrew Lizarraga, Yuhao Huang, Sirui Xie, and Ying Nian Wu. Latent plan transformer for trajectory abstraction: Planning as latent space inference. *Advances in Neural Information Processing Systems*, 37:123379–123401, 2024.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Hongkang Li, Songtao Lu, Pin-Yu Chen, Xiaodong Cui, and Meng Wang. Training Nonlinear Transformers for Chain-of-Thought Inference: A Theoretical Generalization Analysis. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of Thought Empowers Transformers to Solve Inherently Serial Problems, 2024. URL https://arxiv.org/abs/2402.12875.

Yiheng Lin, James A. Preiss, Emile Anand, Yingying Li, Yisong Yue, and Adam Wierman. Online adaptive policy selection in time-varying systems: No-regret via contractive perturbations. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 53508–53521. Curran Associates, Inc., 2023.

Yiheng Lin, James A. Preiss, Fengze Xie, Emile Anand, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Online policy optimization in unknown nonlinear systems. In Shipra Agrawal and Aaron Roth, editors, *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pages 3475–3522. PMLR, 30 Jun–03 Jul 2024a.

Ying-Chun Lin, Jennifer Neville, Jack Stokes, Longqi Yang, Tara Safavi, Mengting Wan, Scott Counts, Siddharth Suri, Reid Andersen, Xiaofeng Xu, Deepak Gupta, Sujay Kumar Jauhar, Xia Song, Georg Buscher, Saurabh Tiwary, Brent Hecht, and Jaime Teevan. Interpretable user satisfaction estimation for conversational systems with large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11100–11115, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.598.

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. *arXiv preprint arXiv:2407.14207*, 2024.

William Merrill and Ashish Sabharwal. The Expressive Power of Transformers with Chain of Thought, 2024. URL https://arxiv.org/abs/2310.07923.

William Merrill and Ashish Sabharwal. A Little Depth Goes a Long Way: The Expressive Power of Log-Depth Transformers, 2025. URL https://arxiv.org/abs/2503.03961.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's Think Dot by Dot: Hidden Computation in Transformer Language Models, 2024. URL https://arxiv.org/abs/2404.15758.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Keshav Ramji, Tahira Naseem, and Ramón Fernandez Astudillo. Thinking Without Words: Efficient Latent Reasoning with Abstract Chain-of-Thought, 2026.

Vedant Shah, Johan Obando-Ceron, Vineet Jain, Brian Bartoldson, Bhavya Kailkhura, Sarthak Mittal, Glen Berseth, Pablo Samuel Castro, Yoshua Bengio, Nikolay Malkin, Moksh Jain, Siddarth Venkatraman, and Aaron Courville. A Comedy of Estimators: On KL Regularization in RL Training of LLMs, 2026. URL https://arxiv.org/abs/2512.21852.

Max Simchowitz, Karan Singh, and Elad Hazan. Improper learning for non-stochastic control. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 3320–3436. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/simchowitz20a.html.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What Formal Languages Can Transformers Express? A Survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00663.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Archer Wang, Emile Anand, Yilun Du, and Marin Soljačić. Unsupervised Decomposition and Recombination with Discriminator-Driven Diffusion Models, 2026.

Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. Transformers can learn temporal difference methods for in-context reinforcement learning. *arXiv preprint arXiv:2405.13861*, 2024a.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding Language Model Reasoning with Planning Tokens, 2024b.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2023. URL https://arxiv.org/abs/2201.11903.

Jianzhe Wei, Siyu Chen, Jianliang He, and Zhuoran Yang. HOW TRANSFORMERS LEARN CAUSAL STRUCTURES IN-CONTEXT: EXPLAINABLE MECHANISM MEETS THEORETICAL GUARANTEE. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=bpF8zgSt41.

Tengyang Xie, Dylan J. Foster, Akshay Krishnamurthy, Corby Rosset, Ahmed Awadallah, and Alexander Rakhlin. Exploratory Preference Optimization: Harnessing Implicit Q*-Approximation for Sample-Efficient RLHF, 2024. URL https://arxiv.org/abs/2405.21046.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Kunhe Yang, Lin F. Yang, and Simon S. Du. Q-learning with Logarithmic Regret, 2021.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Emergence of superposition: Unveiling the training dynamics of chain of continuous thought. *arXiv preprint arXiv:2509.23365*, 2025a.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025b.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by Superposition: A Theoretical Perspective on Chain of Continuous Thought, 2025c.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.

Limitations and Future Work

We identify several directions of future work. Future theoretical work should determine whether these implementations can emerge from gradient-based training. Since those algorithms are worst-case, optimal distributional information may allow modified algorithms to perform substantially better, so this question is likely to be deep. A related question with both theoretical and empirical angles is the circumstances under which various RL approaches are optimal, with well-known statistical/computational tradeoffs for model-based versus model-free approaches. Here we only considered imitation learning of an observed Q-learner; a substantial extension of this work would be to identify the emergence of Q-learning when training to maximize reward. Finally, we restricted our study to simplified decision-making settings, and it is unclear to what extent our results may generalize to real-world reasoning.

Outline of the Appendix

- Appendix A: Related work
- Appendix B: Table of notation for our theoretical results.
- Appendix C: Theoretical results showing the optimality of the Weighted Majority Algorithm under a particular loss.
- Appendix D: Full Tabular Q-learning construction: Generalized Attention Chooser and main proof
- Appendix E: Empirical validation of our theoretical constructions by implementations.
- Appendix F: Details to our experiments.

A Related Works

Expressiveness of transformers. There is a substantial body of work studying the expressivity of transformers. Most relevant to this paper are those which examine how some form of recurrence can improve expressiveness (most notably in chain-of-thought). For example, Chen et al. (2024) studies low depth transformers for arbitrary problems and shows that for any L there is some problem that a CoT transformer can solve in L layers that a non-CoT transformer will need $L + 1$ layers for. Wei et al. (2023); Feng et al. (2023) shows that constant-depth transformers with CoT can solve certain P-complete problems. Chen et al. (2024) provides constructions for certain constant-depth transformers for each problem in $\mathsf{P} \setminus \mathsf{poly}$ with CoT. There is also work that shows that logarithmically many steps of CoT in input length can expand the upper bound of constant-depth transformer expressiveness from TC^0 to L, while a linear number of steps can further expand the upper bound to NC^1 complete (Merrill and Sabharwal, 2025; Li et al., 2024; Chiang et al., 2023; Strobl et al., 2024; Li et al., 2025). However, these typically focus on discrete CoTs whereas we look at the continuous version.

Reasoning in the context of optimization. Many reasoning tasks have an optimization character, and indeed reasoning itself in LLMs has been been studied as an exploration/exploitation tradeoff (Lin et al., 2024a; Xie et al., 2024; Anand and Liaw, 2025). It has previously been shown that transformers can learn to realize gradient descent (Gatmiry et al., 2024; Huang et al., 2025; Innocenti and Achour, 2025; Kim et al., 2026; Chan et al., 2025; Wei et al., 2026; Laskin et al., 2022; Fu et al., 2024), which are important precursors to this work, where we apply similar ideas to the latent reasoning setting.

Latent reasoning. Prior work shows that LLMs contain structured internal representations, including linear encodings of space and time (Kong et al., 2024; Gurnee and Tegmark, 2023) and compact vectors representing in-context functions (Todd et al., 2024). Recent methods make such hidden computation more explicit through pause (Goyal et al., 2024), planning (Wang et al., 2024b), and filler tokens that enable hidden computation without text (Pfau et al., 2024; Ramji et al., 2026; Liu et al., 2024; Wang et al., 2026; Cao et al., 2026). Here, we study transformers which autoregressively generate their own future inputs (as in chain-of-thought), but without discretizing into discrete tokens. Building models which use this reasoning space effectively is not trivial, since the scaffolding provided by pre-training on natural language does not necessarily benefit. Various approaches have had some success: Amos et al. (2026) proposed an architecture based around continuous tokens produced as a supervised summary of a natural language chain-of-thought. The Huggin architecture contains directly recurrent modules which may iterated arbitrarily for latent reasoning (Geiping et al.,

2025), and these latent reasoning chains have been shown to encode information about predictions (Du et al., 2026). COCONUT allows for fully continuous chains-of-thought (Hao et al., 2025), warm-starting training with a curriculum that first trains reasoning using discrete chains and then relaxes the constraint. Follow-up theory explained its advantage via superposition of multiple reasoning paths (Zhu et al., 2025b;a); the approach we consider here was most directly inspired by COCONUT.

B Notation and Building Blocks

Basic notations. For any integer $N > 0$, we use $[N]$ to denote the set $\{1, 2, \ldots, N\}$. For any finite set $\mathcal{X}$, let $|\mathcal{X}|$ denote the cardinality of $\mathcal{X}$. Let $\mathbb{R}$ be the set of real numbers. We use $\mathbb{1}\{\cdot\}$ to denote the indicator function. We use lower-case bold letters (e.g., $\mathbf{x}, \theta$) and upper-case bold letters (e.g., $\mathbf{W}, \mathbf{U}$) to denote vectors and matrices, respectively. In particular, we use $\mathbf{I}_d$ to denote a $d \times d$ identity matrix, use $\mathbf{0}_{m\times n}$ (or $\mathbf{0}_m$) to denote an $m \times n$ zero matrix (or an m-dimensional zero vector), and use $\mathbf{e}_i$ to denote a one-hot vector of which the i-th entry is one and other entries are all zero, where the dimension of $\mathbf{e}_i$ can be inferred from the context. We also use $\|\cdot\|_\infty$ and $\|\cdot\|_2$ to represent ℓ_∞ and ℓ_2 norm, respectively. For vectors $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$, let $\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^\top \in \mathbb{R}^{m\times n}$ denote their outer product. Finally, for any vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we define the softmax function $\mathsf{SoftMax} : \mathbb{R}^d \to \mathbb{R}^d$ as $\mathsf{SoftMax}(\mathbf{x})_i = \exp(x_i)/(\sum_{j=1}^d \exp(x_j))$.

Tokens and Embeddings. Fix a vocabulary $\mathcal{V} = [V]$. Each token $v \in \mathcal{V}$ has an embedding $\mathbf{u}_v \in \mathbb{R}^d$, whose dimension $d = k\, d_{\text{TE}} + d_{\text{PE}}$ decomposes as k token-embedding blocks (one identity block and $k-1$ buffers) and one positional block. For a vector $\mathbf{x} = (x_1, x_2 \cdots, x_d)^\top \in \mathbb{R}^d$, we define $\mathrm{id}(\mathbf{x}) = (x_1, \cdots, x_{d_{\text{TE}}})^\top$, $\mathrm{buf}_j(\mathbf{x}) = (x_{j\cdot d_{\text{TE}}+1}, \cdots, x_{(j+1)\cdot d_{\text{TE}}})^\top$ for $j \in [k-1]$, and $\mathrm{pos}(\mathbf{x}) = (x_{k\cdot d_{\text{TE}}+1}, \cdots, x_d)$. The multiplicative weights construction uses $k = 2$ and the Q-learning construction $k = 3$. Let $\tilde{\mathbf{u}}_v = \mathrm{id}(\mathbf{u}_v) \in \mathbb{R}^{d_{\text{TE}}}$ for $v \in \mathcal{V}$. Next, let $\mathbf{U} = [\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \cdots, \tilde{\mathbf{u}}_V] \in \mathbb{R}^{d_{\text{TE}}\times V}$ and assume token orthonormality, $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_V$. Latent tokens are not in $\mathcal{V}$: they have $\mathrm{id} = \mathbf{0}$, and only their buffers are written by the latent update of Algorithm 1.

B.1 Token Vocabulary

B.1.1 Multiplicative Weights Setting

A shared discrete vocabulary of size $n + 6$ is used, where n is the number of experts:

Token(s)	Meaning
$\langle e_1\rangle, \ldots, \langle e_n\rangle$	expert tokens
$\langle q_0\rangle, \langle q_1\rangle$	binary outcome tokens; the predictions $\langle p_{i,t}\rangle$ and the label $\langle y_t\rangle$ take these values
$\langle \ell_0\rangle, \langle \ell_1\rangle$	loss tokens; $\langle \ell_{i,t}\rangle$ takes the value $\ell_{i,t} = \mathbb{1}\{p_{i,t} \neq y_t\}$
$\langle p?\rangle$	prediction query token
$\langle w?\rangle$	weight-update query token

The latent token $\langle z_t\rangle$ holds the weight superposition $\mathrm{buf}_1(z_t) = \sum_{i=1}^n \lambda_{i,t}\, \tilde{\mathbf{u}}_{e_i}$ with $\lambda_{i,t} = \log w_{i,t}$.

B.1.2 Tabular Q-learning Setting

A shared discrete vocabulary of size $S_{\max} + A_{\max} + 6$ is used across all episodes:

Token(s)	Meaning
$\langle \mathsf{BOS}\rangle$	sequence start and attention sink, holds value $\mathbf{0}$
$\langle s_1\rangle, \ldots, \langle s_{S_{\max}}\rangle$	state tokens
$\langle a_1\rangle, \ldots, \langle a_{A_{\max}}\rangle$	action tokens
$\langle r\rangle$	reward token; $\langle r_t\rangle$ carries r_t along $\tilde{\mathbf{u}}_r$ in buf_2
$\langle \mathsf{Select}\rangle$	greedy-action query token
$\langle Q_{\text{curr}}\rangle$, $\langle Q_{\text{next}}\rangle$, $\langle \text{Update}\rangle$	Q-learning phase markers

State and action tokens are pre-allocated up to $S_{\max}$ and $A_{\max}$ so that MDPs of varying sizes share a unified vocabulary. The latent tokens $c_1, \ldots, c_{|\mathcal{A}|}$ hold the columns of the Q-table, $\mathrm{buf}_2(c_a) = \sum_{s \in \mathcal{S}} Q_t(s,a)\, \tilde{\mathbf{u}}_s$.

B.2 Generalized Attention Chooser

Both constructions route information between positions with causal softmax heads whose scores are built from the position embeddings. This section states the fixed-offset head $\mathrm{FO}(\mathcal{T}, -\ell)$ used in Theorem 4.1, the mechanism of Zhu et al. (2025c) for orthonormal position embeddings.

Setting. Consider one forward pass over positions $1, \ldots, L$. Position 1 holds $\langle \mathsf{BOS} \rangle$; every other position holds either a token $v_i \in \mathcal{V}$ or a latent slot, which carries no identity. The position embeddings $\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_L \in \mathbb{R}^{d_{\mathrm{PE}}}$ are orthonormal, and throughout the forward pass

$$\mathrm{pos}(\mathbf{h}_i) = \bar{\mathbf{p}}_i, \qquad \mathrm{id}(\mathbf{h}_i) = \tilde{\mathbf{u}}_{v_i} \text{ at a token position}, \qquad \mathrm{id}(\mathbf{h}_i) = \mathbf{0} \text{ at a latent slot},$$

which holds whenever no head or MLP writes into the identity or positional subspace. For $\mathcal{U} \subseteq \mathcal{V}$ let $\tilde{\mathbf{u}}_{\mathcal{U}} = \sum_{v \in \mathcal{U}} \tilde{\mathbf{u}}_v$, so that $\langle \tilde{\mathbf{u}}_{\mathcal{U}}, \tilde{\mathbf{u}}_v \rangle = \mathbb{1}\{v \in \mathcal{U}\}$. We write $\mathbf{x}^{\mathrm{id}}$ and $\mathbf{x}^{\mathrm{pos}}$ for a vector placed in the identity or positional subspace, and $\sigma_{ij} = \langle \mathbf{W}_Q \mathbf{h}_i, \mathbf{W}_K \mathbf{h}_j \rangle$ for the score of query i on key $j \le i$.

Hard attention. A head *attends to* $J \subseteq [i]$ at position i with margin $\Delta > 0$ if $\sigma_{ij} = \max_{j' \le i} \sigma_{ij'}$ for $j \in J$ and $\sigma_{ij} \le \max_{j'} \sigma_{ij'} - \Delta$ for $j \notin J$. The softmax then puts total weight at most $i\, e^{-\Delta}$ outside J and spreads the rest uniformly over J; when Δ is proportional to the inverse temperature β, it attends exactly to J in the limit $\beta \to \infty$.

Lemma B.1 (Fixed-offset head)**.** *Let $\ell \ge 1$, $\mathcal{T} \subseteq \mathcal{V} \setminus \{\mathsf{BOS}\}$, $\beta > 0$, and let the head $\mathrm{FO}(\mathcal{T}, -\ell)$ have query and key maps into $\mathbb{R}^{L+1}$ with basis $\mathbf{e}_0, \ldots, \mathbf{e}_L$,*

$$\mathbf{W}_Q = \beta \sum_{i=\ell+1}^{L} \mathbf{e}_{i-\ell}(\bar{\mathbf{p}}_i^{\mathrm{pos}})^\top + 2\beta\, \mathbf{e}_0(\tilde{\mathbf{u}}_{\mathcal{V}\setminus\mathcal{T}}^{\mathrm{id}})^\top, \qquad \mathbf{W}_K = \sum_{j=1}^{L} \mathbf{e}_j(\bar{\mathbf{p}}_j^{\mathrm{pos}})^\top + \mathbf{e}_0(\tilde{\mathbf{u}}_{\mathsf{BOS}}^{\mathrm{id}})^\top.$$

Then at every position i: (a) if $v_i \in \mathcal{T}$ and $i > \ell$, the head attends to $\{i - \ell\}$ with margin β; (b) if $v_i \notin \mathcal{T}$, it attends to $\{1\}$, the sink $\langle \mathsf{BOS} \rangle$, with margin at least β; (c) at a latent slot, it attends to $\{i - \ell\}$ with margin β if $i > \ell$, and uniformly to $[i]$ if $i \le \ell$.

Proof. By orthonormality of the $\bar{\mathbf{p}}_j$ and the invariants above,

$$\sigma_{ij} = \beta\, \mathbb{1}\{i > \ell,\ j = i - \ell\} + 2\beta\, \langle \tilde{\mathbf{u}}_{\mathcal{V}\setminus\mathcal{T}}, \mathrm{id}(\mathbf{h}_i) \rangle\, \mathbb{1}\{j = 1\},$$

because $\langle \tilde{\mathbf{u}}_{\mathsf{BOS}}, \mathrm{id}(\mathbf{h}_j) \rangle = \mathbb{1}\{j = 1\}$. (a) $\tilde{\mathbf{u}}_{v_i} \perp \tilde{\mathbf{u}}_{\mathcal{V}\setminus\mathcal{T}}$, so the second term vanishes; the scores are β at $i - \ell$ and 0 elsewhere. (b) The second term equals $2\beta\, \mathbb{1}\{j = 1\}$, so $\sigma_{i1} \ge 2\beta$ while $\sigma_{ij} \le \beta$ for $j \ge 2$. (c) $\mathrm{id}(\mathbf{h}_i) = \mathbf{0}$, so only the first term remains; it is identically 0 if $i \le \ell$. □

In words, a token in $\mathcal{T}$ reads the position ℓ steps back, and every other token reads $\langle \mathsf{BOS} \rangle$. A construction that gives $\langle \mathsf{BOS} \rangle$ the value $\mathbf{0}$ thus makes tokens outside $\mathcal{T}$ read nothing. Case (c) covers the latent slots, which carry no identity: they do not fall back to the sink, but read a position ℓ steps back or average over the first ℓ positions, and the constructions ensure that these reads are never used.

B.3 Exact ReLU Gating

Lemma B.2 (Exact ReLU gate)**.** *Let $C > 0$. For all $e \in \{0, 1\}$ and $|x| \le C$,*

$$e\, x = \mathrm{ReLU}(x - C(1 - e)) - \mathrm{ReLU}(-x - C(1 - e)).$$

Hence one ReLU layer with $2n$ neurons computes $\sum_{i=1}^{n} e_i x_i$ exactly for $e \in \{0,1\}^n$ and $x \in [-C, C]^n$.

Proof. If $e = 1$, both offsets vanish and $\mathrm{ReLU}(x) - \mathrm{ReLU}(-x) = x = e\, x$. If $e = 0$, the two arguments are $x - C$ and $-x - C$, both non-positive because $|x| \le C$, so both terms vanish and the output is $0 = e\, x$. For the second claim, the i-th pair of neurons has pre-activations $\pm x_i + C e_i - C$, which are affine in the input (the offset $-C$ is the MLP bias), and the output layer sums the n pair differences, giving $\sum_i e_i x_i$. □

Unlike the softmax heads, the gate is exact at finite parameters; it only requires the a priori bound $|x| \le C$. In the construction, $Q_0 \equiv 0$ and the update (4.1) keep $|Q_t(s,a)| \le R_{\max}/(1-\gamma)$ by induction, and the increment satisfies $|\alpha(r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t))| \le 2R_{\max}/(1-\gamma)$, so $C = 2R_{\max}/(1-\gamma)$ suffices for both MLPs. With a one-hot selector e, the second claim is an exact table lookup x_{i^*}; Layer 1 uses it to read $Q_t(s,a)$ from the retrieved column, and Layer 2 to place the increment on the coordinate of s_t. The role indicators of Layer 1 use the same idea on Boolean inputs: for $a, b, c \in \{0,1\}$, $\mathrm{ReLU}(a+b-1) = a \wedge b$ and $\mathrm{ReLU}(a-b-c) = a \wedge \neg b \wedge \neg c$.

C Theoretical Results for the Weighted Majority Algorithm

C.1 The Population Log-Loss Minimizer Is Bayesian Multiplicative Weights

Population log-loss targets the MWU predictor. In this subsection, we show that under a latent-expert data-generating model, the population log-loss minimizer is the Bayesian posterior predictive mixture, whose weights follow multiplicative weights.

The setup is that $\mathcal{H}_t$ denotes the history available at round t, which is the current expert advice and the past outcomes. Let expert i emit a predictive distribution $p_{i,t}(\cdot) \in \Delta_K$. Let the latent state token encode logits $\lambda_t \in \mathbb{R}^n$, and the normalized expert weights are $w_{t,i} = \mathrm{softmax}(\lambda_t)_i$, and the model is trained by sequence log-loss

$$\mathcal{L}(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} -\log q_{\theta,t}(Y_t|\mathcal{H}_t)\right]. \tag{4}$$

We assume that the data is generated by a latent expert $I^* \sim \pi$ and $Y_t|(I^* = i, \mathcal{H}_t) \sim p_{i,t}(\cdot)$.

Lemma C.1 (Log-loss is uniquely minimized the true conditional)**.** *For a true distribution P and a predicted distribution Q, we have that $\mathbb{E}_{Y\sim P}[-\log Q(Y)] = H(P) + \mathrm{KL}(P\|Q)$.*

Proof. By expanding the LHS, we have

$$\begin{aligned}
\mathbb{E}_{Y\sim P}[-\log Q(Y)] &= -\sum_y P(y)\log Q(y) \\
&= -\sum_y P(y)\log P(y) + \sum_y P(y)\log\frac{P(y)}{Q(y)} \\
&= H(P) + \mathrm{KL}(P\|Q).
\end{aligned}$$

Since the KL divergence is non-negative and 0 if and only if $P = Q$, the minimizer is unique. □

Theorem C.2. *Let $A_t = (p_{1,t}, \ldots, p_{n,t})$ denote the expert advice, $\mathcal{F}_{t-1} = \sigma(A_1, Y_1, \ldots, A_{t-1}, Y_{t-1})$, and $\mathcal{H}_t = \sigma(\mathcal{F}_{t-1}, A_t)$. Assume that $I^* \sim \pi$, that $A_t \perp\!\!\!\perp I^* \mid \mathcal{F}_{t-1}$, and that $\Pr(Y_t = y \mid I^* = i, \mathcal{H}_t) = p_{i,t}(y)$. Define $w_{t,i} = \Pr(I^* = i \mid \mathcal{H}_t)$. Then, almost surely on histories with positive probability,*

$$q_t^*(y \mid \mathcal{H}_t) = \sum_{i=1}^{n} w_{t,i} p_{i,t}(y), \qquad w_{t+1,i} = \frac{w_{t,i} p_{i,t}(Y_t)}{\sum_{j=1}^{n} w_{t,j} p_{j,t}(Y_t)}.$$

The predictor q^* uniquely minimizes the unrestricted population log-loss up to almost-sure equality. If the model class contains q^*, every population minimizer has the same predictive distributions almost surely. With $\ell_{i,t} = -\log p_{i,t}(Y_t)$, the posterior recurrence is multiplicative weights with learning rate one.

Proof. By the tower property, we have that

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \mathbb{E}[\mathbb{E}[-\log q_{\theta,t}(Y_t|\mathcal{H}_t)]\mathcal{H}_t],$$

where for any fixed history h, Lemma C.1 shows that $\mathbb{E}[-\log q_{\theta,t}(Y_t|\mathcal{H}_t)]$ is uniquely minimized by

$$q_{\theta,t}(\cdot|h) = \Pr[Y_t = \cdot|\mathcal{H}_t = h],$$

and hence the unique population minimizer is q^*. Now, we compute the conditional distribution under the latent expert model

$$\Pr[Y_t = y|\mathcal{H}_t] = \sum_{i=1}^{m} \Pr[I^* = i|\mathcal{H}_t] \Pr[Y_t = y|I^* = i, \mathcal{H}_t] = \sum_{i=1}^{n} w_{t,i} p_{i,t}(y),$$

and hence q_t^* is the expert mixture. Next, after observing Y_t, Bayes' rule gives that

$$w_{t+1,i} = \Pr[I^* = i|\mathcal{H}_t, Y_t] = \frac{\Pr[Y_t|I^* = i, \mathcal{H}_t] \Pr[I^* = i|\mathcal{H}_t]}{\sum_j \Pr[Y_t|I^* = j, \mathcal{H}_t] \Pr[I^* = j|\mathcal{H}_t]}.$$

Then,

$$\Pr[Y_t|I^* = i, \mathcal{H}_t] = p_{i,t}(Y_t) \implies w_{t+1,i} = \frac{w_{t,i} p_{i,t}(Y_t)}{\sum_{j=1}^{n} w_{t,j} p_{j,t}(Y_t)}.$$

Finally, writing $p_{i,t}(Y_t) = e^{-\ell_{i,t}}$ recovers the multiplicative weights update. □

Corollary C.3. *Suppose that the transformer with latent context contains the multiplicative weights update predictor. Then, any global minimizer of the population log-loss implements the MWU predictor exactly, and if gradient-based training returns $\hat{\theta}$ with excess population risk at most ϵ such that $\mathcal{L}(\hat{\theta}) - \mathcal{L}(\theta^*) \leq \epsilon$, then*

$$\sum_{t=1}^{T} \mathbb{E}[\mathrm{KL}(q_t^*(\cdot|\mathcal{H}_t) \| q_{\hat{\theta},t}(\cdot|\mathcal{H}_t))] \leq \epsilon. \tag{5}$$

Proof. Since $q_t^*(\cdot|\mathcal{H}_t) = \Pr[Y_t = \cdot|\mathcal{H}_t]$, applying Lemma C.1 gives that

$$-\mathbb{E}[\log q_{\hat{\theta},t}(Y_t|\mathcal{H}_t)|\mathcal{H}_t] - \mathbb{E}[-\log q_t^*(Y_t|\mathcal{H}_t)|\mathcal{H}_t] = \mathrm{KL}(q_t^*(\cdot|\mathcal{H}_t) \| q_{\hat{\theta},t}(\cdot|\mathcal{H}_t)).$$

Therefore, by the tower property,

$$\sum_{t=1}^{T} \mathbb{E}[\mathrm{KL}(q_t^*(\cdot|\mathcal{H}_t) \| q_{\hat{\theta},t}(\cdot|\mathcal{H}_t))] = \mathcal{L}(\hat{\theta}) - \mathcal{L}(\theta^*) \leq \epsilon,$$

which proves the corollary. □

Moreover, the gradient field of the one-step log-loss is the displacement from the current expert mixture to the multiplicative weights posterior. We formalize this below.

Lemma C.4. *Fix one round and define $r_{i,t} \coloneqq p_{i,t}(Y_t)$ and $L_t(\lambda) = -\log \sum_{i=1}^{n} \mathrm{softmax}(\lambda)_i r_{i,t}$. Let $w = \mathrm{softmax}(\lambda)$, and let $w_i^+ = \frac{w_i r_{i,t}}{\sum_j w_j r_{j,t}}$. Then, w^+ is the multiplicative weights update and*

$$\nabla_\lambda L_t(\lambda) = w - w^+. \tag{6}$$

Proof. Let $q(\lambda) = \sum_j w_j r_{j,t}$. For the softmax function, we have that

$$\begin{aligned}
\frac{\partial L_t}{\partial \lambda_i} &= -\frac{1}{q} \frac{\partial q}{\partial \lambda_i} \\
&= -\frac{1}{q} \sum_j r_{j,t} \frac{\partial w_j}{\partial \lambda_i} \\
&= -\frac{1}{q} \sum_j r_{j,t} w_j (\mathbb{1}\{i = j\} - w_i) \\
&= -\frac{1}{q} w_i (r_{i,t} - q) = w_i - \frac{w_i r_{i,t}}{q} = w_i - w_i^+,
\end{aligned}$$

which completes the proof. □

C.2 Proofs for the Weighted Majority Algorithm Construction

We give every weight matrix of the construction behind Theorem 3.1 and prove that it implements the WMA. It is the network verified numerically in Appendix E.

C.2.1 Setup

Round layout. Each round is processed as one sequence of $L = 4n + 4$ positions ($i \in [n]$):

position	1	$2i$	$2i+1$	$2n+2$	$2n+3$	$2n+2i+2$	$2n+2i+3$	$4n+4$
token	$\langle z_t \rangle$	$\langle e_i \rangle$	$\langle p_{i,t} \rangle$	$\langle p? \rangle$	$\langle y_t \rangle$	$\langle e_i \rangle$	$\langle \ell_{i,t} \rangle$	$\langle w? \rangle$

We write v_j for the token at position $j \geq 2$. The prediction and label tokens are $\langle q_0 \rangle$ or $\langle q_1 \rangle$, and $\langle \ell_{i,t} \rangle$ is $\langle \ell_0 \rangle$ or $\langle \ell_1 \rangle$. Position 1 holds the latent, the only information carried between rounds.

Embeddings. We take $d_{\mathrm{TE}} = |\mathcal{V}| = n + 6$, orthonormal $\{\tilde{\mathbf{u}}_v\}_{v \in \mathcal{V}}$, and orthonormal position embeddings $\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_L \in \mathbb{R}^{d_{\mathrm{PE}}}$ with $d_{\mathrm{PE}} = L$, so that

$$d = 2d_{\mathrm{TE}} + d_{\mathrm{PE}} = 6n + 16.$$

For $\mathbf{w} \in \mathbb{R}^{d_{\mathrm{TE}}}$ we write $\mathbf{w}^{\mathrm{id}}, \mathbf{w}^{\mathrm{buf}_1} \in \mathbb{R}^d$ for $\mathbf{w}$ placed in the identity or buffer block and zero elsewhere, and likewise $\bar{\mathbf{p}}_j^{\mathrm{pos}}$. The input of round t is

$$\mathbf{h}_1^{(0)} = \mathbf{z}_t^{\mathrm{buf}_1} + \bar{\mathbf{p}}_1^{\mathrm{pos}}, \qquad \mathbf{h}_j^{(0)} = \tilde{\mathbf{u}}_{v_j}^{\mathrm{id}} + \bar{\mathbf{p}}_j^{\mathrm{pos}} \quad (j \geq 2),$$

where $\mathbf{z}_t = \mathrm{buf}_1(\mathbf{c}_1)$ is the latent at the start of round t. LayerNorm is the identity. We use

$$\tilde{\mathbf{u}}_{\mathcal{V}} = \sum_{v \in \mathcal{V}} \tilde{\mathbf{u}}_v, \qquad \mathbf{\Pi}_{\mathcal{E}} = \sum_{i=1}^{n} \tilde{\mathbf{u}}_{e_i}^{\mathrm{buf}_1} \big(\tilde{\mathbf{u}}_{e_i}^{\mathrm{buf}_1}\big)^\top,$$

$$\mathbf{g}_t = \sum_{i=1}^{n} (1 - \ell_{i,t})\, \tilde{\mathbf{u}}_{e_i}, \qquad \lambda_{i,t} = \eta \sum_{s<t} (1 - \ell_{i,s}) = \log w_{i,t}.$$

$\mathbf{\Pi}_{\mathcal{E}}$ projects onto the expert directions of buf_1, and $\langle \tilde{\mathbf{u}}_{\mathcal{V}}, \tilde{\mathbf{u}}_v \rangle = 1$ for every token v. Query and key vectors live in $\mathbb{R}^{L+1}$ with standard basis $\mathbf{e}_0, \ldots, \mathbf{e}_L$, and $\sigma_{jk} = \langle \mathbf{W}_Q \mathbf{h}_j, \mathbf{W}_K \mathbf{h}_k \rangle$ is the score of query j on key $k \leq j$.

Invariant. The product $\mathbf{W}_O \mathbf{W}_V$ of every head and the output matrix $\mathbf{W}_2$ of the MLP map into buf_1. Hence, at every layer, $\mathrm{id}(\mathbf{h}_j) = \tilde{\mathbf{u}}_{v_j}$ for $j \geq 2$, $\mathrm{id}(\mathbf{h}_1) = \mathbf{0}$, and $\mathrm{pos}(\mathbf{h}_j) = \bar{\mathbf{p}}_j$.

Hard-attention limit. Every score below has the form $\sigma_{jk} = \beta a_{jk} + b_{jk}$ with a_{jk}, b_{jk} independent of β. As $\beta \to \infty$, the attention distribution of query j converges to the softmax of $(b_{jk})_k$ restricted to the maximizers $J_j = \arg\max_{k \leq j} a_{jk}$. For the routing heads $b = 0$, so the head averages its values over J_j (Appendix B.2). For Head 2.1, b carries the log-weights. We compute the network with every head replaced by this limit.

C.2.2 Layer 1

Head 1.1: reading the latent.

$$\mathbf{W}_Q = \beta \sum_{j=1}^{L} \mathbf{e}_j \big(\bar{\mathbf{p}}_j^{\mathrm{pos}}\big)^\top + 2\beta\, \mathbf{e}_0 \big(\tilde{\mathbf{u}}_{p?}^{\mathrm{id}}\big)^\top, \qquad \mathbf{W}_K = \sum_{j=1}^{L} \mathbf{e}_j \big(\bar{\mathbf{p}}_j^{\mathrm{pos}}\big)^\top + \mathbf{e}_0 \big(\bar{\mathbf{p}}_1^{\mathrm{pos}}\big)^\top,$$

$$\mathbf{W}_V = \mathbf{\Pi}_{\mathcal{E}}, \qquad \mathbf{W}_O = \mathbf{I}_d.$$

Layer 1 MLP: correctness indicators. With n hidden neurons and $\mathbf{e}_i$ the standard basis of $\mathbb{R}^n$,

$$\mathbf{W}_1 = \sum_{i=1}^{n} \mathbf{e}_i \big(\tilde{\mathbf{u}}_{\ell_0}^{\mathrm{id}} + \bar{\mathbf{p}}_{2n+2i+3}^{\mathrm{pos}}\big)^\top, \qquad \mathbf{b}_1 = -\sum_{i=1}^{n} \mathbf{e}_i, \qquad \mathbf{W}_2 = \sum_{i=1}^{n} \tilde{\mathbf{u}}_{e_i}^{\mathrm{buf}_1} \mathbf{e}_i^\top.$$

Lemma C.5 (Layer 1)**.** *Let $\mathbf{z}_t \in \mathrm{span}\{\tilde{\mathbf{u}}_{e_1}, \ldots, \tilde{\mathbf{u}}_{e_n}\}$. After Layer 1,*

$$\mathrm{buf}_1\big(\mathbf{h}_{2n+2}^{(1)}\big) = \mathbf{z}_t, \qquad \mathrm{buf}_1\big(\mathbf{h}_{2n+2i+3}^{(1)}\big) = (1 - \ell_{i,t})\, \tilde{\mathbf{u}}_{e_i} \quad (i \in [n]),$$

and $\mathrm{buf}_1(\mathbf{h}_j^{(1)}) = \mathbf{0}$ at every other position $j \geq 2$.

Proof. Head 1.1. By orthonormality of the $\bar{\mathbf{p}}_j$,

$$\sigma_{jk} = \beta\,\mathbb{1}\{k = j\} + 2\beta\,\mathbb{1}\{v_j = p?,\ k = 1\}.$$

So $\langle p? \rangle$ attends to position 1 (score 2β, against β on itself), and every other position attends to itself. The value $\mathbf{\Pi}_{\mathcal{E}}\mathbf{h}_k^{(0)}$ is $\mathbf{z}_t^{\mathrm{buf}_1}$ at $k = 1$ and $\mathbf{0}$ at $k \geq 2$. Hence $\langle p? \rangle$ receives $\mathbf{z}_t$ in buf_1, and the head adds nothing at the other token positions.

MLP. The MLP reads only the identity and position blocks. Neuron i at position j has pre-activation $x + e - 1$, with

$$x = \mathbb{1}\{v_j = \ell_0\} \in \{0, 1\}, \qquad e = \mathbb{1}\{j = 2n + 2i + 3\} \in \{0, 1\}.$$

By Lemma B.2 with $C = 1$, $e\,x = \mathrm{ReLU}(x - (1 - e)) - \mathrm{ReLU}(-x - (1 - e))$, and the second term vanishes because $-x - (1 - e) \leq 0$. So neuron i outputs $\mathrm{ReLU}(x + e - 1) = e\,x$, which is $1 - \ell_{i,t}$ at the loss position of expert i and 0 elsewhere. $\mathbf{W}_2$ writes it along $\tilde{\mathbf{u}}_{e_i}$ in buf_1. □

C.2.3 Layer 2

Head 2.1: exponential weights. With $\mathbf{J} = \sum_{i=1}^{n} \mathbf{e}_i \big(\bar{\mathbf{p}}_{2i+1}^{\mathrm{pos}}\big)^\top$, which maps the position of $\langle p_{i,t} \rangle$ to $\mathbf{e}_i$,

$$\mathbf{W}_Q = 4\beta\,\mathbf{e}_0\big(\tilde{\mathbf{u}}_{\mathcal{V}}^{\mathrm{id}}\big)^\top + \sum_{i=1}^{n} \mathbf{e}_i\big(\tilde{\mathbf{u}}_{e_i}^{\mathrm{buf}_1}\big)^\top, \qquad \mathbf{W}_K = \mathbf{e}_0\big(\tilde{\mathbf{u}}_{q_0}^{\mathrm{id}} + \tilde{\mathbf{u}}_{q_1}^{\mathrm{id}}\big)^\top + \mathbf{J},$$

$$\mathbf{W}_V = \tilde{\mathbf{u}}_{q_1}^{\mathrm{buf}_1}\big(\tilde{\mathbf{u}}_{q_1}^{\mathrm{id}}\big)^\top, \qquad \mathbf{W}_O = \mathbf{I}_d.$$

Head 2.2: the update.

$$\mathbf{W}_Q = 4\beta\,\mathbf{e}_0\big(\tilde{\mathbf{u}}_{w?}^{\mathrm{id}}\big)^\top, \qquad \mathbf{W}_K = \mathbf{e}_0\big(\tilde{\mathbf{u}}_{\ell_0}^{\mathrm{id}} + \tilde{\mathbf{u}}_{\ell_1}^{\mathrm{id}}\big)^\top,$$

$$\mathbf{W}_V = \mathbf{\Pi}_{\mathcal{E}}, \qquad \mathbf{W}_O = \eta n\,\mathbf{I}_d.$$

The Layer 2 MLP is zero.

Lemma C.6 (Layer 2)**.** *Under the hypothesis of Lemma C.5, with $\lambda_i := \langle \mathbf{z}_t, \tilde{\mathbf{u}}_{e_i} \rangle$,*

$$\big\langle \tilde{\mathbf{u}}_{q_1}, \mathrm{buf}_1\big(\mathbf{h}_{2n+2}^{(2)}\big) \big\rangle = \frac{\sum_{i:\ p_{i,t}=1} e^{\lambda_i}}{\sum_{i=1}^{n} e^{\lambda_i}}, \qquad \mathbf{\Pi}_{\mathcal{E}}\,\mathbf{h}_L^{(2)} = \eta\,\mathbf{g}_t^{\mathrm{buf}_1}.$$

Proof. Head 2.1 at $\langle p? \rangle$. By Lemma C.5, the query at $\langle p? \rangle$ has identity $\tilde{\mathbf{u}}_{p?}$ and $\mathrm{buf}_1 = \mathbf{z}_t$, so for keys $k \leq 2n + 2$

$$\sigma_{2n+2,\,k} = 4\beta\,\mathbb{1}\{v_k \in \{q_0, q_1\}\} + \sum_{i=1}^{n} \lambda_i\,\mathbb{1}\{k = 2i + 1\}.$$

The prediction tokens $\langle p_{i,t} \rangle$ at positions $2i + 1$ are exactly the maximizers of the β-part. The label $\langle y_t \rangle$ is also a q token, but it comes later and is masked. The head therefore attends to $\langle p_{i,t} \rangle$ with weight $e^{\lambda_i} / \sum_{i'} e^{\lambda_{i'}}$. The value of $\langle p_{i,t} \rangle$ is $\mathbb{1}\{p_{i,t} = 1\}\,\tilde{\mathbf{u}}_{q_1}^{\mathrm{buf}_1}$, which gives the claimed coordinate. At $\langle p? \rangle$, the residual $\mathbf{z}_t$ and the output of Head 2.2 lie in the expert directions, which are orthogonal to $\tilde{\mathbf{u}}_{q_1}$.

Head 2.2 at $\langle w? \rangle$. The query scores 4β on the n loss tokens, whatever their values, and 0 elsewhere. So the head averages $\mathbf{\Pi}_{\mathcal{E}}\mathbf{h}_k^{(1)}$ over the loss positions and, by Lemma C.5, outputs

$$\eta n \cdot \frac{1}{n} \sum_{i=1}^{n} (1 - \ell_{i,t})\,\tilde{\mathbf{u}}_{e_i}^{\mathrm{buf}_1} = \eta\,\mathbf{g}_t^{\mathrm{buf}_1}.$$

The residual at $\langle w? \rangle$ has $\mathrm{buf}_1 = \mathbf{0}$ (Lemma C.5), and Head 2.1 adds only a multiple of $\tilde{\mathbf{u}}_{q_1}^{\mathrm{buf}_1}$, which $\mathbf{\Pi}_{\mathcal{E}}$ removes. □

C.2.4 Latent write and proof of Theorem 3.1

Let $\mathbf{W}_{\mathrm{ctx}} = \mathbf{\Pi}_{\mathcal{E}}$. The last position of a round is $\langle w? \rangle$, so $\mathsf{TF}_\theta(\mathbf{x}) = \mathbf{h}_L^{(2)}$, and the latent write of Algorithm 1 is

$$\mathbf{c}_1 \leftarrow \mathbf{c}_1 + \mathbf{\Pi}_{\mathcal{E}}\, \mathbf{h}_L^{(2)}.$$

Proof of Theorem 3.1. We show by induction on t that $\mathbf{c}_1 = \mathbf{z}_t^{\mathrm{buf}_1}$ at the start of round t, with

$$\mathbf{z}_t = \sum_{i=1}^{n} \lambda_{i,t}\, \tilde{\mathbf{u}}_{e_i}.$$

At $t = 1$, $\mathbf{c}_1 = \mathbf{0}$ and $\lambda_{i,1} = 0$. Assume the claim at round t. Then $\mathbf{z}_t$ lies in the expert directions, and $\lambda_i = \lambda_{i,t} = \log w_{i,t}$ in Lemma C.6. So the output at $\langle p? \rangle$ is $\hat{p}_t$ of (1), and the write gives

$$\mathbf{c}_1 \leftarrow \big(\mathbf{z}_t + \eta\, \mathbf{g}_t\big)^{\mathrm{buf}_1} = \sum_{i=1}^{n} \big(\lambda_{i,t} + \eta(1 - \ell_{i,t})\big) \tilde{\mathbf{u}}_{e_i}^{\mathrm{buf}_1} = \mathbf{z}_{t+1}^{\mathrm{buf}_1}.$$

By causality, $\mathbf{h}_{2n+2}^{(2)}$ depends only on positions $1, \ldots, 2n+2$, so the prediction is formed before y_t is revealed. The network has two layers, three heads, one ReLU MLP with n neurons and $d = 6n + 16 = O(n)$, and no parameter depends on T. □

Remark C.7 (Finite β)**.** At finite β, the routing heads put weight $O(Le^{-\beta})$ outside their targets. Heads 1.1 and 2.2 then leak small multiples of $\mathbf{z}_t$ into the write, so the network runs WMA with slightly perturbed learning rates, and the errors compound over rounds. In Head 2.1, the positions outside the prediction tokens receive total weight at most $(n+2)e^{-4\beta}/\sum_i e^{\lambda_{i,t}} \le 3e^{-4\beta}$ for every t, because the log-weights $\lambda_{i,t} \ge 0$ count correct predictions and never decrease. Since $\lambda_{i,t} \le \eta t$, an accuracy ε in $\hat{p}_t$ up to horizon T is reached with $\beta = O(\log(\eta n T/\varepsilon))$.

C.3 Finite Temperature Approximation of the Weighted Majority Algorithm

Lemma C.8 (Uniform finite-temperature approximation of WMA)**.** *Fix $n, T \ge 1$ and $\eta > 0$. For each $\beta > 0$, let θ_β denote the parameters prescribed in Appendix D, evaluated at finite routing inverse temperature β, with identity LayerNorm and the original causal attention masks. For each round t, let $h_t = (\mathbf{p}_1, y_1, \ldots, \mathbf{p}_{t-1}, y_{t-1}, \mathbf{p}_t)$ be an arbitrary binary history, where $\mathbf{p}_s \in \{0,1\}^n$ and $y_s \in \{0,1\}$. Let $f_{\beta,t}(h_t)$ denote the scalar prediction obtained by running the transformer through its own recurrent latent updates from initial latent state zero. Define*

$$c_{i,t} = \sum_{s<t} \mathbf{1}\{p_{i,s} = y_s\}, \qquad p_t^{\mathrm{WMA}}(h_t) = \frac{\sum_{i=1}^n e^{\eta c_{i,t}} p_{i,t}}{\sum_{j=1}^n e^{\eta c_{j,t}}}.$$

Then, for every $\varepsilon > 0$, there exists a finite $\beta_0 = \beta_0(n, T, \eta, \varepsilon) > 0$ such that

$$\max_{1 \le t \le T} \sup_{h_t} \big| f_{\beta,t}(h_t) - p_t^{\mathrm{WMA}}(h_t) \big| \le \varepsilon$$

for all $\beta \ge \beta_0$, where the supremum ranges over all binary histories available at round t.

Proof. We derive the exact finite-temperature latent update and prediction, then bound their deviation from the weighted majority algorithm.

We begin by studying layer 1 at finite temperature. Let $m = 2n+2$, $L = 4n+4$, and $\mathcal{E} = \{2n+2i+3 : i \in [n]\}$. Thus m is the prediction-query position, L is the update-query position, and $\mathcal{E}$ is the set of expert-loss positions. The latent-write projection keeps the recurrent state in the expert subspace. We hence write its buffer coordinates at the start of round t as $\mathbf{z}_t^\beta = \sum_{i=1}^n \lambda_{i,t}^\beta \widetilde{\mathbf{u}}_{e_i}$, for $\lambda_{i,1}^\beta = 0$.

Now, let $g_{i,t} = \mathbf{1}\{p_{i,t} = y_t\}$. The scores of Head 1.1 are given by $\sigma_{jk} = \beta \mathbf{1}\{k = j\} + 2\beta \mathbf{1}\{j = m,\ k = 1\}$ with $k \le j$. At the prediction query, the attention weight assigned to the latent position is therefore

$$a_\beta = \frac{e^{2\beta}}{e^{2\beta} + e^\beta + m - 2}.$$

At a position $j \geq 2$ with $j \neq m$, the attention weight assigned to the latent position is $(e^\beta + j - 1)^{-1}$. At position 1, the head reads the latent itself; together with the residual connection, this gives twice the incoming latent vector. For convenience, we define the following schedule:

$$b_{j,\beta} = \begin{cases} 2, & j = 1, \\ a_\beta, & j = m, \\ (e^\beta + j - 1)^{-1}, & j \geq 2,\ j \neq m. \end{cases}$$

The Layer 1 MLP reads only the identity and positional blocks, which remain unchanged at finite β. It therefore computes the correctness indicators exactly, as in Appendix D. Hence, the buffer after Layer 1 satisfies

$$\mathrm{buf}_1\big(\mathbf{h}_{j,t}^{(1),\beta}\big) = b_{j,\beta}\mathbf{z}_t^\beta + \sum_{i=1}^{n} \mathbf{1}\{j = 2n + 2i + 3\} g_{i,t} \widetilde{\mathbf{u}}_{e_i}. \tag{7}$$

Now, we are ready to study the latent recurrence under finite-temperature. At the update query, Head 2.2 assigns score 4β to positions in $\mathcal{E}$ and zero elsewhere. Its attention weights are $\omega_{j,\beta} = e^{4\beta \mathbf{1}\{j \in \mathcal{E}\}}/D_\beta$, where $D_\beta = ne^{4\beta} + L - n$. By (7), the residual contributes $b_{L,\beta}\mathbf{z}_t^\beta$. Head 2.1 is removed by the latent-write projection, and the Layer 2 MLP is zero. Including the external additive latent write and starting from zero therefore gives

$$\mathbf{z}_{t+1}^\beta = \underbrace{\left(1 + b_{L,\beta} + \eta n \sum_{j=1}^{L} \omega_{j,\beta} b_{j,\beta}\right)}_{=:U_\beta} \mathbf{z}_t^\beta + \underbrace{\frac{\eta n e^{4\beta}}{D_\beta}}_{=:V_\beta} \sum_{i=1}^{n} g_{i,t} \widetilde{\mathbf{u}}_{e_i}, \tag{8}$$

Thus, we have that $\lambda_{i,t+1}^\beta = U_\beta \lambda_{i,t}^\beta + V_\beta g_{i,t}$ for $\lambda_{i,1}^\beta = 0$ and $\lambda_{i,t}^\beta = V_\beta \sum_{s<t} U_\beta^{t-1-s} g_{i,s}$.

So, as $\beta \to \infty$, we have $a_\beta \to 1$, $V_\beta \to \eta$, and $U_\beta \to 1$. For the last limit, $b_{L,\beta} \to 0$, while $b_{j,\beta} \to 0$ on $\mathcal{E}$ and $\omega_{j,\beta} \to 0$ outside $\mathcal{E}$, but we point out that all $b_{j,\beta}$ remain bounded.

Therefore, with this, we can characterize an exact finite-temperature prediction. At the prediction query, the Layer 1 buffer is $a_\beta \mathbf{z}_t^\beta$, so Head 2.1 assigns score $4\beta + a_\beta \lambda_{i,t}^\beta$ and scalar value $p_{i,t}$ to expert i's prediction position. The remaining $m - n = n + 2$ positions have score and scalar value zero. Since the residual and Head 2.2 contribute only expert directions,

$$f_{\beta,t}(h_t) = \frac{\sum_{i=1}^{n} e^{a_\beta \lambda_{i,t}^\beta} p_{i,t}}{\sum_{i=1}^{n} e^{a_\beta \lambda_{i,t}^\beta} + (n+2)e^{-4\beta}}. \tag{9}$$

Finally, we are ready to establish the uniform error bound. For $\mathbf{p} \in \{0,1\}^n$, define $F_{\mathbf{p}}(\mathbf{x}) = \sum_i e^{x_i} p_i / \sum_j e^{x_j}$. Writing $\pi_i = e^{x_i}/\sum_j e^{x_j}$ and $F = F_{\mathbf{p}}(\mathbf{x})$, differentiation gives $\partial_i F_{\mathbf{p}} = \pi_i(p_i - F)$. Because the p_i are binary, $\|\nabla F_{\mathbf{p}}(\mathbf{x})\|_1 = 2F(1-F) \leq 1/2$. Thus $F_{\mathbf{p}}$ is $1/2$-Lipschitz with respect to the ℓ_∞ norm.

Set $x_{i,t}^\beta = a_\beta \lambda_{i,t}^\beta$ and $\lambda_{i,t}^{\mathrm{WMA}} = \eta \sum_{s<t} g_{i,s}$, so that $p_t^{\mathrm{WMA}}(h_t) = F_{\mathbf{p}_t}(\boldsymbol{\lambda}_t^{\mathrm{WMA}})$. Also let $Z_{\beta,t} = \sum_i e^{x_{i,t}^\beta}$ and $c_\beta = (n+2)e^{-4\beta}$. The nonnegativity of $a_\beta, U_\beta, V_\beta$ and Equation 8 imply $x_{i,t}^\beta \geq 0$, hence $Z_{\beta,t} \geq n$. Using Equation 9, $0 \leq F_{\mathbf{p}_t} \leq 1$, and the Lipschitz bound, we obtain, for every $t \leq T$ and history h_t, that

$$\begin{aligned} \left|f_{\beta,t}(h_t) - p_t^{\mathrm{WMA}}(h_t)\right| &\leq \left|f_{\beta,t}(h_t) - F_{\mathbf{p}_t}(\mathbf{x}_t^\beta)\right| + \left|F_{\mathbf{p}_t}(\mathbf{x}_t^\beta) - F_{\mathbf{p}_t}(\boldsymbol{\lambda}_t^{\mathrm{WMA}})\right| \\ &\leq \frac{c_\beta}{Z_{\beta,t} + c_\beta} + \frac{1}{2}\|\mathbf{x}_t^\beta - \boldsymbol{\lambda}_t^{\mathrm{WMA}}\|_\infty \\ &\leq \frac{c_\beta}{n} + \frac{1}{2}\max_{i \in [n]} \left|\sum_{s<t} \left(a_\beta V_\beta U_\beta^{t-1-s} - \eta\right) g_{i,s}\right| \\ &\leq \frac{n+2}{n} e^{-4\beta} + \frac{1}{2}\sum_{k=0}^{T-2} \left|a_\beta V_\beta U_\beta^k - \eta\right|. \end{aligned} \tag{10}$$

Here sums over an empty index set are zero, including the final sum when $T = 1$. The last inequality uses $g_{i,s} \in \{0, 1\}$. The bound is independent of t and h_t, so it also bounds $\max_{1 \le t \le T} \sup_{h_t} |f_{\beta,t}(h_t) - p_t^{\mathrm{WMA}}(h_t)|$. For fixed n, T, η, its right-hand side tends to zero because $a_\beta \to 1$, $U_\beta \to 1$, and $V_\beta \to \eta$. Consequently, for every $\varepsilon > 0$, there exists a finite $\beta_0 = \beta_0(n, T, \eta, \varepsilon) > 0$ such that this uniform error is at most ε for all $\beta \ge \beta_0$. □

D Proofs for the Tabular Q-learning Construction

We give every weight matrix of the two-layer transformer of Theorem 4.1 and verify what each position holds after each layer. It is the network verified numerically in Appendix E. Throughout, $n := |\mathcal{A}|$ with $\mathcal{A} = \{a_1, \dots, a_n\}$, $C := 2R_{\max}/(1-\gamma)$ is the gate constant of Lemma B.2, and LayerNorm is the identity.

D.1 Setup

Step layout. A step occupies $L = 3n + 9$ positions ($j, i \in [n]$):

Position	Content	Input embedding
1	$\langle \mathsf{BOS} \rangle$	$\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathsf{BOS}}$
$j+1$	context slot c_{a_j}	$\mathbf{c}_{a_j} \in \mathrm{span}\{\tilde{\mathbf{u}}^{\mathrm{buf}_2}_s : s \in \mathcal{S}\}$
$n+2, n+3, n+4$	$\langle Q_{\mathrm{curr}} \rangle$, $\langle s_t \rangle$, $\langle a_t \rangle$	$\tilde{\mathbf{u}}^{\mathrm{id}}_{Q_{\mathrm{curr}}}$, $\tilde{\mathbf{u}}^{\mathrm{id}}_{s_t}$, $\tilde{\mathbf{u}}^{\mathrm{id}}_{a_t}$
$n+5$	$\langle r_t \rangle$	$\tilde{\mathbf{u}}^{\mathrm{id}}_r + r_t\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_r$
$n+6$	$\langle Q_{\mathrm{next}} \rangle$	$\tilde{\mathbf{u}}^{\mathrm{id}}_{Q_{\mathrm{next}}}$
$n+5+2i, n+6+2i$	$\langle s_{t+1} \rangle$, candidate $\langle a_i \rangle$	$\tilde{\mathbf{u}}^{\mathrm{id}}_{s_{t+1}}$, $\tilde{\mathbf{u}}^{\mathrm{id}}_{a_i}$
$3n+7, 3n+8, 3n+9$	$\langle \mathsf{Select} \rangle$, $\langle a^* \rangle$, $\langle \mathrm{Update} \rangle$	$\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathsf{Select}}$, $\tilde{\mathbf{u}}^{\mathrm{id}}_{a^*}$, $\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathrm{Update}}$

The position embeddings $\bar{\mathbf{p}}_1, \dots, \bar{\mathbf{p}}_L \in \mathbb{R}^{d_{\mathrm{PE}}}$ are orthonormal with $d_{\mathrm{PE}} = L$, and $\mathbf{h}_i^{(0)} = \mathbf{x}_i + \bar{\mathbf{p}}_i^{\mathrm{pos}}$. We write v_i for the token at a token position i, $\mathbf{c}_a^{(t)}$ for the content of slot c_a at step t, and $\mathbf{h}^{(1/2)}, \mathbf{h}^{(1)}, \mathbf{h}^{(3/2)}, \mathbf{h}^{(2)}$ for the stream after each sublayer.

Notation. We take $d_{\mathrm{TE}} = |\mathcal{V}| = |\mathcal{S}| + |\mathcal{A}| + 6$, so $\{\tilde{\mathbf{u}}_v\}_{v \in \mathcal{V}}$ is an orthonormal basis of $\mathbb{R}^{d_{\mathrm{TE}}}$. For $\mathbf{w} \in \mathbb{R}^{d_{\mathrm{TE}}}$ and a block $B \in \{\mathrm{id}, \mathrm{buf}_1, \mathrm{buf}_2\}$, $\mathbf{w}^B \in \mathbb{R}^d$ is $\mathbf{w}$ placed in block B and zero elsewhere, and likewise $\bar{\mathbf{p}}_i^{\mathrm{pos}}$. For $\mathcal{X} \subseteq \mathcal{V}$ let $\tilde{\mathbf{u}}_{\mathcal{X}} := \sum_{v \in \mathcal{X}} \tilde{\mathbf{u}}_v$,

$$\mathbf{\Pi}^B_{\mathcal{X}} := \sum_{v \in \mathcal{X}} \tilde{\mathbf{u}}_v^B \big(\tilde{\mathbf{u}}_v^B\big)^\top, \qquad \mathbf{S}^{\mathcal{X}}_{A \to B} := \sum_{v \in \mathcal{X}} \tilde{\mathbf{u}}_v^B \big(\tilde{\mathbf{u}}_v^A\big)^\top,$$

the projector onto the $\mathcal{X}$-directions of block B and the map that copies them from block A to block B. We drop the superscript $\mathcal{X}$ when $\mathcal{X} = \mathcal{V}$ and write $\mathbf{\Pi}^B_v := \mathbf{\Pi}^B_{\{v\}}$. To avoid a clash with the Q-function, the attention parameters are $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_O$. Query and key vectors have standard basis $\mathbf{e}_0, \mathbf{e}_1, \dots$, and $\sigma_{ij} = \langle \mathbf{W}_Q \mathbf{h}_i, \mathbf{W}_K \mathbf{h}_j \rangle$ is the score of query i on key $j \le i$. An MLP is $\mathbf{h} \mapsto \mathbf{W}_2\, \mathrm{ReLU}(\mathbf{W}_1 \mathbf{h} + \mathbf{b}_1)$. Finally, let

$$M_t := \max_{a \in \mathcal{A}} Q_t(s_{t+1}, a), \qquad \mathcal{A}_t^* := \arg\max_{a \in \mathcal{A}} Q_t(s_{t+1}, a),$$

$$\bar{\mathbf{u}}_t^* := \frac{1}{|\mathcal{A}_t^*|} \sum_{a \in \mathcal{A}_t^*} \tilde{\mathbf{u}}_a, \qquad \delta_t := \alpha\big(r_t + \gamma M_t - Q_t(s_t, a_t)\big).$$

Invariants. Every head's $\mathbf{W}_O \mathbf{W}_V$ and every MLP's $\mathbf{W}_2$ map into $\mathrm{buf}_1 \oplus \mathrm{buf}_2$. Hence, at every layer, $\mathrm{pos}(\mathbf{h}_i) = \bar{\mathbf{p}}_i$, $\mathrm{id}(\mathbf{h}_i) = \tilde{\mathbf{u}}_{v_i}$ at token positions, and $\mathrm{id}(\mathbf{h}_i) = \mathbf{0}$ at slots. In particular,

$$\langle \tilde{\mathbf{u}}_{\mathcal{V}}, \mathrm{id}(\mathbf{h}_i) \rangle = \mathbb{1}\{i \text{ is a token}\}, \qquad \langle \tilde{\mathbf{u}}_{\mathsf{BOS}}, \mathrm{id}(\mathbf{h}_j) \rangle = \mathbb{1}\{j = 1\}.$$

Hard-attention limit. Every score below is linear in the inverse temperatures β (routing) and $\beta_{\max}$ (Head 2.1). A head *attends to* a set J if the positions in J tie for the highest score and every other position scores lower by a margin that grows linearly in β and $\beta_{\max}$ (Appendix B.2). In the limit

$\beta, \beta_{\max} \to \infty$ of Theorem 4.1, such a head outputs $\mathbf{W}_O \frac{1}{|J|} \sum_{j \in J} \mathbf{W}_V \mathbf{h}_j$, and we compute the network in this limit.

Attention sink. Every head except the fixed-offset heads has the sink rows

$$\tfrac{\beta}{2}\, \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathcal{V}}\big)^\top \text{ in } \mathbf{W}_Q, \qquad \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathsf{BOS}}\big)^\top \text{ in } \mathbf{W}_K,$$

so every token scores $\beta/2$ on $\langle \mathsf{BOS} \rangle$, and every slot scores 0. Every $\mathbf{W}_V$ below maps $\mathbf{h}_1$ to $\mathbf{0}$, so a token that scores less than $\beta/2$ everywhere else reads $\mathbf{0}$.

D.2 Layer 1: routing, context fetch and Q-value evaluation

Heads 1.1 and 1.2: $\mathrm{FO}(\mathcal{A}, -1)$ and $\mathrm{FO}(\mathcal{A}, -2)$. $\mathbf{W}_Q$ and $\mathbf{W}_K$ are those of Lemma B.1 with $\mathcal{T} = \mathcal{A}$ and $\ell = 1$ (Head 1.1) or $\ell = 2$ (Head 1.2). Its hypotheses hold by the invariants, with $\langle \mathsf{BOS} \rangle$ at position 1 and the context slots as latent slots. Both heads use

$$\mathbf{W}_V = \mathbf{\Pi}^{\mathrm{id}}_{\mathcal{V} \setminus \{\mathsf{BOS}\}}, \qquad \mathbf{W}_O = \mathbf{S}_{\mathrm{id} \to \mathrm{buf}_1},$$

so a position holding a token $v_j \neq \mathsf{BOS}$ has value $\tilde{\mathbf{u}}^{\mathrm{buf}_1}_{v_j}$, and $\langle \mathsf{BOS} \rangle$ and the slots have value $\mathbf{0}$.

Head 1.3: context-slot fetch.

$$\mathbf{W}_Q = \tfrac{\beta}{2} \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathcal{V}}\big)^\top + \beta \sum_{k=1}^{n} \mathbf{e}_{k+1} \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{a_k}\big)^\top, \qquad \mathbf{W}_K = \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathsf{BOS}}\big)^\top + \sum_{p=1}^{L} \mathbf{e}_p \big(\bar{\mathbf{p}}^{\mathrm{pos}}_p\big)^\top,$$

$$\mathbf{W}_V = \mathbf{\Pi}^{\mathrm{buf}_2}_{\mathcal{S}}, \qquad \mathbf{W}_O = \mathbf{I}_d.$$

The scores are

$$\sigma_{ij} = \tfrac{\beta}{2} \langle \tilde{\mathbf{u}}_{\mathcal{V}}, \mathrm{id}(\mathbf{h}_i) \rangle \, \mathbb{1}\{j = 1\} + \beta \sum_{k=1}^{n} \langle \tilde{\mathbf{u}}_{a_k}, \mathrm{id}(\mathbf{h}_i) \rangle \, \mathbb{1}\{j = k+1\}.$$

A token $\langle a_k \rangle$ thus scores β on the slot c_{a_k} at position $k+1$ and $\beta/2$ on $\langle \mathsf{BOS} \rangle$. Any other token scores only $\beta/2$ on $\langle \mathsf{BOS} \rangle$, and a slot scores 0 everywhere.

Layer 1 MLP: Q-value evaluation. With $2|\mathcal{S}|$ hidden units (s, σ), $\sigma \in \{\pm 1\}$,

$$\mathbf{W}_1 = \sum_{s \in \mathcal{S}} \sum_{\sigma = \pm 1} \mathbf{e}_{s,\sigma} \big(\sigma\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_{s} + C\, \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{s}\big)^\top, \qquad \mathbf{b}_1 = -C \sum_{s,\sigma} \mathbf{e}_{s,\sigma},$$

$$\mathbf{W}_2 = \sum_{s \in \mathcal{S}} \sum_{\sigma = \pm 1} \sigma\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}} \mathbf{e}^\top_{s,\sigma}.$$

With $x_s := \langle \tilde{\mathbf{u}}_s, \mathrm{buf}_2(\mathbf{h}) \rangle$ and $e_s := \langle \tilde{\mathbf{u}}_s, \mathrm{buf}_1(\mathbf{h}) \rangle$, the pre-activations are $\sigma x_s - C(1 - e_s)$, so by Lemma B.2

$$\mathrm{MLP}_1(\mathbf{h}) = \Big(\sum_{s \in \mathcal{S}} e_s x_s\Big) \tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}} \qquad \text{whenever } e \in \{0,1\}^{\mathcal{S}} \text{ and } \max_s |x_s| \leq C.$$

Lemma D.1 (Stream after Layer 1). *Suppose that $\mathbf{c}^{(t)}_a = \sum_s Q_t(s,a)\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_s$ for all $a \in \mathcal{A}$ and that $\|Q_t\|_\infty \leq C/2$. Then $\mathbf{h}^{(1)}_i = \mathbf{h}^{(0)}_i + \mathbf{\Delta}_i$, where:*

(i) $\mathbf{\Delta}_i = \mathbf{0}$ at every token that is not an action token;

(ii) at $\langle a_t \rangle$ (position $n+4$),

$$\mathbf{\Delta} = \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{s_t} + \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{Q_{\mathrm{curr}}} + \sum_s Q_t(s,a_t)\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_s + Q_t(s_t,a_t)\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}};$$

(iii) at the candidate $\langle a_i \rangle$ (position $n+6+2i$), with $w_1 = Q_{\mathrm{next}}$ and $w_i = a_{i-1}$ for $i \geq 2$,

$$\mathbf{\Delta} = \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{s_{t+1}} + \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{w_i} + \sum_s Q_t(s,a_i)\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_s + Q_t(s_{t+1},a_i)\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}};$$

(iv) at $\langle a^* \rangle$ (position $3n+8$),

$$\mathbf{\Delta} = \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{\mathsf{Select}} + \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{a_n} + \sum_s Q_t(s, a^*)\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_s;$$

(v) at the slot c_{a_j} (position $j+1$), $\mathbf{\Delta} = \frac{1}{j+1} \sum_{k=1}^{j} \mathbf{c}^{(t)}_{a_k}$.

Proof. Heads 1.1 and 1.2. Every action token sits at a position $\geq n+4 > 2$, so by Lemma B.1(a) it reads the identities of the tokens one and two positions back:

$$(\langle s_t \rangle, \langle Q_{\mathrm{curr}} \rangle) \text{ for } \langle a_t \rangle, \qquad (\langle s_{t+1} \rangle, \langle w_i \rangle) \text{ for } \langle a_i \rangle, \qquad (\langle \mathsf{Select} \rangle, \langle a_n \rangle) \text{ for } \langle a^* \rangle.$$

The two heads write disjoint directions of buf_1, since the token one back lies in $\mathcal{S} \cup \{\mathsf{Select}\}$ and the token two back in $\mathcal{A} \cup \{Q_{\mathrm{curr}}, Q_{\mathrm{next}}\}$. By Lemma B.1(b), every other token reads $\langle \mathsf{BOS} \rangle$. By Lemma B.1(c), a slot reads $\langle \mathsf{BOS} \rangle$ or slots. All of these have value $\mathbf{0}$.

Head 1.3. The token $\langle a_k \rangle$ attends to c_{a_k} and reads $\mathbf{\Pi}^{\mathrm{buf}_2}_{\mathcal{S}} \mathbf{h}^{(0)}_{k+1} = \mathbf{c}^{(t)}_{a_k}$. Every other token attends to $\langle \mathsf{BOS} \rangle$ and reads $\mathbf{0}$. The slot at position $j+1$ has zero query, so it averages positions $1, \ldots, j+1$ and reads $\frac{1}{j+1} \sum_{k \leq j} \mathbf{c}^{(t)}_{a_k}$.

MLP. At $\langle a_t \rangle$ and $\langle a_i \rangle$, buf_1 contains exactly one state direction (s_t, resp. s_{t+1}), so e is one-hot, and $|x_s| = |Q_t(s,a)| \leq C/2$. The MLP adds $Q_t(s_t, a_t)$, resp. $Q_t(s_{t+1}, a_i)$, along $\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}}$. At every other position $e = \mathbf{0}$, and $|x_s| \leq C$: at $\langle a^* \rangle$, $|x_s| \leq C/2$; at the other tokens, $x = \mathbf{0}$, since the only nonzero entry of buf_2 among them is r_t along $\tilde{\mathbf{u}}_r$; and at the slot c_{a_j}, $|x_s| \leq |Q_t(s, a_j)| + \max_k |Q_t(s, a_k)| \leq C$. So the MLP adds $\mathbf{0}$ there. □

D.3 Layer 2: selection, maximization and the Bellman update

Head 2.1: greedy selection and $\max$**.** With $B := 4\beta + 2\beta_{\max} C$ (any $B > \beta/2 + \beta_{\max} C$ works),

$$\mathbf{W}_Q = \tfrac{\beta}{2} \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathcal{V}}\big)^\top + \big(B\, \mathbf{e}_1 + \beta_{\max} \mathbf{e}_2\big)\big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathsf{Select}} + \tilde{\mathbf{u}}^{\mathrm{id}}_{\mathrm{Update}}\big)^\top,$$
$$\mathbf{W}_K = \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathsf{BOS}}\big)^\top + \mathbf{e}_1 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathcal{A}} - \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{Q_{\mathrm{curr}}} - \tilde{\mathbf{u}}^{\mathrm{buf}_1}_{\mathsf{Select}}\big)^\top + \mathbf{e}_2 \big(\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}}\big)^\top,$$
$$\mathbf{W}_V = \mathbf{\Pi}^{\mathrm{id}}_{\mathcal{A}} + \mathbf{\Pi}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}}, \qquad \mathbf{W}_O = \mathbf{S}^{\mathcal{A}}_{\mathrm{id} \to \mathrm{buf}_1} + \mathbf{\Pi}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}}.$$

For the queries $\langle \mathsf{Select} \rangle$ and $\langle \mathrm{Update} \rangle$, $\sigma_{ij} = \frac{\beta}{2} \mathbb{1}\{j = 1\} + B\kappa_j + \beta_{\max} q_j$, with

$$\kappa_j := \langle \tilde{\mathbf{u}}_{\mathcal{A}}, \mathrm{id}(\mathbf{h}_j) \rangle - \langle \tilde{\mathbf{u}}_{Q_{\mathrm{curr}}} + \tilde{\mathbf{u}}_{\mathsf{Select}}, \mathrm{buf}_1(\mathbf{h}_j) \rangle, \qquad q_j := \langle \tilde{\mathbf{u}}_{Q_{\mathrm{next}}}, \mathrm{buf}_2(\mathbf{h}_j) \rangle.$$

κ_j marks the candidate action tokens, and $\beta_{\max} q_j$ is the Q-value score. Every other query has only the sink row.

Head 2.2: $\langle \mathrm{Update} \rangle \to \langle a_t \rangle$**.**

$$\mathbf{W}_Q = \tfrac{\beta}{2} \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathcal{V}}\big)^\top + \beta\, \mathbf{e}_1 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathrm{Update}}\big)^\top, \qquad \mathbf{W}_K = \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathsf{BOS}}\big)^\top + \mathbf{e}_1 \big(\tilde{\mathbf{u}}^{\mathrm{buf}_1}_{Q_{\mathrm{curr}}}\big)^\top,$$
$$\mathbf{W}_V = \mathbf{\Pi}^{\mathrm{buf}_1}_{\mathcal{S}} + \mathbf{\Pi}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}}, \qquad \mathbf{W}_O = \mathbf{\Pi}^{\mathrm{buf}_1}_{\mathcal{S}} + \tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{curr}}} \big(\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}}\big)^\top.$$

The key marks $\langle a_t \rangle$ by its $\tilde{\mathbf{u}}_{Q_{\mathrm{curr}}}$ entry in buf_1. The value copies $\tilde{\mathbf{u}}_{s_t}$ in buf_1 and moves $Q_t(s_t, a_t)$ from the $\tilde{\mathbf{u}}_{Q_{\mathrm{next}}}$ to the $\tilde{\mathbf{u}}_{Q_{\mathrm{curr}}}$ direction of buf_2.

Head 2.3: $\langle \mathrm{Update} \rangle \to \langle r_t \rangle$**.** $\mathbf{W}_Q$ is that of Head 2.2, and

$$\mathbf{W}_K = \mathbf{e}_0 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{\mathsf{BOS}}\big)^\top + \mathbf{e}_1 \big(\tilde{\mathbf{u}}^{\mathrm{id}}_{r}\big)^\top, \qquad \mathbf{W}_V = \mathbf{\Pi}^{\mathrm{buf}_2}_{r}, \qquad \mathbf{W}_O = \mathbf{I}_d.$$

Layer 2 MLP: Bellman update. With $2|\mathcal{S}|$ hidden units and $\mathbf{d} := \alpha\big(\tilde{\mathbf{u}}_r + \gamma\, \tilde{\mathbf{u}}_{Q_{\mathrm{next}}} - \tilde{\mathbf{u}}_{Q_{\mathrm{curr}}}\big)$,

$$\mathbf{W}_1 = \sum_{s \in \mathcal{S}} \sum_{\sigma = \pm 1} \mathbf{e}_{s,\sigma} \big(\sigma\, \mathbf{d}^{\mathrm{buf}_2} + C\, \tilde{\mathbf{u}}^{\mathrm{buf}_1}_s\big)^\top, \qquad \mathbf{b}_1 = -C \sum_{s,\sigma} \mathbf{e}_{s,\sigma},$$
$$\mathbf{W}_2 = \sum_{s \in \mathcal{S}} \sum_{\sigma = \pm 1} \sigma\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_s \mathbf{e}^\top_{s,\sigma}.$$

With $e_s := \langle \tilde{\mathbf{u}}_s, \mathrm{buf}_1(\mathbf{h}) \rangle$ and $x := \langle \mathbf{d}, \mathrm{buf}_2(\mathbf{h}) \rangle$, Lemma B.2 gives

$$\mathrm{MLP}_2(\mathbf{h}) = \sum_{s \in \mathcal{S}} e_s\, x\, \tilde{\mathbf{u}}^{\mathrm{buf}_2}_s \qquad \text{whenever } e \in \{0,1\}^{\mathcal{S}} \text{ and } |x| \leq C.$$

Lemma D.2 (Stream after Layer 2)**.** *Under the hypothesis of Lemma D.1,*

$$\mathrm{buf}_1\big(\mathbf{h}^{(2)}_{3n+7}\big) = \bar{\mathbf{u}}^*_t, \qquad \mathrm{buf}_2\big(\mathbf{h}^{(2)}_{3n+7}\big) = M_t\,\tilde{\mathbf{u}}_{Q_{\mathrm{next}}},$$
$$\mathrm{buf}_1\big(\mathbf{h}^{(2)}_{L}\big) = \tilde{\mathbf{u}}_{s_t} + \bar{\mathbf{u}}^*_t, \qquad \mathrm{buf}_2\big(\mathbf{h}^{(2)}_{L}\big) = \delta_t\,\tilde{\mathbf{u}}_{s_t} + M_t\,\tilde{\mathbf{u}}_{Q_{\mathrm{next}}} + Q_t(s_t,a_t)\,\tilde{\mathbf{u}}_{Q_{\mathrm{curr}}} + r_t\,\tilde{\mathbf{u}}_r,$$

and neither row depends on the token $\langle a^*\rangle$*.*

Proof. Only the rows of $\langle\mathsf{Select}\rangle$ (the read-out) and $\langle\mathrm{Update}\rangle$ (the latent write) are used, and by Lemma D.1(i) both equal their input embeddings before Layer 2.

Head 2.1. By Lemma D.1, $\kappa_j = 1$ at the n candidates and $\kappa_j = 0$ elsewhere: at $\langle a_t\rangle$ and $\langle a^*\rangle$, the $\tilde{\mathbf{u}}_{Q_{\mathrm{curr}}}$ or $\tilde{\mathbf{u}}_{\mathsf{Select}}$ entry of buf_1 cancels the action identity, and the other positions have no action identity and no such entry. Moreover, $q_j = Q_t(s_{t+1}, a_i)$ at the candidate $\langle a_i\rangle$, $q_j = Q_t(s_t,a_t)$ at $\langle a_t\rangle$, and $q_j = 0$ elsewhere. Hence

$$\sigma_{ij} \ge B - \tfrac{1}{2}\beta_{\max}C \ \text{ at every candidate}, \qquad \sigma_{ij} \le \max\big\{\tfrac{\beta}{2}, \tfrac{1}{2}\beta_{\max}C\big\} \ \text{ elsewhere},$$

a margin of at least $3.5\beta + \beta_{\max}C$. The candidates contain each action once and score $B + \beta_{\max}Q_t(s_{t+1},a_i)$, so the head attends to the candidates with $a_i \in \mathcal{A}^*_t$. Their values are $\tilde{\mathbf{u}}^{\mathrm{buf}_1}_{a_i} + Q_t(s_{t+1},a_i)\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}}$, so both queries receive $(\bar{\mathbf{u}}^*_t)^{\mathrm{buf}_1} + M_t\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{next}}}$.

Heads 2.2 and 2.3. At $\langle\mathrm{Update}\rangle$, Head 2.2 scores β only at $\langle a_t\rangle$, the one position with a $\tilde{\mathbf{u}}_{Q_{\mathrm{curr}}}$ entry in buf_1, so it writes $\tilde{\mathbf{u}}^{\mathrm{buf}_1}_{s_t} + Q_t(s_t,a_t)\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{Q_{\mathrm{curr}}}$. Head 2.3 scores β only at $\langle r_t\rangle$ and writes $r_t\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_r$. At $\langle\mathsf{Select}\rangle$, both heads have only the sink row and read $\mathbf{0}$.

MLP. At $\langle\mathrm{Update}\rangle$, $\mathrm{buf}_1 = \tilde{\mathbf{u}}_{s_t} + \bar{\mathbf{u}}^*_t$, so e is the one-hot vector of s_t, and $x = \delta_t$ with

$$|\delta_t| \le \alpha\Big(R_{\max} + (1+\gamma)\frac{R_{\max}}{1-\gamma}\Big) = \alpha C \le C.$$

So the MLP adds $\delta_t\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{s_t}$. At $\langle\mathsf{Select}\rangle$, $e = \mathbf{0}$ and $|x| = \alpha\gamma|M_t| \le C$, so it adds $\mathbf{0}$.

Independence of $\langle a^*\rangle$. By causality, $\langle a^*\rangle$ can only affect $\langle\mathrm{Update}\rangle$, as a key and value in Layer 2. By Lemma D.1(iv), it has $\kappa = 0$, $q = 0$, no $\tilde{\mathbf{u}}_{Q_{\mathrm{curr}}}$ entry and no $\tilde{\mathbf{u}}_r$ identity, so none of Heads 2.1–2.3 attends to it. □

D.4 Latent write and proof of Theorem 4.1

Let $\mathbf{W}_{\mathrm{ctx}} = \mathbf{\Pi}^{\mathrm{buf}_2}_{\mathcal{S}}$, which keeps the state directions of buf_2. The last position of a step is $\langle\mathrm{Update}\rangle$, so $\mathsf{TF}_\theta(\mathbf{x}) = \mathbf{h}^{(2)}_L$, and the latent write of Algorithm 1 updates the slot of a_t:

$$\mathbf{c}^{(t+1)}_{a_t} = \mathbf{c}^{(t)}_{a_t} + \mathbf{W}_{\mathrm{ctx}}\,\mathbf{h}^{(2)}_L, \qquad \mathbf{c}^{(t+1)}_{a} = \mathbf{c}^{(t)}_{a} \quad (a \ne a_t).$$

Proof of Theorem 4.1. Let Q_t be the tabular Q-learning iterate on the same transitions (s_t, a_t, r_t, s_{t+1}). We show by induction on t that

$$(\mathrm{I}_t): \qquad \mathbf{c}^{(t)}_a = \sum_{s\in\mathcal{S}} Q_t(s,a)\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_s \ \text{ for all } a \in \mathcal{A}, \qquad \|Q_t\|_\infty \le \frac{R_{\max}}{1-\gamma}.$$

At $t = 0$, the slots are $\mathbf{0}$ and $Q_0 \equiv 0$. Assume (I_t). By Lemma D.2, $\mathbf{W}_{\mathrm{ctx}}\mathbf{h}^{(2)}_L = \delta_t\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{s_t}$, so by (4.1)

$$\mathbf{c}^{(t+1)}_{a_t} = \sum_{s\ne s_t} Q_t(s,a_t)\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_s + \big(Q_t(s_t,a_t) + \delta_t\big)\tilde{\mathbf{u}}^{\mathrm{buf}_2}_{s_t} = \sum_s Q_{t+1}(s,a_t)\,\tilde{\mathbf{u}}^{\mathrm{buf}_2}_s,$$

and the other slots and entries are unchanged. The bound is preserved:

$$|Q_{t+1}(s_t,a_t)| = \big|(1-\alpha)Q_t(s_t,a_t) + \alpha(r_t + \gamma M_t)\big| \le (1-\alpha)\frac{R_{\max}}{1-\gamma} + \alpha\Big(R_{\max} + \frac{\gamma R_{\max}}{1-\gamma}\Big) = \frac{R_{\max}}{1-\gamma}.$$

This proves (I_{t+1}), so the latent tokens hold the columns of Q_t at every step. By Lemma D.2, the row of $\langle\mathsf{Select}\rangle$ carries $\bar{\mathbf{u}}^*_t$ in buf_1, so the read-out $\arg\max_a\langle\tilde{\mathbf{u}}_a, \mathrm{buf}_1(\mathbf{h}^{(2)}_{3n+7})\rangle$ returns a greedy action $a \in \mathcal{A}^*_t$. The parameters depend only on $|\mathcal{S}|$, $|\mathcal{A}|$, α, γ, $R_{\max}$ and the inverse temperatures, not on T, and

$$d = 3d_{\mathrm{TE}} + d_{\mathrm{PE}} = 3(|\mathcal{S}| + |\mathcal{A}| + 6) + 3|\mathcal{A}| + 9 = O(|\mathcal{S}| + |\mathcal{A}|).$$ □

Remark D.3 (Finite inverse temperatures)**.** At finite β, each routing head puts weight $e^{-\Omega(\beta)}$ outside its target. Among the candidates, Head 2.1 at finite $\beta_{\max}$ weights a by $\pi(a) \propto \exp(\beta_{\max} Q_t(s_{t+1}, a))$ and returns the soft maximum, which satisfies, using $xe^{-\beta_{\max} x} \le 1/(e\beta_{\max})$,

$$0 \le M_t - \sum_a \pi(a)\, Q_t(s_{t+1}, a) \le \frac{|\mathcal{A}| - 1}{e\, \beta_{\max}}.$$

The exact update is 1-Lipschitz in $\|\cdot\|_\infty$, so per-step errors accumulate at most additively over the horizon.

Remark D.4 (One network for all MDP sizes)**.** The only position-dependent parameters are the offsets of Heads 1.1 and 1.2 and the slot addresses of Head 1.3, which do not depend on $|\mathcal{A}|$. Building the parameters for $S_{\max}$ states and $A_{\max}$ actions ($d_{\mathrm{PE}} = 3A_{\max} + 9$) therefore gives a single θ for every MDP with $|\mathcal{S}| \le S_{\max}$ and $|\mathcal{A}| \le A_{\max}$. Any $C \ge 2R_{\max}/(1-\gamma)$ may be used in both gates and in B.

E Verification of Transformer Constructions with Robustness Checks

We instantiated the constructions of Appendices C and D as transformers with fixed weights, using only causal softmax heads, ReLU MLPs, residual additions and the latent write of Algorithm 1, and ran them autoregressively on the same inputs as the corresponding algorithms. Embeddings are one-hot, an orthonormal choice, and hard attention is replaced by softmax at finite inverse temperatures. We then relaxed the idealised assumptions of the constructions one at a time.

E.1 Exact execution

Weighted Majority. With $n = 4$ experts of accuracies drawn from $U[0.3, 0.9]$, $\eta = 0.1$ and $T = 500$, the output at $\langle p? \rangle$ differs from $\hat{p}_t$ of (1) by at most $1.5 \cdot 10^{-12}$ for every $\beta \ge 50$, the weights decoded from the latent equal the WMA weights in floating point, and the decisions agree in every round (Figure 6). The same holds at $T = 3000$ with $\eta = 0.5$.

Tabular Q-learning. A single network, built for 8 states and 4 actions, runs 252 random MDPs per setting: every combination of $|\mathcal{S}| \in \{2, \dots, 8\}$, $|\mathcal{A}| \in \{2, 3, 4\}$, $\alpha \in \{0.1, 0.2, 0.5\}$, $\gamma \in \{0.9, 0.95\}$, and deterministic or noisy, negative rewards, each for $T = 500$ steps of an ϵ-greedy behavior policy. The network's own selection is fed back as $\langle a^* \rangle$. The error $\max_t \|\hat{Q}_t - Q_t\|_\infty$ of the Q-values decoded from the context slots is set by $\beta_{\max}$, the only soft maximum (Remark D.3): it is at most $1.9 \cdot 10^{-4}$, $1.3 \cdot 10^{-5}$ and $1.1 \cdot 10^{-6}$ for $\beta_{\max} = 10^3, 10^4, 10^5$, for every $\beta \in \{50, 10^3, 10^4\}$. The greedy action agrees with that of tabular Q-learning at every step of every MDP, and feeding back the teacher's label instead of the network's own selection leaves the Q-values unchanged (**??**). Figure 7 shows one run.

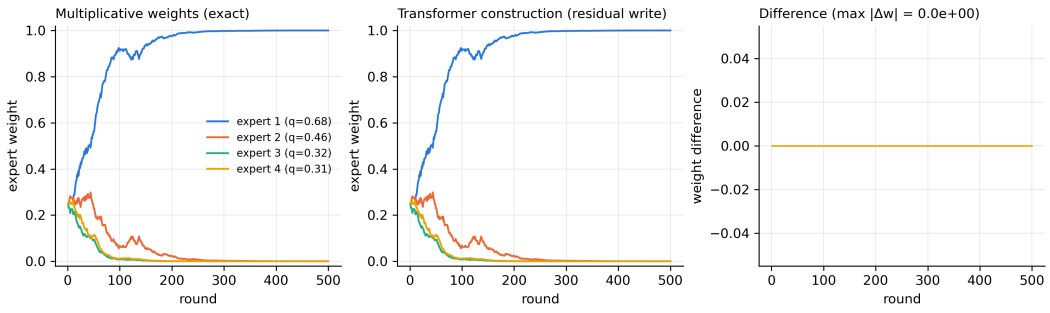


Figure 6: Weighted Majority weights (left) and the weights decoded from the latent of the handwired transformer (middle) for 4 experts, $\eta = 0.1$, $\beta = 10^3$; right: their difference, which is 0 in every round.

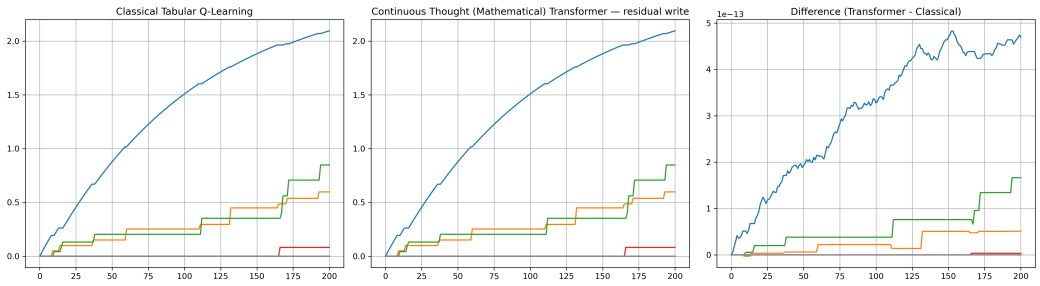


Figure 7: Tabular Q-learning (left) and the Q-values decoded from the context slots of the handwired transformer (middle) on a 4-state, 2-action MDP, $\alpha = 0.1$, $\gamma = 0.9$, $\beta = 10^3$, $\beta_{\max} = 10^4$; right: their difference, below 10^{-12} at every step.

E.2 Robustness to the construction assumptions

The constructions assume orthonormal token embeddings, disjoint buffers, hard attention and exact arithmetic. To relax the first two, we let $\mathbf{\Phi}$ collect the directions of the identity and buffer coordinates, embed the input as $\mathbf{\Phi x}$, and write every weight matrix $\mathbf{W}$ of the construction as $\mathbf{\Phi W R}$. The read-out $\mathbf{R}$ is either *tied*, $\mathbf{R} = \mathbf{\Phi}^\top$, which replaces each $\tilde{\mathbf{u}}_v$ by a non-orthonormal vector, or *dual*, $\mathbf{R} = \mathbf{\Phi}^+$. Residual-stream noise $\mathcal{N}(0, \sigma^2 \mathbf{I})$ is added after every sublayer. We use $n = 4$ experts with $\eta = 0.1$, and random MDPs with 6 states and 3 actions with $\alpha = 0.1$ and $\gamma = 0.9$, and the gate constant of Lemma B.2, $C = 2R_{\max}/(1-\gamma)$. Unless swept, $\beta = 10^3$ and $\beta_{\max} = 10^4$. Each setting is averaged over 20 instances with $T = 500$ (Figures 8 and 9).

(a) Attention hardness. Moderate inverse temperatures suffice. Both constructions are exact from $\beta = 50$ (errors $5.1 \cdot 10^{-15}$ for WMA and $8.0 \cdot 10^{-9}$ for Q-learning). The Q-learning error falls from $1.8 \cdot 10^{-3}$ at $\beta_{\max} = 50$ to $4.1 \cdot 10^{-6}$ at $\beta_{\max} = 10^3$, and greedy agreement is at least 0.989 for every $\beta_{\max} \geq 1$.

(b) Residual-stream noise. The error grows linearly in σ. At $\sigma = 10^{-5}, 10^{-4}, 10^{-3}$ it is $3.6 \cdot 10^{-3}$, $3.6 \cdot 10^{-2}$ and 0.28 for Q-learning, with greedy agreement at least 0.993, and $3.4 \cdot 10^{-3}$, $3.4 \cdot 10^{-2}$ and 0.31 for WMA ($\beta = 50$), with decision agreement at least 0.975. Since routing scores scale with β, very hard heads amplify noise, so moderate β is also the more robust choice.

(c) Non-orthogonal embeddings. Each block uses random unit vectors in $\mathbb{R}^d$. With the dual read-out, both constructions are exact for every $d \geq 16$, and with noise they match the orthonormal errors of (b) once d is moderately larger than the vocabulary. With the tied read-out, the interference between embeddings is fed back through the latent at every step: the WMA error decreases with d, to 0.61 at $d = 16384$, and the Q-learning error stays above 4.

(d) Overlapping buffers. The buffers are orthonormal internally, with cross-block cosines at most ρ. With the dual read-out, both constructions are exact for every $\rho \leq 0.5$, and at $\sigma = 10^{-4}$ the error increases only from $3.6 \cdot 10^{-2}$ to $4.3 \cdot 10^{-2}$ (Q-learning) and from $3.4 \cdot 10^{-2}$ to $3.9 \cdot 10^{-2}$ (WMA) between $\rho = 0$ and $\rho = 0.5$. With the tied read-out, the overlap compounds through the latent, already at $\rho = 0.01$.

Given linearly independent embeddings, orthonormality and disjoint buffers are therefore without loss of generality, moderate softmax temperatures suffice, and errors grow linearly with residual noise.

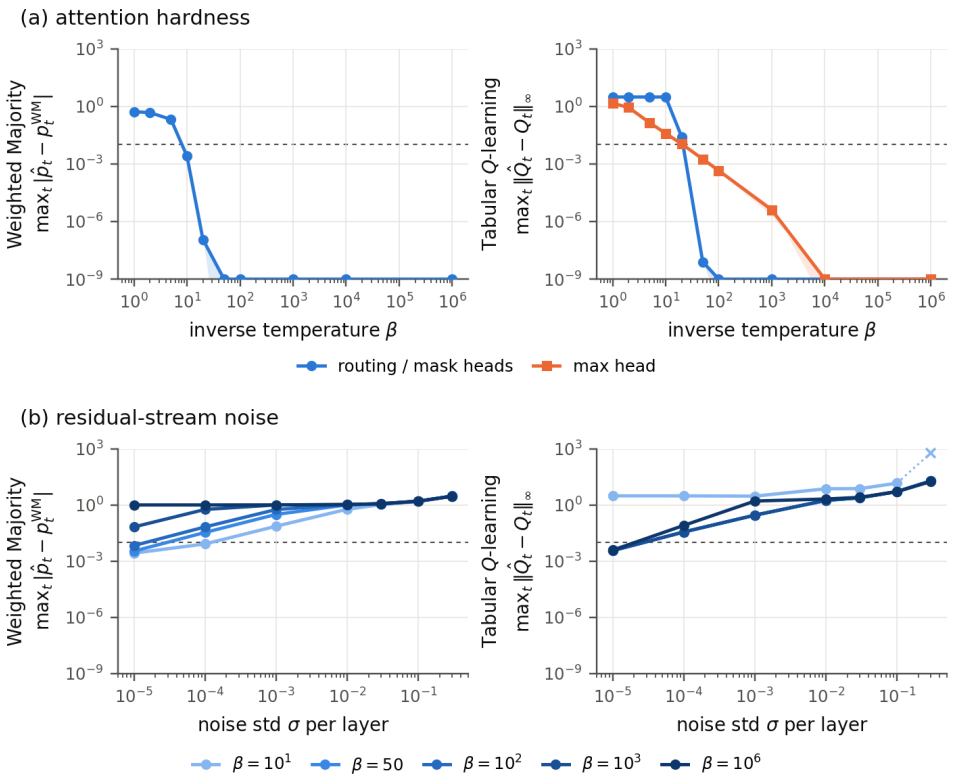


Figure 8: (a) Error against the inverse temperature of the routing heads (β) and of the maximum head of the Q-learning construction ($\beta_{\max}$). (b) Error against residual-stream noise σ for several β. Left: WMA, $\max_t |p_t^{\mathrm{TF}} - \hat{p}_t|$. Right: Q-learning, $\max_t \|\hat{Q}_t - Q_t\|_\infty$. Mean over 20 instances; a cross marks a diverged setting, and the dashed line is 10^{-2}.

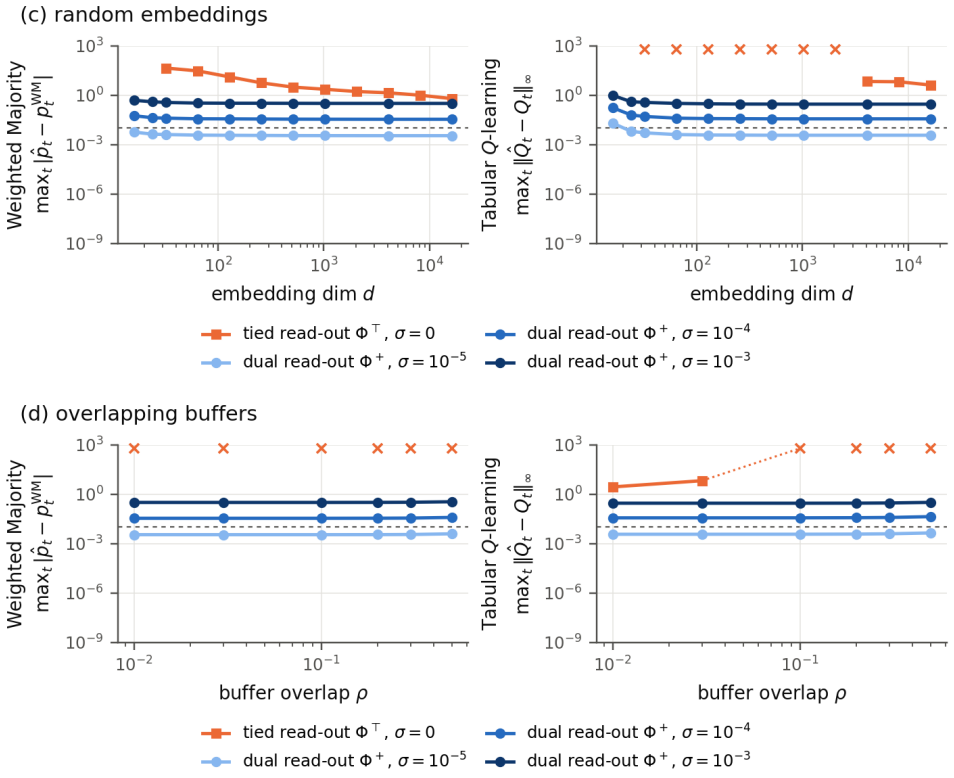


Figure 9: (c) Random unit token embeddings in $\mathbb{R}^d$. (d) Buffers with cross-block cosines at most ρ. Orange: tied read-out without noise. Blue: dual read-out with noise $\sigma \in \{10^{-5}, 10^{-4}, 10^{-3}\}$ (exact without noise). $\beta = 50$, $\beta_{\max} = 10^4$; axes as in Figure 8.

F Experiments with Continuous Transformers

This section describes the setup and additional results for the expert-prediction and tabular Q-learning experiments of Section 5. We provide our codebase here.

All models are pre-norm GPT-2 style decoders (Radford et al., 2019) that process one step at a time (one round of expert prediction, one transition of Q-learning). Attention is causal within a step and there is no attention across steps, so the context tokens are the only channel between steps, and gradients flow through them across steps. Let $\mathbf{h}$ be the final hidden state at the last token of a step ($\langle w? \rangle$ or $\langle \mathrm{Update} \rangle$) and $\mathbf{c}$ the context token it is written to. The *continuous transformer* uses the write of Algorithm 1, $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{W}_{\mathrm{ctx}}\mathbf{h} + \mathbf{b}_{\mathrm{ctx}}$, with learned $\mathbf{W}_{\mathrm{ctx}}$ and $\mathbf{b}_{\mathrm{ctx}}$. The *discrete transformer* instead sets $\mathbf{c}$ to the embedding of the vocabulary token $\arg\max_v \mathbf{u}_v^\top \mathbf{h}$; during training this choice is a straight-through Gumbel-softmax sample whose temperature is annealed from 2.0 to 0.5. The two models are otherwise identical.

F.1 Learning from Experts

F.1.1 Experimental Setup Summary

1. **Model:** 2 layers, 4 heads, $d_{\mathrm{model}} = 64$, MLP width 256, dropout 0.1 (126K parameters). Each round is the 20-token sequence of Theorem 3.1 with $n = 4$ experts at fixed positions,

$$\langle z_t \rangle \ \langle e_1 \rangle \langle p_{1,t} \rangle \cdots \langle e_4 \rangle \langle p_{4,t} \rangle \ \langle p? \rangle \ \langle y_t \rangle \ \langle e_1 \rangle \langle \ell_{1,t} \rangle \cdots \langle e_4 \rangle \langle \ell_{4,t} \rangle \ \langle w? \rangle .$$

   The single context token $\langle z_t \rangle$ starts as a learned vector initialized to $\mathbf{0}$ and is written from $\langle w? \rangle$. The decision is a linear read-out of $\langle p? \rangle$, which cannot attend to $\langle y_t \rangle$ or the losses. No round index is given.

2. **Dataset:** The expert qualities are drawn uniformly at random from $[0.3, 0.9]$ once per sequence. At each round the true label is a random bit, each expert predicts it correctly with probability equal to its quality, and it gets a loss of 0 if correct and 1 if wrong. A fresh set of 20,000 sequences is drawn every epoch; the validation set has 500 sequences of 200 rounds. The loss is $\mathcal{L} := \mathrm{BCE}(\text{predicted decision}, \text{true label})$ at $\langle p? \rangle$; the context is never supervised.

3. **Curriculum Strategy:** 16 stages truncate the sequences to $T \in \{5, 10, \ldots, 50, 65, 80, 95, 130, 165, 200\}$ rounds, so the model never sees longer sequences than the current stage allows.

4. **Training:** AdamW ($\beta = (0.9, 0.95)$, weight decay 10^{-2}), batch size 64, gradient clipping at 1.0, and backpropagation through all rounds of a sequence. Within a stage, the learning rate follows a cosine schedule from 10^{-3} to 10^{-4} over 8 epochs, with early stopping after 3 epochs without improvement of the validation loss. We train two seeds (42, 43) of each model.

5. **Evaluation:** Regret is the learner's number of mistakes minus that of the best expert in hindsight, on 100 fresh sequences per horizon $T \leq 1000$. The reference algorithms are MW with the learning rate $\eta = \sqrt{\ln n / T}$, the Bayes-optimal learner (a $\mathrm{Beta}(1, 1)$ posterior on each expert's quality, which exactly minimizes the training loss), and majority vote. Linear probes from $\langle z_t \rangle$ to each expert's centred cumulative loss (the MW state) are fit on rounds 5–95 of half of 1200 sequences and evaluated on the other half. The retention ρ is fitted as $\hat{s}_t = \rho\, \hat{s}_{t-1} + k\, \bar{\ell}_t + c$ on the decoded state.

F.1.2 Additional Results

Regret and the decoded state. Table 2 gives the regret at several horizons and the accuracy on three expert profiles from the LLM experiments (Appendix G). The anti-signal profile contains an expert that is right less than 10% of the time, outside the training range; the Bayes learner and the continuous transformer invert it, while MW cannot. The MW-state probe has $R^2 = 1.00$ on rounds 31–95 and stays at 0.99–1.00 on rounds 201–400, and the fitted retention is $\rho = 1.000$ for both seeds. A linear probe for the Bayes log-odds reaches $R^2 = 0.90$ on rounds 31–95 but becomes negative beyond round 95, since the log-odds is a nonlinear function of the counts. On the rounds where the Bayes learner and majority vote disagree (32% of rounds), the continuous transformer follows the

Bayes learner on 85% and 96% of them (two seeds), and the discrete transformer on 47% and 46%. For the discrete transformer, no probe decodes the MW state ($R^2 = 0.00$).

Table 2: Expert prediction: regret against the best expert at horizon T (100 sequences; mean over two seeds for the transformers) and accuracy on three expert profiles ($T = 100$).

	Regret				Accuracy		
	T=100	T=200	T=500	T=1000	stratified	flat	anti-signal
Continuous transformer	0.4	−1.4	−7.0	−21.1	0.934	0.621	0.816
Discrete transformer	12.6	25.8	61.8	146.1	0.812	0.553	0.397
MW	2.6	2.6	2.9	4.9	0.936	0.618	0.602
Bayes	0.1	−2.1	−10.5	−27.9	0.935	0.623	0.935
Majority vote	12.9	26.1	62.4	144.1	0.811	0.553	0.380

The state is used. Two interventions show that the model's decisions depend on the decoded state. First, we run two sequences A and B whose best experts differ for 50 rounds and continue B from A's context token: the continuous transformer then follows A's best expert on 92% and 93% of the rounds where the two best experts disagree (control: 6% and 7%), and still on 74% and 80% after 20 rounds, while the discrete transformer is unaffected (44–49%; Figure 10, right). Second, shifting $\langle z_t \rangle$ along the probe direction that adds loss to expert i lowers how often the next decision follows expert i from 61% to 27%, while a random direction of the same norm leaves it at 42–43% (Figure 11).

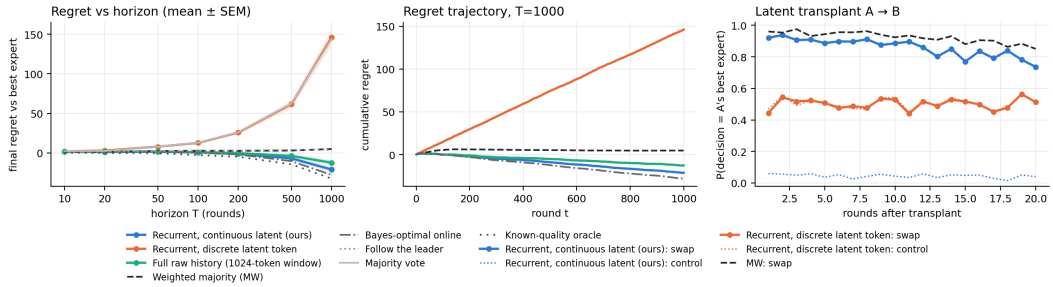


Figure 10: Expert prediction. (Left) Regret against the best expert as a function of the horizon (mean ± s.e.m.). (Middle) Cumulative regret over 1000 rounds. (Right) Fraction of decisions that follow sequence A's best expert after sequence B is continued from A's context token (swap) or from its own (control).

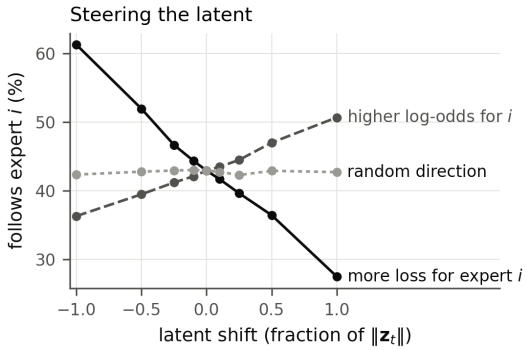


Figure 11: Steering the context token of the continuous transformer at round 80 along the probe direction that adds loss to expert i (solid), the probe direction of expert i's Bayes log-odds (dashed), or a random direction (dotted), by a fraction of its norm. The y-axis is how often the next decision follows expert i on rounds where expert i disagrees with the majority of the others.

Data on which MW is Bayes-optimal. We also train the continuous transformer on sequences in which a hidden expert, chosen uniformly, generates the labels, each flipped with probability $\epsilon = 0.2$.

Each expert predicts the opposite of a common random bit with probability $\delta_i \sim \mathcal{U}[0.2, 0.5]$, so the experts agree often and are hard to tell apart. The Bayes predictor is then MW with $\eta = \ln \frac{1-\epsilon}{\epsilon} = \ln 4$ (Appendix C.1). The training recipe is unchanged. Seed 43 reproduces this MW: the mean distance of its predicted probability to that of MW is 0.0015 on rounds 51–200 and 0.0016 on rounds 201–500, beyond the training horizon, and its log-loss regret at $T = 500$ is 1.51 (MW with $\eta = \ln 4$: 1.39). Seed 42 learns it partially (distance 0.065, regret 21.5). Since the model learns the Bayes predictor of its training distribution, it does not inherit MW's worst-case guarantee: an adversary that chooses each label after seeing the model's prediction drives its regret to 47 within 500 rounds, where MW is bounded by $\ln 4$ (Figure 12).

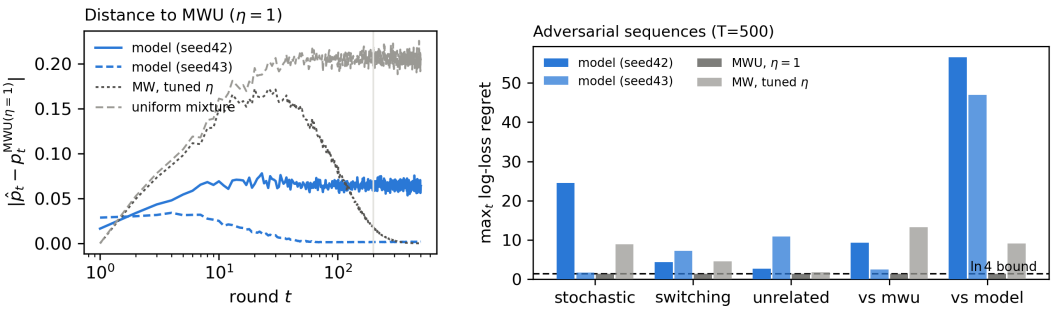


Figure 12: Data on which MW with $\eta = \ln 4$ is Bayes-optimal, both seeds. (Left) Mean distance of the predicted probability to that of MW over rounds; the vertical line is the training horizon. (Right) Maximum log-loss regret over 500 rounds on stochastic and adversarial label sequences; the dashed line is MW's bound $\ln 4$.

Robustness. Figure 13 evaluates the models on 14 further expert profiles ($T = 100$): several informative experts among uninformative ones, a dominant expert, close or spread qualities, a best expert that switches every 10, 25 or 50 rounds, qualities that drift over time, and correlated experts. The continuous transformer has lower regret than MW under switching, drift, a dominant expert and spread qualities, and higher regret when the qualities are close or the experts are correlated.

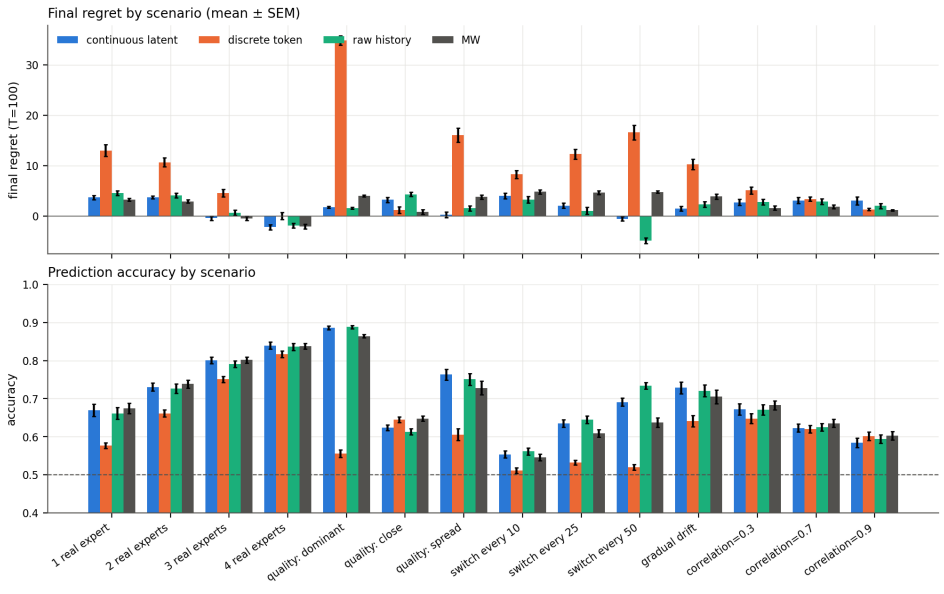


Figure 13: Final regret (top) and accuracy (bottom) at $T = 100$ on further expert profiles (mean $\pm$ s.e.m., 50 sequences each).

Attention. Figure 14 shows the attention of the continuous transformer by token type. At $\langle p? \rangle$, the first layer attends mostly to the prediction tokens and one head to the context token. At $\langle w? \rangle$, the first layer attends to the loss tokens, the operands of the update in Theorem 3.1.

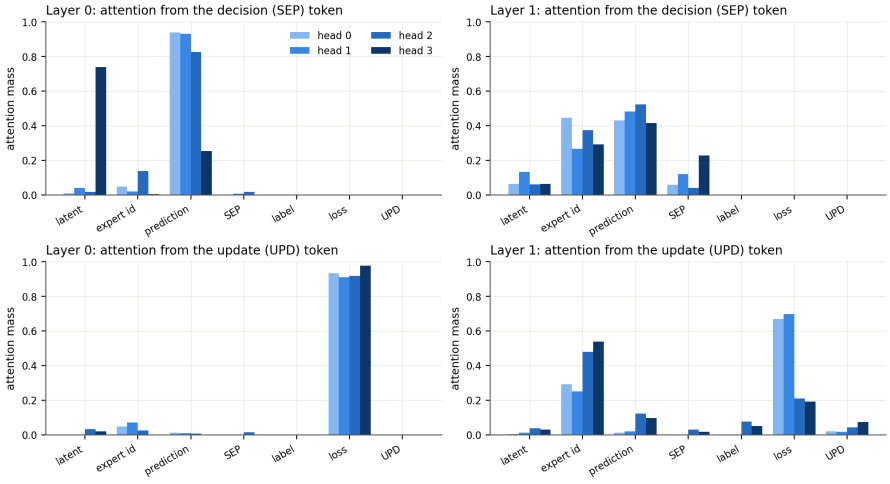


Figure 14: Attention mass per head from the decision token $\langle p? \rangle$ (top) and the update token $\langle w? \rangle$ (bottom) to each token type, continuous transformer (seed 43).

F.1.3 Computational Resources

Each model trains in 2 to 3 hours on a single GPU (NVDIA RTX A4000s, RTX 2080s).

F.2 Tabular Q-learning

F.2.1 MDP and Trajectory Sampling

Each training example is a single Q-learning episode on a freshly sampled finite MDP. For every episode we draw $|\mathcal{S}| \sim \mathcal{U}\{2, \dots, 8\}$, $|\mathcal{A}| \sim \mathcal{U}\{2, 3, 4\}$, the episode length $T \sim \mathcal{U}\{100, \dots, 200\}$, and the behavior policy's exploration rate $\varepsilon \sim \mathcal{U}[0, 1]$. To avoid implicit biases in the reward and dynamics priors, every episode is assigned a (reward family, transition concentration) cell from the Cartesian product

$$\text{reward} \in \{\text{peaked, bimodal, uniform, sparse, dense, Bernoulli}\},$$
$$\text{transition} \in \{\text{near-deterministic, uniform, diffuse}\}.$$

Cells are allocated in equal shares across the dataset and shuffled. Given a cell, the transition kernel $P(\cdot \mid s, a) \sim \mathrm{Dir}(\kappa \mathbf{1})$ is drawn with concentration $\kappa \in \{0.1, 1.0, 5.0\}$, and the mean reward table $R(s, a)$ is drawn from a Beta distribution whose shape parameters realize the named family (e.g. $\mathrm{Beta}(2, 2)$ for *peaked*); the *Bernoulli* family instead uses $R(s, a) \sim \mathcal{U}[0.1, 0.9]$. Episodes cycle through three task types in equal shares: the table R itself, a shifted table $R - 0.5$ with rewards of both signs, and a goal-state task in which every action in one random state pays 0.9 and all others -0.1, so that action values differ only through bootstrapping. Every observed reward carries Gaussian noise, $r_t = R(s_t, a_t) + 0.3\,\xi_t$ with $\xi_t \sim \mathcal{N}(0, 1)$.

Trajectory rollout. Starting from $s_0 \sim \mathcal{U}\{1, \dots, |\mathcal{S}|\}$ and $Q_0 \equiv 0$, we run T steps of tabular Q-learning with $\alpha = 0.2$ and $\gamma = 0.9$ under an ε-greedy behavior policy:

$$a_t \sim \begin{cases} \mathcal{U}(\mathcal{A}) & \text{w.p. } \varepsilon, \\ \arg\max_a Q_t(s_t, a) & \text{otherwise,} \end{cases} \tag{11}$$

$$s_{t+1} \sim P(\cdot \mid s_t, a_t), \tag{12}$$

$$Q_{t+1}(s_t, a_t) = (1 - \alpha) Q_t(s_t, a_t) + \alpha \big(r_t + \gamma \max_{a'} Q_t(s_{t+1}, a')\big), \tag{13}$$

$$a_t^\star = \arg\max_a Q_{t+1}(s_{t+1}, a). \tag{14}$$

We record the transition tuple $\tau_t = (s_t, a_t, r_t, s_{t+1}, a_t^\star)$ together with a snapshot Q_{t+1}. After the rollout we apply independent random permutations $\pi_\mathcal{S}, \pi_\mathcal{A}$ to the state and action labels (and re-index each Q_{t+1} accordingly), so the model cannot exploit a canonical token ordering. The dataset has $22{,}000$ episodes, split 90/10 into train and validation episodes.

The reward noise, the longer episodes and the shifted and goal-state tasks make the teacher's labels depend on averaging over past visits and on bootstrapping. On the dataset of our earlier version ($\alpha = 0.1$, deterministic rewards, 10–50 steps), a myopic learner ($\gamma = 0$) gave the teacher's label on 97.6% of steps; on the present data the two differ on 17% of steps.

F.2.2 Per-Step Input Sequence

Each transition τ_t is rendered into a length-$(2|\mathcal{A}| + 9)$ token sequence with two phases, preceded by the $|\mathcal{A}|$ context tokens:

$$\text{Phase 1 (predict } a^\star\text{):} \quad \text{BOS},\ Q_{curr},\ s_t,\ a_t,\ r_t,\ Q_{next},\ (s_{t+1}, a_1),\ \ldots,\ (s_{t+1}, a_{|\mathcal{A}|}),\ \text{SELECT},$$

$$\text{Phase 2 (commit } a^\star\text{, emit update):} \quad a_t^\star,\ \text{UPDATE}.$$

The number of (s_{t+1}, a_i) pairs equals the MDP's actual $|\mathcal{A}|$, not $A_{\max}$. The reward enters as the embedding of the reward token plus a learned direction scaled by r_t. SELECT precedes $a_t^\star$, so the prediction at step t does not see its own label.

F.2.3 Continuous Context and Recurrence

Every step is prefixed with $|\mathcal{A}|$ continuous *context tokens* $c_1, \ldots, c_{|\mathcal{A}|} \in \mathbb{R}^d$, one per action, initialized to $\mathbf{0}$ at the start of an episode. At step t, the transformer consumes $C_t = [c_1, \ldots, c_{|\mathcal{A}|}]$ together with the discrete sequence above, and the hidden state at the UPDATE position, h_t^{UPD}, is written into the context token of the executed action a_t:

$$C_{t+1}[a, :] = \begin{cases} C_t[a, :] + \mathbf{W}_{\text{ctx}} h_t^{\text{UPD}} + \mathbf{b}_{\text{ctx}} & a = a_t, \\ C_t[a, :] & \text{otherwise.} \end{cases}$$

This recurrence is the only path along which information flows between steps. For the discrete transformer, $C_{t+1}[a_t, :]$ is instead the embedding of the token decoded from h_t^{UPD}. Episodes are batched with a sampler that groups sequences sharing the same $|\mathcal{A}|$, so the context width is constant within a batch.

F.2.4 Experimental Setup Summary

1. **Model:** Pre-norm GPT-2 style decoder ($n_{\text{layers}} = 4$, $n_{\text{heads}} = 8$, $d_{\text{model}} = 256$, $d_{\text{ff}} = 1024$, dropout 0.1, 3.2M parameters) with the continuous-context interface above.
2. **Loss Function:** Per-step cross-entropy on the SELECT logits against the tabular target $a^\star$, masking the logits of actions beyond $|\mathcal{A}|$:

$$\mathcal{L} := \frac{1}{T} \sum_{t=1}^{T} \text{CE}\big(\text{SELECT logits}_t, a_t^\star\big).$$

   Since ties in $a_t^\star$ are broken at random, the best achievable validation accuracy is 97.5%.
3. **Curriculum:** Training is split into 3 equal stages over the action count, with $|\mathcal{A}| = 2$, $|\mathcal{A}| \in \{2, 3\}$ and $|\mathcal{A}| \in \{2, 3, 4\}$.
4. **Training:** AdamW ($\beta = (0.9, 0.95)$), learning rate $3 \cdot 10^{-4}$ with linear warmup (500 steps) then cosine decay, weight decay 10^{-2}, gradient clipping at 1.0, mixed precision, batch size 64. Backpropagation runs through full episodes with the context detached every 50 steps. The continuous transformer trains for 20 epochs and the discrete transformer for 15. Validation runs every 500 steps and at the end of each epoch, and the checkpoint with the lowest validation cross-entropy is kept. Validation accuracies are 92.6% (continuous) and 56.4% (discrete).
5. **Evaluation:** Teacher-forced metrics run the model exactly as in training on validation episodes. The Q-table probe is a ridge regression from each context token c_a to the

column $Q_t(\cdot, a)$, fit on half of the episodes with $|\mathcal{S}| = 8$ and evaluated on the other half. For the discount fit, we replay tabular Q-learners with $\alpha \in \{0.05, \ldots, 0.7\}$ and $\gamma \in \{0, 0.5, 0.7, 0.8, 0.9, 0.95\}$ on the same trajectories and measure how often their greedy action equals the model's. When the model acts on its own, its predicted action is executed (with probability ϵ a uniformly random action instead) and fed back as $a_t^\star$, on 50 MDPs with 8 states and 4 actions in which one action per state pays 1.0 and all others 0.05, for 1000 steps; returns are reported as a percentage of the optimal policy's return.

F.2.5 Additional Experimental Results

Copying the teacher's action. Since $a_{t-1}^\star$ is an input, a model could predict many labels by copying the last $a^\star$ revealed for the same next state. We therefore also evaluate on steps where this copy would not give the teacher's label and Q-learning disagrees with a simpler rule: a myopic learner ($\gamma = 0$), a learner with $\gamma = 0.5$, or the most-tried action. On these steps the continuous transformer follows Q-learning more often than the simpler rule, and the discrete transformer less often (Figure 15). Figure 16 shows the full (α, γ) fit on these steps: the agreement of the continuous transformer peaks at the teacher's $(0.2, 0.9)$, while that of the discrete transformer is nearly flat. The context tokens of the continuous transformer also decode the visit counts ($R^2 = 0.89$) and the part of Q produced only by bootstrapping, $Q_{\gamma=0.9} - Q_{\gamma=0}$ ($R^2 = 0.62$).

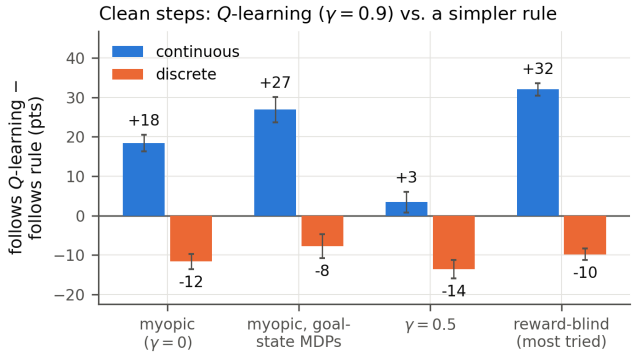


Figure 15: Steps where copying the last $a^\star$ would not give the teacher's label and Q-learning ($\gamma = 0.9$) and a simpler rule disagree: the percentage of steps on which the model follows Q-learning minus the percentage on which it follows the rule (95% intervals).

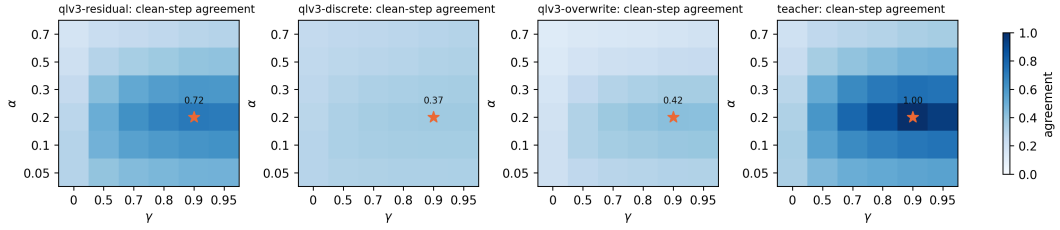


Figure 16: Agreement between the model's action and that of a tabular Q-learner with parameters (α, γ) replayed on the same trajectories, on the steps of Figure 15. Stars mark the best cell. The last panel is the teacher itself.

Acting greedily. With a greedy policy, the continuous transformer earns 27% of the optimal return on the 1000-step MDPs, close to a random policy (29%) and to greedy tabular Q-learning (30%). Its return increases with the exploration rate (35%, 38%, 44% and 47% for $\epsilon = 0.05, 0.1, 0.2, 0.3$), while that of the discrete transformer stays at 32–34%. The low greedy return is present from the first steps and does not grow with the horizon (Figure 17). Meanwhile, the norm of the context tokens grows roughly linearly, because each learned write has a roughly constant norm instead of shrinking like the temporal-difference increments of Q-learning. This does not affect the reads, since every pre-norm block normalizes the context token before attending to it; the construction of Theorem 4.1 has no LayerNorm and its context tokens stay bounded by $R_{\max}/(1-\gamma)$.

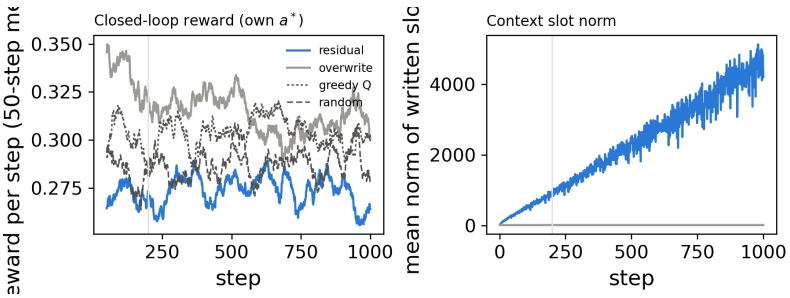


Figure 17: The model acting greedily on 1000-step MDPs. (Left) Reward per step (50-step moving mean). (Right) Mean norm of the values written into the context tokens; the vertical line marks the longest training episode.

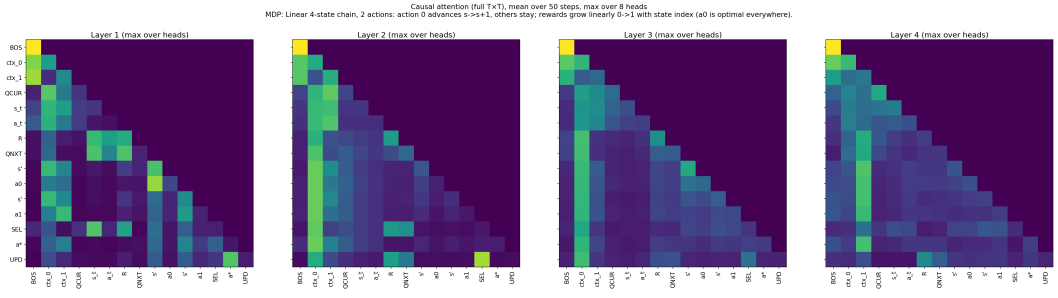


Figure 18: Full causal attention heatmap of the continuous transformer on a 4-state, 2-action linear-chain MDP where a_0 is optimal (mean over 50 steps, maximum over the heads of each layer). The context tokens receive attention from most later tokens, and the Update token attends to $a^\star$, SELECT, the reward token and Q_{next}, the operands of the temporal-difference update.

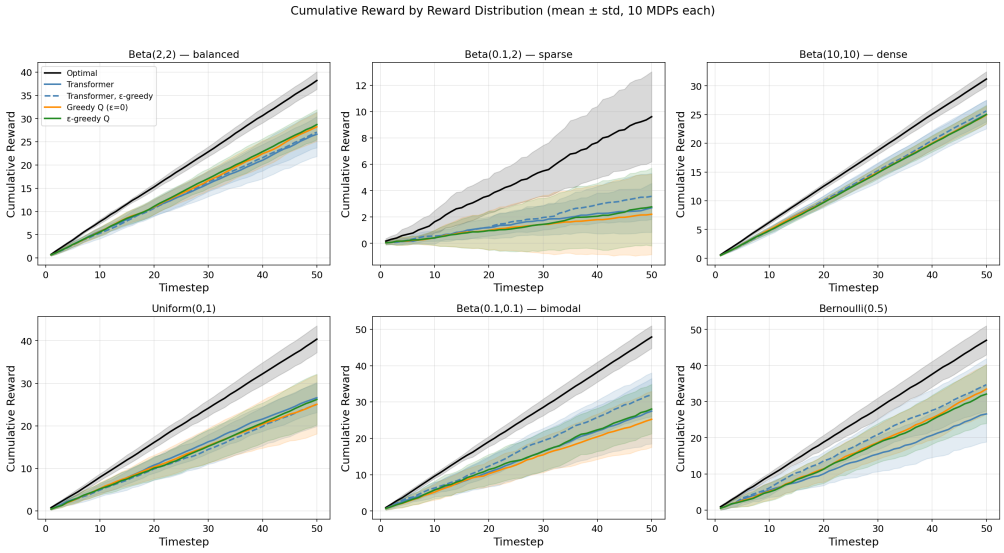


Figure 19: Cumulative reward over a 50-step horizon under six reward distributions, each averaged over 10 MDPs (mean $\pm$ std). We compare the optimal policy within the 50-step horizon against the continuous transformer and tabular Q-learning, each with a greedy and an ϵ-greedy ($\epsilon = 0.2$) policy. All learners track one another and trail the optimal policy by a similar margin, with the largest gap under sparse rewards.

F.2.6 Computational Resources

Each Q-learning training job takes about 3 to 4 hours on a single NVIDIA RTX A4000 GPU.

G LLM Experiments with Discrete Thought

This section provides the full experimental details for the LLM inference experiments in Section 5.3, including task generation, prompting protocols, full prompts, baselines, and additional results and detailed analyses of the notes generated by LLMs. The note that an LLM writes after each round and reads in the next is a visible, natural-language state channel, a discrete analogue of the continuous context tokens of Appendix F.

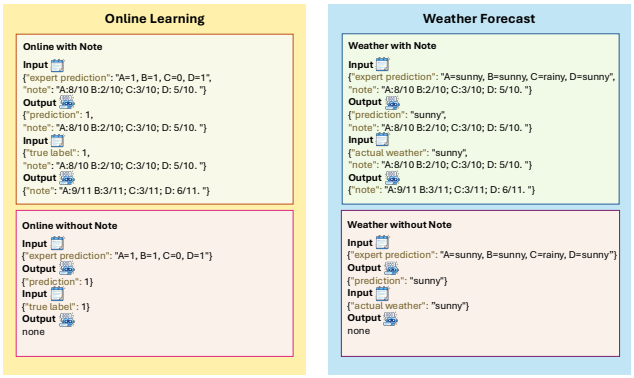


Figure 20: Prompt framing for LLM inference experiments. The (unsupervised) note shown is an illustrative example that shows up in the experiments.

G.1 Experimental Setup

We evaluate frontier LLMs on interactive expert-prediction sequences. Our main models are DeepSeek-V3 (61 layers, 671B parameters) DeepSeek-AI et al. (2025) and Qwen-3-14B (40 layers, 14.8B parameters) (Yang et al., 2025), both evaluated without thinking modes. All experiments are inference-only, with no parameter updates. Each evaluation instance is a length-100 sequence with four fixed experts, Expert A, Expert B, Expert C, and Expert D. At each round, the model observes the four expert predictions and outputs its own prediction. The true outcome is then revealed as feedback before the next round.

We use greedy decoding (temperature $= 0$) for DeepSeek-V3 and sampling with temperature $= 0.7$ and top-$p = 0.8$ for Qwen-3-14B, using the inference settings recommended for our API calls. Tools, code execution, and external retrieval are disabled. The model is instructed to output only valid JSON, with no explanations, intermediate calculations, or chain of thought. This setup lets us evaluate online decision-making behavior directly, rather than the model's ability to use tools or write out long reasoning steps.

Our main experiments use four prompting schemes, shown in Figure 20. These schemes are obtained by crossing two task framings with two state protocols.

The two task framings are:

- An **abstract online-learning framing**, where experts predict binary outcomes (0 or 1) and the task is described in terms of rounds and predictions.
- A **weather-forecasting framing**, where the same binary outcomes are presented as "sunny" or "rainy" and experts are described as weather forecasters.

The two state protocols are:

- **No-note protocol**. At each round, the expert predictions are sent as a user message, and the model responds only with its prediction. The true outcome is then appended as the next user

message, with no model response to this feedback message. The full conversation history is retained across rounds, so the model has implicit access to all prior expert predictions, its own predictions, and the revealed outcomes through the growing context.

- **Note protocol**. At each round, the model first predicts from the expert predictions and the current note. After the true outcome is revealed, the model is asked to produce a short note containing information useful for future prediction. This note is then included in subsequent rounds alongside the conversation history. The note is unsupervised and unconstrained in format, and provides an explicit visible state channel on top of the raw interaction history.

Both protocols maintain the full conversation history. A history-free note variant is discussed in Section G.6 and Section G.7.

The note protocol tests whether an explicit, compressed state improves online adaptation beyond what the model can directly extract from raw history. The note is unsupervised: the model may write any information it finds useful, such as cumulative accuracy tallies for each expert. We do not constrain the note to follow any particular algorithmic format, so any useful structure in the note is self-discovered by the model.

For every run, we record the full interaction trace, including the prompt variant, expert predictions, model predictions, revealed outcomes, raw model responses, parsed JSON outputs, and notes when applicable. We report cumulative regret against the best fixed expert in hindsight on each prefix of the sequence.

G.2 Data Generation and Expert Regimes

Each evaluation instance consists of a length-100 sequence with 4 experts. For each instance, the true label at each round is drawn i.i.d. as $y_t \sim \mathrm{Bernoulli}(0.5)$. Each expert i has a fixed accuracy $p_i \in [0, 1]$: at round t, expert i predicts y_t correctly with probability p_i and predicts $1 - y_t$ otherwise, independently across rounds and experts.

We construct three expert regimes by varying how expert accuracies are sampled. Each regime generates 30 instances.

Stratified. One expert is drawn from $\mathrm{Uniform}(0.9, 1.0)$, one from $\mathrm{Uniform}(0.65, 0.8)$, one from $\mathrm{Uniform}(0.55, 0.7)$, and one from $\mathrm{Uniform}(0.45, 0.6)$. This regime features a clearly dominant expert and tests whether the model can identify and follow it.

Flat. One expert is drawn from $\mathrm{Uniform}(0.6, 0.7)$ and the remaining three from $\mathrm{Uniform}(0.4, 0.6)$. All experts perform similarly, making it hard to distinguish the best expert from the rest. This regime tests the model's ability to extract a weak signal from noisy feedback.

Anti-signal. One expert is drawn from $\mathrm{Uniform}(0.6, 0.7)$, one adversarial expert from $\mathrm{Uniform}(0.0, 0.1)$, and the remaining two from $\mathrm{Uniform}(0.4, 0.6)$. The adversarial expert is correct less than 10% of the time, making it a strong *negative* signal: inverting its predictions would yield accuracy above 90%. This regime tests whether the model can detect a systematically misleading expert and either discount it or use its predictions inversely.

For every instance, expert identities are randomly permuted after sampling the four accuracies.

G.3 Full Prompts

We provide the full prompts for the four prompting schemes used in the LLM experiments. Here we include the exact system instructions and input-output formats used for reproducibility.

Online with note.

```
You are helping in an online learning setting.
```

```
In each round, the same four experts -- Expert A, Expert B, Expert C, and
↪  Expert D -- make predictions about a binary outcome. Each expert
↪  predicts either 0 or 1. Some experts may be consistently more
↪  accurate than others.

Your job is to make your own prediction based on the experts'
↪  predictions. Your goal is to be as accurate as possible over time.
↪  Use past outcomes to figure out which experts are most trustworthy
↪  and improve your future predictions.

This interaction has two kinds of turns.

1. Prediction turn

In a prediction turn, you are given:

* a short note from earlier rounds,
* this round's predictions from the four experts.

You should use the note and the experts' predictions to make your
↪  prediction for this round.

Input format:
{
"turn_type": "prediction",
"note": "<short text>",
"Expert_A": 0 or 1,
"Expert_B": 0 or 1,
"Expert_C": 0 or 1,
"Expert_D": 0 or 1
}

Output format:
{
"prediction": 0 or 1
}

2. Feedback turn

In a feedback turn, you are given:

* the previous note,
* this round's predictions from the four experts,
* the true label for this round.

You should produce a new short note for future rounds. The note should
↪  keep only the information that is most useful for making later
↪  predictions.

Input format:
{
"turn_type": "feedback",
"note": "<short text>",
"Expert_A": 0 or 1,
"Expert_B": 0 or 1,
"Expert_C": 0 or 1,
"Expert_D": 0 or 1,
"true_label": 0 or 1
}

Output format:
{
"note": "<short text>"
}
```

```
Important requirements:

* Do not use any external tools.
* Do not write or execute code.
* Do not provide any explanation or reasoning steps.
* Keep the note short and compact.
* The note must be at most 500 words long.
* The note should contain only the most useful information for future
↪  predictions.
* Do not repeat the full history.
* Output valid JSON only -- no explanation, no reasoning steps, no other
↪  text.
```

Online without note.

```
You are helping in an online learning setting.

In each round, the same four experts -- Expert A, Expert B, Expert C, and
↪  Expert D -- make predictions about a binary outcome. Each expert
↪  predicts either 0 or 1. Some experts may be consistently more
↪  accurate than others.

Your job is to make your own prediction based on the experts'
↪  predictions. Your goal is to be as accurate as possible over time.
↪  Use past outcomes to figure out which experts are most trustworthy
↪  and improve your future predictions.

This interaction has two kinds of turns.

1. Prediction turn

In a prediction turn, you are given:

* this round's predictions from the four experts.

You should use the experts' predictions to make your prediction for this
↪  round.

Input format:
{
"turn_type": "prediction",
"Expert_A": 0 or 1,
"Expert_B": 0 or 1,
"Expert_C": 0 or 1,
"Expert_D": 0 or 1
}

Output format:
{
"prediction": 0 or 1
}

2. Feedback turn

In a feedback turn, you are given:

* the true label for this round.

You do not need to output anything for the feedback turn. Use this
↪  feedback when making later predictions.

Input format:
```

```
{
"turn_type": "feedback",
"true_label": 0 or 1
}

Important requirements:

* Do not use any external tools.
* Do not write or execute code.
* Do not provide any explanation or reasoning steps.
* Output valid JSON only -- no explanation, no reasoning steps, no other
↪  text.
```

Weather with note.

```
You are helping with a daily weather guess.

Each day, the same four weather experts -- Expert A, Expert B, Expert C,
↪  and Expert D -- make predictions about the weather. Each expert
↪  predicts either "sunny" or "rainy". They each have their own way of
↪  making forecasts, so some may be consistently more reliable than
↪  others.

Your job is to make your own prediction based on the experts'
↪  predictions. Your goal is to be as accurate as possible over time.
↪  Use past outcomes to figure out which experts are most trustworthy
↪  and improve your future predictions.

This interaction has two kinds of turns.

1. Forecast turn

In a forecast turn, you are given:

* a short note from earlier days,
* today's predictions from the four experts.

You should use the note and the experts' predictions to make your
↪  prediction for today.

Input format:
{
"turn_type": "forecast",
"note": "<short text>",
"Expert_A": "sunny" or "rainy",
"Expert_B": "sunny" or "rainy",
"Expert_C": "sunny" or "rainy",
"Expert_D": "sunny" or "rainy"
}

Output format:
{
"prediction": "sunny" or "rainy"
}

2. Feedback turn

In a feedback turn, you are given:

* the previous note,
* today's predictions from the four experts,
* the actual weather for today.
```

```
You should produce a new short note for future days. The note should keep
↪  only the information that is most useful for making later guesses.

Input format:
{
"turn_type": "feedback",
"note": "<short text>",
"Expert_A": "sunny" or "rainy",
"Expert_B": "sunny" or "rainy",
"Expert_C": "sunny" or "rainy",
"Expert_D": "sunny" or "rainy",
"actual_weather": "sunny" or "rainy"
}

Output format:
{
"note": "<short text>"
}

Important requirements:

* Do not use any external tools.
* Do not write or execute code.
* Do not provide any explanation or reasoning steps.
* Keep the note short and compact.
* The note must be at most 500 words long.
* The note should contain only the most useful information for future
↪  predictions.
* Do not repeat the full history.
* Output valid JSON only -- no explanation, no reasoning steps, no other
↪  text.
```

Weather without note.

```
You are helping with a daily weather guess.

Each day, the same four weather experts -- Expert A, Expert B, Expert C,
↪  and Expert D -- make predictions about the weather. Each expert
↪  predicts either "sunny" or "rainy". They each have their own way of
↪  making forecasts, so some may be consistently more reliable than
↪  others.

Your job is to make your own prediction based on the experts'
↪  predictions. Your goal is to be as accurate as possible over time.
↪  Use past outcomes to figure out which experts are most trustworthy
↪  and improve your future predictions.

This interaction has two kinds of turns.

1. Forecast turn

In a forecast turn, you are given:

* today's predictions from the four experts.

You should use the experts' predictions to make your prediction for
↪  today.

Input format:
{
"turn_type": "forecast",
"Expert_A": "sunny" or "rainy",
"Expert_B": "sunny" or "rainy",
"Expert_C": "sunny" or "rainy",
```

```
"Expert_D": "sunny" or "rainy"
}

Output format:
{
"prediction": "sunny" or "rainy"
}

2. Feedback turn

In a feedback turn, you are given:

* the actual weather for today.

You do not need to output anything for the feedback turn. Use this
↪  feedback when making later guesses.

Input format:
{
"turn_type": "feedback",
"actual_weather": "sunny" or "rainy"
}

Important requirements:

* Do not use any external tools.
* Do not write or execute code.
* Do not provide any explanation or reasoning steps.
* Output valid JSON only -- no explanation, no reasoning steps, no other
↪  text.
```

G.4 Metrics and Baselines

Metrics. Our primary metric is **cumulative regret** against the best fixed expert in hindsight. Let $L_t = \sum_{s=1}^{t} \mathbf{1}[\hat{y}_s \neq y_s]$ denote the cumulative loss of a strategy and $L_t^{(i)} = \sum_{s=1}^{t} \mathbf{1}[\hat{y}_s^{(i)} \neq y_s]$ the cumulative loss of expert i. The regret at round t is

$$R_t = L_t - \min_{i \in [4]} L_t^{(i)}.$$

Regret measures how much worse a strategy performs compared to always following the single best expert. Lower regret indicates better adaptation to the reliable experts over the sequence. Each evaluation regime contains 30 independent instances. We report final regret at $T = 100$ averaged over these 30 instances. In regret curve plots, the shaded region shows ± 1 standard deviation across instances.

Baselines. We compare LLM performance against five reference strategies.

- **Multiplicative Weights (MW).** MW maintains a nonnegative weight $w_t^{(i)}$ for each expert, initialized uniformly, and predicts by weighted majority vote. After each round, weights are updated as $w_{t+1}^{(i)} = w_t^{(i)} \cdot \exp(-\eta\, \ell_t^{(i)})$ and renormalized, where $\ell_t^{(i)} \in \{0, 1\}$ is the loss of expert i at round t. The learning rate η is drawn uniformly from $[0.05, 0.5]$ per instance. MW serves as the classical long-memory online-learning reference strategy for tracking expert reliability over time.
- **Follow the Leader (FTL).** At each round, FTL predicts according to an expert with the lowest cumulative loss so far, breaking ties uniformly at random. This is a simple deterministic-history strategy that commits fully to the historically best expert.
- **Follow Previous Winners (FPW).** At each round, FPW predicts by majority vote among the experts who were correct in the previous round. If no expert is correct, or if there is a tie,

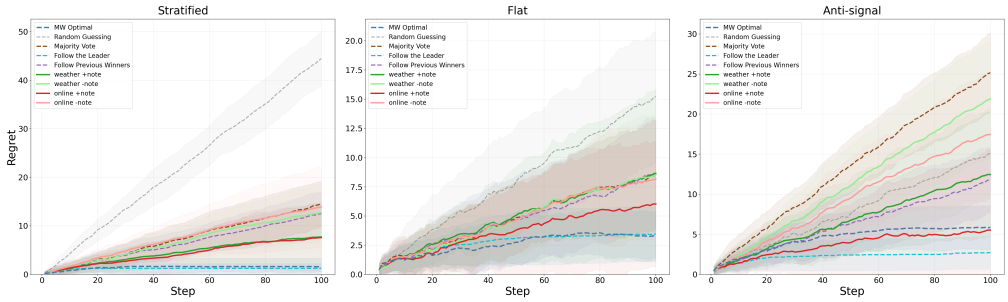


Figure 21: Cumulative regret of DeepSeek-V3 on three regimes. Regret is measured against the best fixed expert on each prefix. Note-based prompting reduces regret across the stratified, flat, and anti-signal regimes compared with no-note prompting and short-memory baselines.

it falls back to a uniform majority vote over all experts. This baseline uses only one round of feedback and therefore captures short-memory adaptation.

- **Majority Vote.** At each round, Majority Vote predicts by a uniform majority vote over all four experts, breaking ties uniformly at random. This is a non-adaptive baseline that ignores all feedback and treats experts as equally reliable.
- **Random Guessing.** Random Guessing predicts 0 or 1 uniformly at random at each round, independent of expert predictions and history. This provides a chance-level non-adaptive reference.

These baselines span different levels of adaptation. MW is the classical long-memory online-learning baseline, FTL tracks the best cumulative expert, FPW uses only the previous round of feedback, and Majority Vote and Random Guessing are non-adaptive references. Comparing LLMs with FPW helps distinguish genuine long-horizon state tracking from short-term reactions to the most recent feedback, while comparison with MW measures how far the model remains from a standard online-learning strategy.

G.5 Main Results

We first present the main LLM results under the standard multi-turn protocol described in Section G.1. In this setting, the full interaction history is retained across rounds. We compare DeepSeek-V3 with and without an explicit note, using the same underlying expert sequences, task framings, and evaluation metrics. We focus on DeepSeek-V3 because its API provides effectively unlimited context, allowing it to complete length-100 runs with the full retained history. Qwen3-14B, whose context window is limited to $40{,}960$ tokens, is evaluated under the history-free variant in Section G.6.

Table 3 reports the final cumulative regret of DeepSeek-V3 across all three expert regimes and both task framings. Figure 21 shows the corresponding regret trajectories over rounds.

	Weather		Online	
Dataset	+note	−note	+note	−note
Stratified	$\mathbf{7.7 \pm 7.0}$	12.7 ± 5.9	$\mathbf{7.6 \pm 7.0}$	13.9 ± 8.2
Flat	8.7 ± 7.1	8.5 ± 5.0	$\mathbf{6.0 \pm 5.4}$	8.2 ± 5.1
Anti-signal	$\mathbf{12.5 \pm 7.7}$	21.9 ± 7.0	$\mathbf{5.0 \pm 10.2}$	17.5 ± 7.6

Table 3: DeepSeek-V3 mean regret ($\pm$ 1 std) under the multi-turn protocol with full history (30 instances, $T = 100$). Bold indicates conditions where the note protocol improves over no-note. Both protocols retain the full conversation history; the note protocol additionally asks the model to produce a state summary after each feedback turn.

Stratified regime. This is the regime shown in Figure 5 in the main text. The four expert accuracies are sampled from separated intervals, $[0.9, 1.0]$, $[0.65, 0.8]$, $[0.55, 0.7]$, and $[0.45, 0.6]$, and are then

randomly assigned to Expert A, B, C, and D. Thus one expert is clearly dominant, making this the cleanest setting for testing whether the model can identify and exploit a reliable source from feedback. As discussed in Section 5.3, the regret curves of the no-note prompts remain close to simple short-memory baselines, while the note prompts achieve substantially lower regret. This suggests that the explicit note helps the model preserve cumulative expert-reliability information across rounds.

The note traces provide one explanation for this improvement. The model is never instructed to use any particular note format, yet in the weather framing it often adopts a cumulative-counter strategy within the first few steps. At step 0, the note is a qualitative description, such as `B,C,D correct; A wrong`. By step 1, it transitions to a running tally, such as `B,C,D correct 2/2; A wrong 2/2`; by step 3, it stabilizes into a compact counter, such as `A 1/4; B 3/4; C 3/4; D 3/4`. About 60% of instances adopt this counter format by the first feedback turn, rising to 70% by step 5. By step 99, notes such as `A:45/100, B:94/100, C:69/100, D:77/100` make the dominant expert immediately apparent. This format effectively stores the sufficient statistics needed by a Follow-the-Leader-style strategy, which explains why the note curves approach that baseline.

In the online framing, the model instead tends to record a round-by-round transcript, for example `Round1: A=1,B=0,C=0,D=0, label=0. A wrong, others correct.` Each step appends a new entry, and by step 10 the note begins compressing into pattern summaries, such as `A wrong when predicts 1 (rounds 1,4,7).` This round-listing format grows linearly and reaches more than $90{,}000$ context tokens by step 100 in the retained-history protocol. Nevertheless, it does not prevent the model from extracting the relevant signal when the full interaction history is also available.

Flat regime. In the flat regime, the best expert is only weakly better than the others, so the feedback signal is noisy and difficult to identify from short prefixes. Three of the four LLM conditions, weather +note, weather −note, and online −note, cluster near the FOLLOW PREVIOUS WINNERS and MAJORITY VOTE baselines. In particular, the note provides little benefit in the weather framing. The exception is online +note, which stays noticeably below the other LLM curves throughout the sequence. This suggests that, even in a low-signal environment, the combination of online framing and an explicit note can extract a weak but consistent advantage.

The note behavior helps explain the difference between regimes. About 80% of flat-regime instances converge to cumulative counters by step 5, similar to the stratified regime. Thus, the note format itself is often reasonable. However, the expert accuracy differences are small enough that a tally such as `A:65/100, B:54/100` may still not yield a confident ranking. The counter faithfully records the evidence, but the evidence itself remains ambiguous. In the online framing, the round-listing note preserves finer-grained information about individual outcomes. Together with the retained interaction history, this appears to give the model additional structure for acting on small accuracy differences that compact weather counters may miss.

Anti-signal regime. The anti-signal regime produces the largest separation between the note and no-note conditions. Since one expert has accuracy between 0 and 10%, MAJORITY VOTE is systematically hurt by this adversarial source and falls below RANDOM GUESSING. Without a note, the LLM performs between RANDOM GUESSING and MAJORITY VOTE, suggesting that it partially avoids the misleading expert but does not maintain a sufficiently stable estimate of expert reliability. Adding a note substantially changes the behavior. The note curves flatten, and in the online framing the note curve comes closest to the MW baseline among the LLM conditions. This indicates that the note helps the model preserve enough reliability information to identify which experts should be trusted and which should be ignored.

The mechanism is visible in the note evolution. By step 5, the cumulative counter already reveals the adversarial expert's near-zero accuracy, for example `B:1/6`, while the best expert begins to stand out, for example `C:4/6`. By step 10, the signal is clearer, as in `A:5/11, B:1/11, C:7/11, D:4/11`; by step 50, it is unambiguous, as in `A:19/51, B:3/51, C:31/51, D:24/51`. The model uses this information primarily for exclusion, following the best expert while ignoring the adversarial one. Across 30 instances, the final note correctly identifies the adversarial expert as the worst in 60% of parseable cases. However, the visible notes in our runs do not explicitly describe an inversion strategy, namely, predicting the opposite of the adversarial expert.

G.6 History-Free Note Variant and Cross-Model Comparison

We next evaluate a *history-free variant* of the note protocol for cross-model comparison. This variant modifies only the note protocol: the raw conversation history is not carried across rounds, and the note is the model's only explicit state. Each round consists of two independent API calls. In the prediction call, the model receives the system prompt, the note from the previous round, and the current expert predictions, and then outputs its prediction. In the feedback call, the model receives the same inputs together with the revealed outcome, and outputs an updated note.

For the $-$note baseline, we keep the standard multi-turn no-note protocol used throughout the paper: the model outputs only predictions, the true outcomes are appended to the conversation history, and the full interaction history is retained across rounds. Thus, the comparison asks whether a model-generated note, used as the only compressed state, can match or improve upon the model's ability to use raw conversation history without an explicit note.

This variant is useful for two reasons. First, it directly tests whether a natural-language note can serve as a compressed online-learning state. Second, it enables matched comparisons between models with different context-length capabilities. In the standard multi-turn note protocol, the context grows linearly with the number of rounds because every previous prediction turn, feedback turn, and note update remains in the conversation history. In our implementation, at $T = 100$, the conversation history reaches approximately $20{,}000$ tokens for the weather framing and more than $90{,}000$ tokens for the online framing. DeepSeek-V3 can complete these length-100 retained-history runs in our setup, but Qwen-3-14B is limited to $40{,}960$ tokens by its positional encoding and exhausts this budget around rounds 42 to 50. We therefore use the history-free variant for matched cross-model comparisons over the full length-100 horizon.

Table 4 compares DeepSeek-V3 and Qwen-3-14B under the history-free variant. This comparison does not test whether a note improves performance on top of raw history, as in Section G.5. Instead, it tests whether different LLMs can use a self-generated note as the *only* cross-round state for online prediction.

		Weather		Online	
Dataset	Model	+note	$-$note	+note	$-$note
Stratified	DeepSeek-V3	$\mathbf{8.1 \pm 6.3}$	12.7 ± 5.9	$\mathbf{9.3 \pm 8.0}$	13.9 ± 8.2
	Qwen-3-14B	32.0 ± 8.9	13.1 ± 6.3	20.7 ± 15.3	17.8 ± 7.6
Flat	DeepSeek-V3	$\mathbf{7.4 \pm 5.2}$	8.5 ± 5.0	$\mathbf{7.9 \pm 5.3}$	8.2 ± 5.1
	Qwen-3-14B	12.0 ± 5.3	8.6 ± 4.3	10.4 ± 9.4	9.7 ± 4.7
Anti-signal	DeepSeek-V3	$\mathbf{13.3 \pm 6.9}$	21.9 ± 7.0	$\mathbf{11.3 \pm 10.3}$	17.5 ± 7.6
	Qwen-3-14B	19.3 ± 9.9	19.5 ± 5.5	20.3 ± 17.7	16.5 ± 9.1

Table 4: Mean regret ($\pm$ 1 std) over 30 instances of length $T = 100$. The $+$note columns use the history-free note variant, where the note is the only cross-round state. The $-$note columns use the standard multi-turn no-note protocol, where the full interaction history is retained but no explicit note is provided.

The two models exhibit qualitatively opposite responses to the note mechanism.

DeepSeek-V3. The note consistently reduces regret across all regimes. DeepSeek-V3 often uses a cumulative-counter format for its notes, such as `A:45/100, B:94/100, C:69/100, D:77/100`, effectively maintaining a running tally of accuracy. Under the history-free variant, where the note is the only cross-round state, this counter serves as a useful statistic for identifying reliable experts.

Qwen-3-14B. In contrast, the note hurts performance for Qwen-3-14B in all six comparisons in Table 4. Across the Qwen-3-14B note runs in this table, the model never produces cumulative statistics. Its notes describe only the most recent feedback, for example, `Expert A was correct, Experts B, C, and D were incorrect`. Such notes overwrite rather than accumulate past information. Since the history-free variant provides no raw conversation history, Qwen-3-14B

effectively has only one round of memory despite having access to the note channel. As a result, the note can mislead the model into overreacting to the most recent feedback.

This comparison shows that the ability to use an external scratchpad for online learning is not a generic consequence of giving an LLM a note field. It depends on whether the model can discover an effective compression strategy for the relevant online state. DeepSeek-V3 discovers a cumulative-counting strategy without being instructed to do so, while Qwen-3-14B does not. This difference may be related to differences in model scale and training, but our experiment does not isolate the cause. Understanding which model properties enable such self-discovered compression strategies is an interesting direction for future work.

G.7 Ablation: History Retention and Note Format

We next compare two ways of using the note with DeepSeek-V3. In the standard multi-turn note protocol, the model receives both the retained conversation history and the latest note. In the history-free note variant, the model receives only the latest note and the current expert predictions, so the note is the sole cross-round state. This ablation asks whether the natural-language note is sufficient as a compressed state, and what additional role is played by the retained raw history.

Comparing the +note entries across Table 3 and Table 4, we find that most conditions are similar between the two settings. The main exception is the anti-signal regime under the online framing: retaining history reduces regret from 11.3 to 5.0, a gap of 6.3. The behavior is consistent with an implicit inversion strategy: with full history, the model can observe that the anti-signal expert is almost always wrong and use this pattern to improve prediction. However, this strategy is not explicitly described in the written note.

Note format summary. Table 5 summarizes the note formats adopted by DeepSeek-V3 at step 99 across all conditions, for both the retained-history and history-free settings. The contrast between settings and framings is striking.

			Note format at step 99 (out of 30)				
Setting	Dataset	Framing	Counter	Round list	Cond. ctr	Last-step	Other
Full history	Stratified	weather	21	0	1	8	0
		online	8	18	0	2	2
	Flat	weather	24	0	0	5	1
		online	8	19	0	3	0
	Anti-signal	weather	23	0	0	6	1
		online	10	17	0	3	0
History-free	Stratified	weather	16	0	10	2	2
		online	21	1	0	7	1
	Flat	weather	21	0	7	0	2
		online	27	1	0	2	0
	Anti-signal	weather	15	0	7	6	2
		online	23	2	0	1	4

Table 5: DeepSeek-V3 note format at step 99 over 30 instances per condition. "Counter" is a cumulative accuracy tally, e.g., `A:45/100, B:94/100`. "Round list" is an appended per-round transcript or compressed pattern summary, e.g., `Round1: A=1,B=0,C=0,D=0,label=0; A wrong, others correct. Round2: all 0,label=0; all correct.` "Cond. ctr" splits accuracy by weather outcome, e.g., `A: sunny 22/39, rainy 18/61`. "Last-step" records only the latest feedback, e.g., `A correct; B,C,D wrong`. "Other" includes vague, mixed-format, or malformed notes. Full history denotes the retained-conversation protocol; history-free denotes the note-only variant.

The table shows that note format is primarily driven by prompt framing and the availability of history. With retained history, weather prompts mostly produce cumulative counters, whereas online prompts mostly produce round-by-round transcripts. Once the raw history is removed, online prompts also shift toward counters. This suggests that round listing is used mainly when the conversation history

can serve as a fallback; when the note is the only cross-round state, the model is pushed toward a compressed statistic.

The weather framing also reveals a history-free failure mode: in roughly a quarter of final notes, the model writes conditional counters that split expert accuracy by weather outcome instead of maintaining a single overall reliability estimate. These notes preserve partial information, but make the relevant expert ranking less directly available than a standard counter. This failure mode nearly disappears when the raw history is retained. Overall, the same patterns appear across stratified, flat, and anti-signal regimes, suggesting that note format is governed more by framing and memory constraints than by the difficulty of the expert sequence.