TIME-π_0: A TIME SERIES ACTION MODEL FOR DECISION-MAKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series foundation models (TSFMs) have recently demonstrated a strong capability in modeling temporal dynamics, yet their potential for real-world operational decision-making remains largely unexplored. In this work, we formulate time series decision-making as the task of generating actions based on historical observations of an exogenous time series and the current state of a system. We introduce Time-π_0, a simple yet effective framework that adapts pretrained TSFMs for direct action generation. Specifically, a TSFM encodes temporal history into context tokens, while a lightweight flow-matching action expert combines these representations with the system state to generate discrete action chunks. We further construct `Cassandra`, a testbed comprising five real-world operational systems across energy, finance, telecommunication, supply chains and green industries. Each system couples an action-independent exogenous time series with a controllable system state. Experiments under single-chunk and rollout evaluations show that Time-π_0 achieves strong performance compared with forecast-then-optimize and direct offline policy baselines. Overall, our findings lay the groundwork for expanding the utility of TSFMs, shifting their paradigm from standard forecasting to actionable, real-world decision-making. The code and testbed are available at `https://anonymous.4open.science/r/TimePai0`.

1 INTRODUCTION

Real-world operational decisions, such as trading strategy (Kovačević et al., 2023), replenishing inventory (Gijsbrechts et al., 2022), or charging batteries (Hornek et al., 2025), depend on evolving temporal conditions, such as prices, product demand, and resource loads. Traditional learning has developed rich tools for forecasting, classification, and anomaly detection. Recent time series foundation models (TSFMs) (Liang et al., 2024) have pushed this paradigm further by learning temporal representations from massive, multi-source datasets.

The temporal signal learned by TSFMs is often an 'exogenous variable', such as price, demand, or resource load generated by broader socio-economic activities, which must be observed and anticipated but is not itself controlled by the decision maker. Actions instead operate on a concrete operational system, such as a warehouse with inventory (Gijsbrechts et al., 2022) or a battery with a state of charge (Hornek et al., 2025). Consequently, the same exogenous forecast can call for different actions depending on the evolving system state: a forecast of high demand may warrant replenishment when inventory is low, but no replenishment when inventory is sufficient. Existing applications of TSFM largely focus on prediction of exogenous time series (Yang et al., 2026), with actions determined by a separate downstream decision module, such as a model predictive control (MPC) loop over learned dynamics (Chua et al., 2018; Nagabandi et al., 2018; Zhuang et al., 2026; Liang et al., 2026), following a forecast-then-optimize paradigm in Figure 1 (a). This leaves open the question: *Can pretrained TSFMs generate actions by conditioning on the current state of a specific operational system?*

Action generation is well established in the robot learning. Behavior cloning (Bain & Sammut, 1995; Lipman et al., 2022a) learns policies directly from expert demonstrations, while Decision Transformer (Chen et al., 2021) formulates offline decision-making as conditional sequence modeling over states, actions, and returns. These approaches show that actions can be learned from trajectories without requiring an explicit forecast or online optimizer, and therefore provide natural

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.

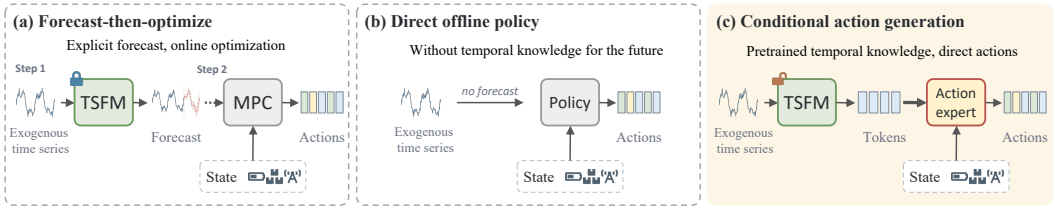

Figure 1: Time series decision-making paradigms. Unlike forecast-then-optimize and offline policies, Time-π_0 directly generates actions from TSFM tokens conditioned on the system state.

alternatives for our setting in Figure 1 (b). Meanwhile, recent vision-language-action (VLA) (Kim et al., 2024; Black et al., 2024b) models connect large pretrained vision models to action generation. Emerging world action models (Shen et al., 2026) take a further step by incorporating predictive world modeling, generating actions by future world states, showing the necessity of future aware abilities (Ye et al., 2026). Generative policies such as Diffusion Policy (Chi et al., 2025) similarly demonstrate the value of modeling action sequences with expressive conditional generative models. While action generation from trajectories is well established in robot learning, it remains largely unexplored for TSFMs. Here, the reusable knowledge lies in an often exogenous temporal process, whereas actions are grounded in specific operational systems with different states, action spaces, and objectives.

In this study, we introduce Time-π_0, which directly connects pretrained TSFMs to operational decisions. We formulate time series decision-making as an action generation task, conditioned on the historical exogenous time series and the current state of the controllable system. Our goal is not to build a one-size-fits-all model for time series decision-making, but to identify a minimal design that generates actions with pretrained TSFMs. To this end, Time-π_0 uses a TSFM to encode the temporal history into context tokens, while a lightweight flow-matching action expert conditions on these tokens and the system state to generate discrete action chunks, without producing an explicit forecast or invoking an optimizer at inference time. We further construct `Cassandra`, a testbed of five real-world operational systems spanning energy, finance, telecommunications, supply chains, and green industries. Experiments under single-chunk and rollout evaluations show that Time-π_0 achieves strong performance compared with forecast-then-optimize and direct offline policy baselines. We also show that Time-π_0 can accommodate different TSFM backbones while maintaining strong performance in terms of both accuracy and reward. Our main contributions in this work can be summarized as follows:

- **Formulation and Method.** We formulate time series decision-making as action generation conditioned on the history of an exogenous time series and the current state of the controllable system. Based on this formulation, we propose Time-π_0, which adapts a pretrained TSFM with a lightweight flow-matching action expert to generate discrete action chunks (Section 3).
- **New Testbed.** We construct `Cassandra`, a testbed of five real-world operational systems. The testbed pairs real exogenous time series with system states and covers three different decision horizons. Optimal actions are computed by dynamic programming in hindsight. This testbed supports both single-chunk evaluation and rollout evaluation (Section 4).
- **Empirical Findings.** Time-π_0 demonstrates strong performance against forecast-then-optimize and direct offline policy baselines, while remaining compatible with various TSFM backbones. Furthermore, we observe that higher action accuracy does not consistently translate to higher rewards, which motivates the reporting of both metrics during evaluation (Section 5).

2 Related Work

TSFMs for Decision-Making. Recent studies have extended TSFMs beyond forecasting to decision-making, and existing approaches fall into two categories according to what the TSFM predicts. In the first category, a TSFM forecasts the exogenous time series, and a downstream optimizer uses the forecasts to select actions (Zhang et al., 2026). In the second category, a TSFM is adapted or distilled into a dynamics model of the controlled system, which a model predictive controller queries to evaluate candidate actions (Zhuang et al., 2026; Liang et al., 2026; Wang, 2026). In both

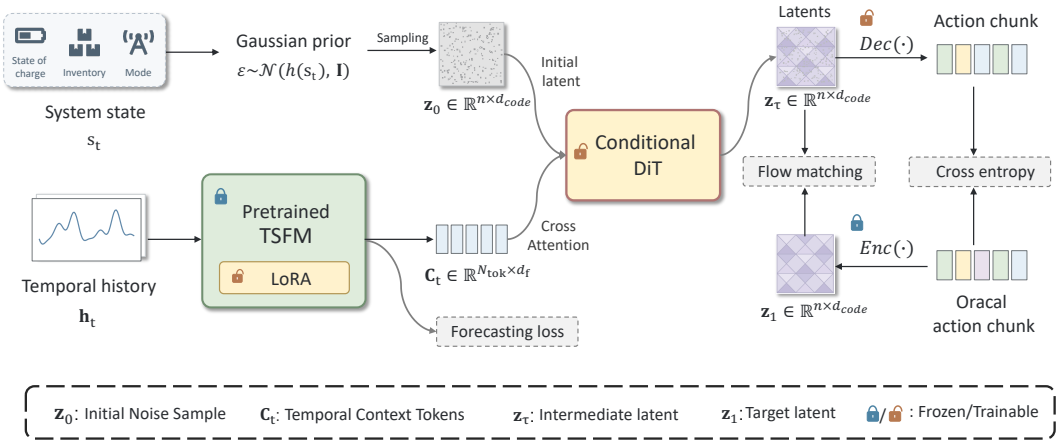

Figure 2: Architecture of Time-π_0. A LoRA-adapted TSFM encodes the temporal history into context tokens, and a conditional DiT attends to these tokens via cross-attention to transform state-conditioned Gaussian noise into latents that are decoded into a discrete action chunk.

categories, the TSFM, or a model distilled from it, serves only as a predictor, and actions are still obtained by a separate optimizer or planner at inference time. In contrast, Time-π_0 uses the TSFM representations to generate actions directly, without an explicit forecast or an online optimizer.

Generative Decision-Making. Policies can be learned directly from demonstrations through behavior cloning (Bain & Sammut, 1995) or conditional sequence modeling (Chen et al., 2021), and generative policies such as Diffusion Policy (Chi et al., 2025) further model action sequences with expressive diffusion or flow-matching models (Lipman et al., 2022b). VLA models extend this idea to pretrained foundation models (Brohan et al., 2023; Kim et al., 2024; Li et al., 2024; Bjorck et al., 2025; Black et al., 2024a). World action models go a step further by jointly predicting future observations and actions (Ye et al., 2026; Shen et al., 2026). Closest to our training setup, hindsight learning trains policies for decision problems with exogenous inputs from actions that are optimal given the realized future (Sinclair et al., 2023; Brown et al., 2010). Time-π_0 follows this setup by learning from hindsight-optimal action chunks, so it requires the system dynamics only to generate training labels and needs neither a dynamics model nor an online optimizer at inference time.

3 Methodology

In the following, we first present the problem definition, followed by our model design. At the core of Time-π_0 is the idea of adapting a pretrained TSFM to generate action chunks directly from historical exogenous observations and the current system state. As depicted in Figure 2, Time-π_0 consists of two main components: (1) TSFM-based temporal encoding (Section 3.2) and (2) flow-matching action generation (Section 3.3). A pretrained TSFM adapted with LoRA encodes the historical exogenous observations $\mathbf{h}_t$ into temporal context tokens, which are subsequently integrated into the flow-matching action expert via cross-attention to guide the transformation of state-conditioned Gaussian noise into action latents. A decoder then maps these latents to a discrete action chunk $\hat{\mathbf{a}}_t$.

3.1 Problem Definition

Let $x_t \in \mathcal{X}$ denote the exogenous observation, $s_t \in \mathcal{S}$ the operational state, and $a_t \in \mathcal{A}$ the action at time t. For a fixed exogenous horizon m and execution horizon n, define the observed history $\mathbf{h}_t := (x_{t-m+1}, \ldots, x_t)$, the future exogenous time series $\mathbf{x}_t := (x_{t+1}, \ldots, x_{t+n})$, and the action chunk $\mathbf{a}_t := (a_{t+1}, \ldots, a_{t+n})$. Let $\mathcal{A}_n(s_t)$ denote the set of feasible action chunks from state s_t, and let $R_n(\cdot)$ denote the cumulative reward over the horizon. An offline hindsight oracle μ^* observes the realized future time series $\mathbf{x}_t$ and returns an optimal action chunk:

$$\mathbf{a}_t^* = \mu^*(s_t, \mathbf{x}_t) \in \underset{\mathbf{a}_t \in \mathcal{A}_n(s_t)}{\arg\max}\ R_n(s_t, \mathbf{a}_t, \mathbf{x}_t). \tag{1}$$

Given an offline demonstration dataset $\mathcal{D} = \{(\mathbf{h}_t, s_t, \mathbf{a}_t^*)\}$, the learned model f_θ has access only to $(\mathbf{h}_t, s_t)$ and predicts $\hat{\mathbf{a}}_t = f_\theta(s_t, \mathbf{h}_t)$. Let $\mathcal{L}(\hat{\mathbf{a}}_t, \mathbf{a}_t^*)$ denote the loss between the predicted and oracle action chunks. The optimization problem is formalized as

$$\theta^* = \arg\min_\theta \mathbb{E}_{(\mathbf{h}_t, s_t, \mathbf{a}_t^*) \sim \mathcal{D}} \left[\mathcal{L}(f_\theta(s_t, \mathbf{h}_t), \mathbf{a}_t^*) \right]. \tag{2}$$

3.2 Conditioning on a Time Series Foundation Model

TSFMs learn temporal patterns from large and diverse time series corpora, providing a source of pretrained temporal knowledge for decision-making. We adopt a pretrained TSFM consisting of a backbone and its native forecasting head, adapt only the backbone with LoRA (Hu et al., 2021), and keep the forecasting head frozen. Given historical exogenous observations $\mathbf{h}_t$, the adapted backbone produces temporal context tokens $\mathbf{C}_t \in \mathbb{R}^{N_{\text{tok}} \times d_f}$, where N_{tok} is the number of tokens and d_f is the backbone hidden dimension. These tokens provide temporal context for the action generator.

Empirically, we find that auxiliary forecasting supervision helps adapt the TSFM to exogenous dynamics in the target domain and improves action-generation accuracy. This supervision may help the temporal representations capture future dynamics relevant to action selection. We therefore retain the frozen forecasting head as an auxiliary training interface and supervise its predictions with the ground-truth future trajectory $\mathbf{x}_t$ using an forecasting loss ℓ_{forecast}. The resulting gradients update the LoRA parameters, while the forecasting head is used only during training. Our design accommodates both encoder-only and decoder-only TSFMs, as further examined in Section 5.5.

3.3 Flow-Matching Action Expert

The action expert generates action chunks using the temporal context from the TSFM and the current system state. It is a lightweight Diffusion Transformer (DiT) (Peebles & Xie, 2023) that serves as the velocity network in flow matching, which learns to transform an initial noise sample $\mathbf{z}_0$ into a target latent $\mathbf{z}_1$, with the subscripts denoting the endpoints of flow time. Here, we use fixed sinusoidal action embeddings and a learned decoder. To apply flow matching to discrete actions, we use a fixed sinusoidal codebook $\mathbf{E} \in \mathbb{R}^{|\mathcal{A}| \times d_{code}}$, assigning each discrete action a d_{code}-dimensional embedding. Codebook lookup, denoted by Enc, maps the oracle action chunk to continuous target latents:

$$\mathbf{z}_1 = \mathrm{Enc}(\mathbf{a}_t^\star) \in \mathbb{R}^{n \times d_{code}}, \qquad \mathrm{Dec}(\mathbf{z}) \in \mathbb{R}^{n \times |\mathcal{A}|}. \tag{3}$$

Here, n is the chunk length and $|\mathcal{A}|$ is the number of discrete actions. The codebook provides fixed targets for flow matching, while Dec is a trainable two-layer MLP that maps each latent to discrete-action logits.

State and Temporal Conditioning. We use a system state-conditioned Gaussian prior, inspired by learnable priors for generative policies (Dai et al., 2026; Lee et al., 2021). Each position of the initial noise sample $\mathbf{z}_0$ is independently sampled from $\mathcal{N}(h(s_t), \mathbf{I}_{d_{code}})$, where $h(\cdot)$ is a learnable MLP mapping from the current system state to a d_{code}-dimensional mean vector. To incorporate temporal context, we use cross-attention (Rombach et al., 2022) to condition action-latent generation on temporal information. Specifically, the action expert uses the hidden representations of the current action latents as queries and the TSFM context tokens $\mathbf{C}_t$ as keys and values in cross-attention. This allows the expert to retrieve temporal information relevant to predicting the flow velocity.

Following conditional flow matching (Lipman et al., 2022b), we interpolate between the initial samples and target latents:

$$\mathbf{z}_\tau = (1 - \tau)\mathbf{z}_0 + \tau \mathbf{z}_1, \qquad \mathbf{v}^\star = \mathbf{z}_1 - \mathbf{z}_0, \tag{4}$$

where $\tau \sim \mathcal{U}[0, 1]$ denotes flow time. The action expert predicts the velocity $\widehat{\mathbf{v}} = \mathbf{v}_\theta(\mathbf{z}_\tau, \tau, \mathbf{C}_t)$. At inference, we integrate this velocity field from the system state-conditioned initial samples and decode the resulting latents into an action chunk.

Joint Training Objective. We train the action expert to match the target velocity $\mathbf{v}^\star$. To also supervise discrete action prediction, we reconstruct the target latent as $\widehat{\mathbf{z}}_1 = \mathbf{z}_\tau + (1 - \tau)\widehat{\mathbf{v}}$ and apply cross-entropy to the decoder output. Together with the auxiliary forecasting supervision, the training objective is

$$\mathcal{L}_{\text{total}} = \mathbb{E}\Big[\alpha \, \mathrm{MSE}(\widehat{\mathbf{v}}, \mathbf{v}^\star) + \beta \, \mathrm{CE}\left(\mathrm{Dec}_\theta(\widehat{\mathbf{z}}_1), \mathbf{a}_t^\star \right) + \gamma \ell_{\text{forecast}} \Big], \tag{5}$$

where ℓ_{forecast} is the auxiliary forecasting loss depending on native model such as quantile loss in Chronos-2, and $\alpha, \beta, \gamma > 0$ balance the three loss terms.

4 THE CASSANDRA TESTBED

To evaluate whether pretrained TSFMs can effectively support decision-making across diverse operational systems, we construct `Cassandra`, a testbed that pairs real world exogenous time series with operational systems capturing domain specific objectives and constraints. The tasks span energy (Hornek et al., 2025), finance (Kovačević et al., 2023), telecommunications (Salem et al., 2019; Liu et al., 2024), supply chains (Gijsbrechts et al., 2022) and green industries (Yang et al., 2026).

We describe the dataset construction and evaluation settings in Section 4.1 and oracle trajectory generation in Section 4.2. As a general principle, we adopt constraints and optimization objectives widely used in each operational system to guide the design of its oracle. We document the data sources and the system design in Appendix B.

4.1 DATASET CONSTRUCTION AND EVALUATION SETTINGS

For each operational system, we construct a dataset $\mathcal{D} = \{(\mathbf{h}_t, s_t, \mathbf{x}_t, \mathbf{a}_t^\star)\}$, where $(\mathbf{h}_t, s_t)$ are the model inputs, and $\mathbf{x}_t$ and $\mathbf{a}_t^\star$ provide forecasting and action supervision, respectively. Given the current system state s_t and the ground-truth future exogenous trajectory $\mathbf{x}_t$, a hindsight oracle computes the optimal action chunk $\mathbf{a}_t^\star$ by maximizing cumulative reward under the system-specific dynamics and operational constraints, such as battery energy limits as shown in Figure 5c in Appendix C. Table. 1 provides an overview.

Table 1: Overview of `Cassandra`. $|\mathcal{S}|$, $|\mathcal{A}|$, and #Traj. denote the number of initial states per window, actions, and trajectories, respectively.

Dataset	Task description	Time series observation	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	#Traj.
`battery`	Battery price arbitrage	ENTSO-E (ENTSO-E, 2026)	100	3	36,500
`economy`	Stock trading	Yahoo Finance (Yahoo Finance, 2026)	280	2	21,000
`telecom`	Station mode switching	Telecom Italy (Barlacchi et al., 2015)	4	4	496
`supply`	Inventory replenishment	M5 Forecasting (Makridakis et al., 2022)	301	300	21,973
`carbon`	Carbon-aware job scheduling	CAISO OASIS (CAISO, 2026)	100	12	54,700

Expanding Datasets with State Sampling. The same exogenous trajectory can require different decisions depending on the different system initial state. Training from a single fixed state would therefore provide limited coverage of the operating conditions encountered during deployment, which is a key limitation in other datasetsRoss et al. (2011); Ross & Bagnell (2010). To broaden this coverage, we randomly sample k valid states $\{s_t^{(j)}\}_{j=1}^k$ for each time window t and compute their corresponding oracle action chunks $\{\mathbf{a}_t^{\star(j)}\}_{j=1}^k$ (Section 4.2). Each window therefore contributes k training samples sharing the same exogenous history and future trajectory.

Expanding Datasets Across Horizons. Real world decision-making problems operate over different execution horizons. We therefore construct training examples at three horizon settings: Short, Medium, and Long. For each setting, we jointly vary the exogenous history horizon m and the execution horizon n, setting $m = n$ in this study for simplicity. The settings with $m \neq n$ are discussed in Appendix A.4. We compute the oracle action chunk for each horizon separatly. The specific horizon lengths differ across datasets (see Table 2 and Table 3), depending on the physical meaning of each task and the sampling resolution of the underlying time series (e.g., minute, hour, or day).

Open-Loop and Rollout Evaluations. We evaluate Time-π_0 under two complementary settings. In single-chunk open-loop evaluation, each history–state pair $(\mathbf{h}_t, s_t)$ is taken from the dataset, and the predicted n-step action chunk $\hat{\mathbf{a}}_t$ is evaluated against $\mathbf{a}_t^\star$ without execution. Samples are evaluated independently, so predicted actions do not affect subsequent inputs. In closed-loop rollout evaluation, $\hat{\mathbf{a}}_t$ is executed under the realized exogenous trajectory $\mathbf{x}_t$, yielding

$$\mathbf{s}_t := (s_{t+1}, \ldots, s_{t+n}) = T_n(s_t, \hat{\mathbf{a}}_t, \mathbf{x}_t)\,, \tag{6}$$

where T_n returns the n-step state sequence. The model replans every n steps using the resulting state s_{t+n} and updated history. Performance is measured by cumulative task reward over the rollout.

4.2 Oracle Trajectory Generation

We partition the exogenous series into non-overlapping history–future windows $(\mathbf{h}_t, \mathbf{x}_t)$ and pair each window with the initial states sampled in Section 4.1. For each pair $(s_t, \mathbf{x}_t)$, the future exogenous trajectory is known in hindsight, so the objective in Equation equation 2 becomes a deterministic finite-horizon problem (Brown et al., 2010). Its optimal value satisfies the Bellman recursion (Bellman, 1957; Bertsekas, 2017)

$$V_k(s) = \max_{a \in \mathcal{A}(s)} \Big[r(s, a, x_{t+k+1}) + V_{k+1}\left(f(s, a, x_{t+k+1})\right) \Big], \tag{7}$$

for $k = n-1, \ldots, 0$, where $\mathcal{A}(s)$ is the finite and feasible action set, f is the state transition function, and r is the one-step reward (e.g., the return in the stock trading task), and $V_n(s)$ is the terminal reward of the system, or zero if none applies. We solve this recursion with dynamic programming (DP). DP gives the best action at every step to maximum final reward, and together these form the oracle action chunk $\mathbf{a}_t^\star$. We use $\mathbf{a}_t^\star$ as the ground-truth actions for training and evaluation. Its return $V_0(s_t)$ is the highest return achievable in the window. Details for each system are in Appendix B.

5 Experiment

In this section, we evaluate the effectiveness of Time-π_0 for decision-making on `Cassandra`. We compare it with forecast-then-optimize and direct offline policy baselines on both single-chunk and rollout evaluations in Section 5.2 and Section 5.3, respectively. We present a battery arbitrage case study to show how decisions interact with exogenous time series, system state, and reward in Section 5.4. We demonstrate that Time-π_0 is compatible with a range of TSFM backbones in Section 5.5, with additional ablations provided in Section 5.5.

5.1 Experimental Setup

Baselines. We compare Time-π_0 with seven baselines spanning four categories. (1) A *domain heuristic* follows a standard domain specific operational rule for each system, providing a simple and realizable reference policy. For example, the battery heuristic charges below the 30th price percentile and discharges above the 70th percentile. (2) Two *forecast-then-optimize* baselines, MPC-native and MPC-TSFM, first forecast the future exogenous time series and then optimize the action sequence over the decision horizon. They differ only in the forecasting model: MPC-native uses a seasonal-naive forecaster, whereas MPC-TSFM uses zero-shot forecasts from Chronos-2 (Ansari et al., 2025). (3) Two *offline RL* baselines, DoubleDQN and DiscreteCQL (Kumar et al., 2020), learn action values from offline trajectories and act according to the learned values. (4) Two *imitation-based* baselines, Behavior Cloning and Decision Transformer, directly learn from oracle action trajectories. Behavior Cloning uses a flow-matching policy, while Decision Transformer uses a causal Transformer.

Implementation Details. We chronologically split the data in `Cassandra` into training, validation, and test sets with a 7:1:2 ratio. We use three-round rollouts for closed-loop evaluation, the maximum supported by the length of the test series. For the TSFM, Time-π_0 uses Chronos-2 (Ansari et al., 2025) by default and adapts it with rank-16 LoRA adapters (Hu et al., 2021). The action expert is a DiT with 4 blocks, a hidden dimension of 192, and operating on 128 dimensional action latents. Further details about baseline and implementation are provided in Appendix A.1 and Appendix A.3.

Metrics. We report action accuracy (**Acc**) and normalized reward (**NR**). Accuracy is the percentage of time steps at which the predicted action matches the oracle action, measuring how closely a method imitates the oracle. Normalized reward measures the task performance of the executed actions:

$$\mathrm{NR} = 100 \times \frac{R - R_{\mathrm{rand}}}{R_{\mathrm{DP}} - R_{\mathrm{rand}}}, \tag{8}$$

where R, R_{rand}, and R_{DP} are the cumulative rewards obtained by the evaluated method, a random policy that samples feasible actions uniformly, and the hindsight DP oracle, respectively, each averaged over the evaluation windows. Normalized reward values of 0 and 100 correspond to the random

Table 2: Single round (open-loop) evaluation results. S/M/L denote short, medium, and long horizons. The best and second-best results for each horizon and for Avg. are highlighted in **bold yellow** and underlined, respectively. Results are mean ± std over three runs.

Dataset		battery		economy		telecom		supply		carbon	
Method	Horizon	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑
Heuristic	S	45.4	32.9	47.9	12.5	43.0	52.5	1.0	68.3	31.8	70.9
	M	48.0	31.2	42.7	-23.3	24.0	50.8	0.0	79.1	34.0	72.2
	L	47.5	29.3	44.4	-0.0	48.0	65.6	0.0	81.7	36.5	70.6
	Avg.	46.9	31.1	45.0	-3.6	38.4	56.3	0.3	76.4	34.1	71.2
MPC-native	S	66.9±0.1	80.9±0.1	50.2±2.3	3.8±10.1	38.3±1.1	33.3±1.1	8.7±0.0	**87.0**±0.0	30.6±1.1	72.0±0.8
	M	52.4±0.3	51.8±0.6	49.5±2.4	-2.9±4.2	29.7±0.3	-0.2±1.2	0.9±0.3	84.2±1.1	17.7±0.5	44.8±1.3
	L	46.4±0.3	41.1±0.5	47.3±2.0	-6.8±4.6	38.6±0.8	27.4±0.8	1.7±0.6	73.6±0.3	13.3±0.2	28.3±0.3
	Avg.	55.2	57.9	49.0	-2.0	35.6	20.2	3.7	**81.6**	20.5	48.4
MPC-TSFM	S	65.0±0.5	80.9±0.2	50.3±2.0	2.9±10.0	39.0±1.1	34.6±1.3	5.2±0.1	49.8±0.1	30.8±0.9	72.0±0.4
	M	53.8±0.5	58.2±0.6	50.1±3.8	-5.7±8.3	**35.9**±0.5	31.4±0.3	0.8±0.4	79.6±1.0	17.6±0.4	45.8±1.7
	L	48.0±0.7	44.4±0.7	48.1±1.7	-4.6±5.5	37.5±0.4	21.1±1.0	1.6±0.5	71.9±0.9	13.4±0.2	28.6±0.1
	Avg.	55.6	61.1	49.5	-2.5	37.5	29.0	2.6	67.1	20.6	48.8
DiscreteCQL	S	47.1±1.1	21.2±1.3	52.7±3.6	18.8±14.4	**56.1**±3.9	66.0±3.9	19.7±0.7	55.7±0.6	20.5±1.4	43.3±4.2
	M	31.7±7.7	-1.2±0.9	54.2±3.1	-6.9±10.7	19.3±15.4	21.5±33.3	**4.8**±0.5	**85.2**±1.0	12.0±1.1	5.3±8.1
	L	38.7±10.4	4.6±1.8	51.5±1.3	**22.8**±1.0	60.8±0.0	41.0±0.0	**7.7**±3.6	**91.9**±1.8	11.4±1.7	-14.1±10.9
	Avg.	39.2	8.2	52.8	**11.6**	45.4	42.8	**10.7**	77.6	14.6	11.5
DoubleDQN	S	53.5±0.3	28.4±1.4	49.9±3.0	**19.2**±18.8	33.5±13.3	48.7±5.5	2.1±1.1	-103.4±30.1	8.3±4.7	-54.5±12.1
	M	47.5±0.1	10.2±0.9	48.0±3.5	-11.5±15.9	26.5±3.9	8.4±32.4	0.6±0.5	-21.7±9.6	6.0±9.9	-56.9±30.3
	L	45.4±2.0	5.0±2.1	48.9±1.9	-10.1±17.9	60.8±0.0	41.0±0.0	0.0±0.0	-152.4±29.0	1.3±2.0	-64.0±2.9
	Avg.	48.8	14.5	48.9	-0.8	40.3	32.7	0.9	-92.5	5.2	-58.5
Behavior Cloning	S	60.5±0.9	70.4±1.7	59.8±0.4	1.8±0.4	42.8±2.3	45.5±6.6	16.2±2.1	68.5±5.0	43.2±1.6	74.8±1.9
	M	50.8±1.0	39.3±2.6	46.8±2.4	-29.1±15.1	31.8±0.2	23.6±0.7	2.3±0.5	39.0±7.5	21.3±0.7	38.4±1.0
	L	50.6±0.1	41.7±0.5	**55.7**±2.2	5.0±3.7	38.5±1.1	20.4±0.8	2.4±0.8	55.9±2.8	20.1±0.1	38.1±1.8
	Avg.	54.0	50.5	54.1	-7.4	37.7	29.9	7.0	54.5	28.2	50.4
Decision Transformer	S	68.1±1.2	78.3±2.2	61.3±6.7	7.6±21.8	33.0±6.8	62.3±7.2	0.7±0.1	74.5±1.9	**60.4**±1.9	**88.7**±1.5
	M	64.5±2.6	64.9±4.2	49.3±3.5	-0.6±15.7	35.1±3.2	**68.4**±2.3	1.8±1.8	64.6±3.9	42.6±10.4	77.0±9.8
	L	68.2±4.9	69.7±8.2	52.6±5.1	-9.1±9.9	53.6±5.1	70.3±7.5	1.8±0.9	72.8±8.1	**53.8**±2.8	**83.9**±2.2
	Avg.	66.9	71.0	54.4	-0.7	40.6	67.0	1.4	70.6	**52.3**	83.2
Time-π_0	S	**77.2**±1.1	**83.5**±1.8	**64.1**±0.6	11.3±9.2	49.5±7.0	**68.7**±4.2	**21.7**±1.3	72.1±2.9	48.6±3.0	84.0±1.2
	M	**73.2**±2.1	**73.6**±3.4	**58.9**±2.3	**0.6**±4.5	35.7±7.9	64.4±9.3	4.2±1.0	68.7±2.3	**54.1**±3.6	**85.6**±1.3
	L	**74.7**±1.3	**78.5**±1.9	52.5±2.8	8.7±4.7	**64.0**±4.3	**73.5**±1.9	1.8±1.1	68.0±1.2	50.8±2.8	83.7±1.7
	Avg.	**75.1**	**78.5**	**58.5**	6.9	**49.7**	**68.9**	9.2	69.6	51.2	**84.4**

policy and the DP oracle, and negative values indicate performance worse than random. Cumulative rewards are computed separately for the single-chunk and rollout settings. The two metrics are complementary. Accuracy treats every step equally and counts only exact matches, whereas the reward depends on how costly each deviation is, so a few errors at critical steps can reduce the reward substantially, while near-optimal but different actions can still achieve a high reward. The reward function of each dataset is defined in Appendix B.

5.2 Open-loop evaluation

As shown in Table 2, Time-π_0 achieves competitive results across datasets, ranking first or second in horizon-averaged accuracy on all five datasets and in normalized reward on four of them. It also achieves higher normalized reward than MPC-TSFM in most settings, suggesting that plugging a TSFM into a forecast-then-optimize pipeline alone does not reproduce its gains. Time-π_0 is also robust to the horizon choice, with its normalized reward varying by at most 10.7% across horizons within each dataset. However, Time-π_0 underperforms MPC-native and DiscreteCQL on the Supply dataset, which is likely because the inventory level in this system is unbounded, whereas `Cassandra` discretizes order quantities into 300 classes. Interestingly, higher accuracy does not guarantee higher normalized reward. For example, on the `economy` dataset, Time-π_0 outperforms DiscreteCQL in accuracy by 5.7 %, but underperforms its normalized reward by 4.7 %. This is because accuracy weights every step equally, whereas the return can be dominated by a few steps with large price movements. This emphasizes the necessity of reporting both action accuracy and realized reward to evaluate the model in real-world decisions.

5.3 Roll-out evaluation

We further evaluate whether Time-π_0 maintains its performance under the rollout setting, which is more common in real-world deployment, where the state at the end of one round becomes the initial state of the next. As shown in Table 3, interestingly, the results are broadly consistent with the single round evaluation, despite longer rollouts and different initial states. Time-π_0 ranks first or second in horizon-averaged accuracy on all datasets and in normalized reward on four of them.

Table 3: Evaluation under three round rollouts, where each round's final state becomes the next round's initial state. Accuracy is close to the single round results obtained with the same checkpoint, despite longer rollouts and different initial states. Columns and highlighting follow Table 2.

Environment		battery		economy		telecom		supply		carbon	
Method	Horizon	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑
Heuristic	S	45.0	37.3	48.0	20.7	43.2	53.0	1.0	84.4	30.2	71.4
	M	47.7	33.1	44.4	2.2	22.3	53.1	0.0	**96.2**	33.3	73.0
	L	46.8	29.1	44.4	-10.1	45.0	65.6	0.2	88.5	35.4	69.1
	Avg.	46.5	33.2	45.6	4.3	36.8	57.2	0.4	89.7	33.0	71.2
MPC-native	S	66.9±0.1	77.3±0.3	51.4±2.1	**26.8**±8.2	37.5±0.7	31.6±1.5	9.8±0.6	**93.7**±0.4	30.0±1.2	73.0±0.7
	M	52.3±0.8	46.0±1.9	48.0±1.9	3.6±7.0	29.3±0.0	0.0±1.2	4.0±0.6	95.3±0.1	17.7±0.0	46.3±0.9
	L	47.0±0.8	37.2±1.7	49.9±3.3	-13.0±15.8	39.0±0.7	27.4±0.7	3.33±0.21	73.4±0.58	13.3±0.2	26.8±0.2
	Avg.	55.4	53.5	49.7	5.8	35.3	19.7	5.7	87.5	20.3	48.7
MPC-TSFM	S	65.5±0.2	76.2±1.1	50.1±2.7	24.7±10.9	38.6±1.4	33.4±1.8	15.6±1.9	70.1±0.6	30.2±1.2	72.9±0.3
	M	54.1±0.3	53.3±0.9	47.7±0.9	4.9±3.6	**36.6**±0.3	33.1±0.5	4.4±1.0	94.0±0.7	17.5±0.2	47.2±0.9
	L	48.8±0.4	40.7±1.1	48.4±0.7	-15.1±9.5	37.7±0.8	19.7±1.1	3.87±1.22	70.4±0.032	13.5±0.3	27.9±0.6
	Avg.	56.2	56.7	48.7	4.8	37.6	28.7	8.0	78.2	20.4	49.3
DiscreteCQL	S	39.6±2.8	9.8±1.0	52.4±3.4	21.9±14.0	**56.6**±5.3	65.0±5.1	17.5±0.5	83.9±0.4	16.3±0.8	39.5±4.6
	M	29.5±7.1	-2.0±2.5	55.6±3.8	22.2±15.3	20.0±16.1	22.4±37.1	4.4±1.4	95.6±0.2	10.4±1.1	3.8±6.7
	L	38.3±9.1	1.4±1.4	51.5±1.3	**13.9**±13.6	55.7±0.0	31.5±0.0	**7.74**±3.61	**96.0**±0.09	11.9±1.8	-12.7±12.5
	Avg.	35.8	3.1	53.2	**19.3**	44.1	39.6	9.9	**91.8**	12.9	10.2
DoubleDQN	S	46.7±0.1	9.4±0.8	49.8±2.0	25.9±11.6	31.8±13.6	46.1±6.4	0.6±0.5	-145.7±11.9	13.4±8.3	-35.5±29.0
	M	46.0±0.0	2.2±1.0	48.9±6.7	11.8±29.3	27.4±5.7	6.6±37.5	0.8±0.7	-58.6±42.8	8.1±13.8	-45.9±49.6
	L	44.5±1.9	1.5±1.3	48.5±1.3	-25.0±5.6	55.7±0.0	31.5±0.0	0.2±0	-1.52±0.2	4.7±7.9	-56.6±20.4
	Avg.	45.7	4.4	49.1	4.2	38.3	28.1	0.5	-68.6	8.7	-46.0
Behaviour Cloning	S	60.8±0.1	68.8±1.1	58.3±2.0	6.7±2.2	43.9±1.7	46.8±6.4	16.3±2.6	84.7±2.8	43.2±1.7	75.4±2.3
	M	51.2±0.7	39.1±2.4	51.0±1.3	-5.3±6.2	32.1±0.4	25.4±1.4	**16.7**±1.1	58.9±11.9	21.4±0.9	39.5±1.1
	L	50.5±0.4	39.9±0.7	**53.5**±1.3	-7.8±3.7	38.6±1.3	19.8±1.2	1.19±1.36	78.2±0.1	19.9±0.1	36.8±2.4
	Avg.	54.2	49.2	54.3	-2.1	38.2	30.6	**11.4**	73.9	28.1	50.6
Decision Transformer	S	68.9±2.1	76.1±1.6	60.9±6.8	13.2±10.5	33.4±7.4	61.2±7.9	0.3±0.5	90.0±0.5	**58.4**±1.4	**89.5**±1.6
	M	65.5±2.6	62.5±3.0	51.1±4.4	7.0±8.4	33.8±1.3	**69.4**±1.7	1.2±0.0	84.0±5.9	42.1±10.1	77.4±10.4
	L	68.6±5.3	69.5±8.8	52.6±5.1	-20.2±1.2	56.0±5.5	70.8±9.3	2.83±0.98	55±0.05	**54.7**±2.5	**84.4**±1.6
	Avg.	67.6	69.4	54.9	-0.0	41.1	67.1	1.4	76.3	51.7	83.8
Time-π_0	S	**76.9**±0.7	**83.6**±0.9	**62.7**±2.4	15.7±9.0	50.1±7.7	**69.3**±4.8	**20.6**±1.0	88.5±2.6	50.9±4.1	86.2±1.9
	M	**73.8**±2.2	**74.2**±3.6	**60.1**±3.3	**23.5**±3.4	35.1±7.2	65.4±8.5	5.2±1.2	67.6±2.2	**53.9**±3.8	**87.5**±1.3
	L	**75.0**±1.0	**78.5**±2.0	52.7±1.9	-0.9±4.0	**68.7**±3.9	**78.1**±0.6	3.08±0.55	69±0.03	51.0±3.0	83.6±1.7
	Avg.	**75.2**	**78.8**	**58.5**	12.8	**51.3**	**70.9**	9.6	75.0	**51.9**	**85.8**

The variation across rounds is also modest. For example, on the Battery dataset, accuracy across successive rounds changes from 75.2% to 75.8% and 76.1%. This stability is likely related to the state sampling strategy used to construct `Cassandra` (Section 4.1). In a rollout, the initial state of each later round is determined by the earlier decisions and may differ from the states in the demonstrations, a distribution shift known to cause compounding errors in imitation learning (Ross & Bagnell, 2010; Ross et al., 2011). Through state sampling, each training window is paired with multiple initial states, so Time-π_0 learns to act on the same exogenous history under diverse system states, which helps it handle the states reached during rollouts.

5.4 Battery Arbitrage: A Case Study

We examine a 72-hour battery arbitrage trajectory in Figure 3. At each hour, the model observes the day-ahead electricity price and the state of charge (SoC) of the battery, and decides whether to charge, hold, or discharge, earning the price difference between buying and selling energy. Since the remaining energy carries over to the next step, the policy must balance current trading opportunities against future price peaks. Time-π_0 achieves a final reward of 831.84, corresponding to 91.6% of the DP reward of 907.92 and 12.8% higher than MPC, and its actions match DP at 61 of the 72 steps. At step 33, for example, Time-π_0 sells at a local price peak, whereas MPC buys, likely because its forecast overestimates later prices. The reward curves also show that DP maximizes the cumulative reward at the end of the horizon rather than at every step. At step 65, Time-π_0 has accumulated 831.84 with an

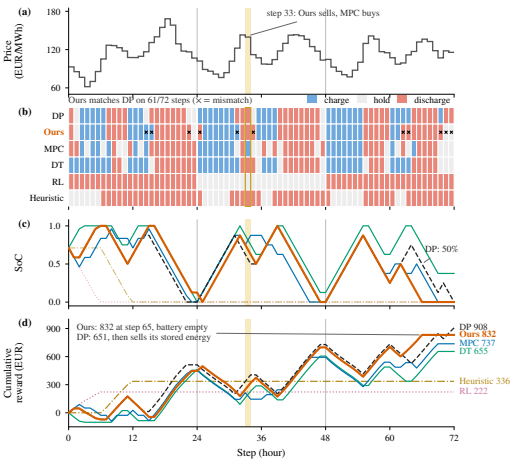

Figure 3: Battery arbitrage case study over 72 hourly steps. (a) Price. (b) Actions. (c) State of Charge (SoC). (d) Cumulative Reward.

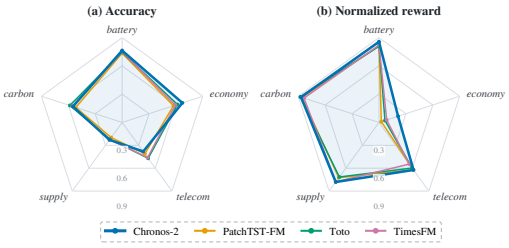

Figure 4: Performance of Time-π_0 with different TSFM backbones across five tasks, measured by accuracy and normalized reward. Higher values indicate better performance.

Table 4: Normalized reward of Time-π_0 and its ablated variants. Frozen TSFM keeps the pretrained TSFM fixed without LoRA, and w/o TSFM replaces it with a linear layer followed by layer normalization. Values in parentheses are the gains of the full model over each variant.

Dataset	Time-π_0	w/o forecasting loss	Frozen TSFM	w/o TSFM
`battery`	**82.5**	80.7 (+1.8)	82.6 (-0.1)	81.3 (+1.2)
`economy`	**14.6**	6.3 (+8.3)	9.7 (+4.9)	10.9 (+3.7)
`telecom`	**66.4**	45.3 (+21.1)	59.1 (+7.3)	40.8 (+25.6)
`supply`	**80.2**	78.1 (+2.1)	67.6 (+12.6)	76.5 (+3.7)
Mean	**60.9**	52.6 (+8.3)	54.8 (+6.1)	52.4 (+8.5)

empty battery, whereas DP has accumulated only 651.04 but retains 50% SoC, which it sells later to obtain the highest final reward. In contrast, DoubleDQN and the heuristic never charge and become inactive once the battery is completely empty. Charging yields a immediate negative reward that pays off only after later energy sales, which DoubleDQN fails to capture. The heuristic charges only when the price falls below its fixed lower threshold, which the prices in this window never reach.

5.5 Analysis of the TSFM Component

Compatibility with Different TSFM Backbones. We evaluate Time-π_0 with a range of TSFM backbones in the single-chunk setting, covering encoder-only models, Chronos-2 (Ansari et al., 2025) and PatchTST-FM (Wen et al., 2026), and decoder-only models, Toto (Khwaja et al., 2026) and TimesFM (Das et al., 2023). As shown in Figure 4, Time-π_0 accommodates all four backbones well, with normalized reward varying within about only 5 points across backbones on `battery`, `supply`, and `carbon`. Among them, Chronos-2 achieves the highest average across the three tasks normalized reward and is therefore used as the default backbone, although Toto attains slightly higher average accuracy, again showing that higher accuracy does not necessarily lead to higher reward. Overall, these results clearly suggest that Time-π_0 is not tied to a specific TSFM model and serves as a general architecture for bridging pretrained temporal knowledge and decision-making.

Ablation of TSFM Adaptation. Table 4 ablates the three components of Time-π_0, namely the auxiliary forecasting loss, the adaptation of the TSFM, and the pretrained TSFM itself. Removing the forecasting loss reduces the average normalized reward from 60.9 to 52.6, indicating that forecasting supervision helps adapt the TSFM to the target domain. Freezing the TSFM reduces it to 54.8, and the gain from adaptation is task-dependent. Removing the TSFM reduces the average normalized reward to 52.4, confirming the value of pretrained temporal representations for downstream decision-making task. The frozen TSFM ablation results suggest a possible mismatch between forecasting pretraining and downstream decision-making, which adaptation can reduce.

6 Conclusion

We formulate time series decision-making as action generation task conditioned on the exogenous time series and the current system state, and propose Time-π_0, a simple yet effective framework that adapts a pretrained TSFM with a lightweight flow-matching action expert to generate action chunks directly. On `Cassandra`, a testbed of five real-world operational systems supporting both single-chunk and rollout evaluations, Time-π_0 achieves strong performance compared with forecast-then-optimize and direct offline policy baselines. Our study has several limitations. First, Time-π_0 relies on hindsight-optimal labels, whose computation requires known system dynamics. Second, rollouts are limited to three rounds by the length of the available real-world time series. Third, Time-π_0 is less effective on the Supply dataset, which has a large action space and no upper bound on the inventory level. Overall, our findings lay the groundwork for expanding the utility of TSFMs, shifting their paradigm from general forecasting to actionable, real-world decision-making.

AI USE STATEMENT

In this work, we used generative AI tools for 'formulate mathematical claims; design or provide feedback on research methodology or experiments; implement methods; assist with translation; interpret results'. We have not used generative AI tools for 'help develop theoretical models or conceptual frameworks; clean and reformat dataset; support qualitative and thematic data analysis' and 'Generate synthetic data sets; provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses' are not applicable to this work. Additionally, we used generative AI tools for 'edit a research paper to improve readability; create or modify scientific figures or images; propose a title or keywords for a research paper'. We have reviewed all AI-assisted work. We checked the LLM-generated experiment result descriptions and LLM-generated code was verified and tested for correctness by the authors with double check. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine intelligence 15*, pp. 103–129, 1995.

Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. A multi-source dataset of urban life in the city of milan and the province of trentino. *Scientific data*, 2(1): 150055, 2015.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 4 edition, 2017.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024b.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

David B. Brown, James E. Smith, and Peng Sun. Information relaxations and duality in stochastic dynamic programs. *Operations Research*, 58(4):785–801, 2010.

CAISO. CAISO open access same-time information system (OASIS) and today's outlook: Emissions. Online grid data portal, 2026.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Meipo Dai, Qiyuan Zhuang, He-Yang Xu, Ying-Jie Shuai, Yijun Wang, Qi Dou, and Xiu-Shen Wei. Where should action generation begin? a learnable source prior for generative robot policies. *arXiv preprint arXiv:2606.17408*, 2026.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

ENTSO-E. Transparency platform. Online data platform, 2026.

Joren Gijsbrechts, Robert N Boute, Jan A Van Mieghem, and Dennis J Zhang. Can deep reinforcement learning improve inventory management? performance on lost sales, dual-sourcing, and multi-echelon problems. *Manufacturing & Service Operations Management*, 24(3):1349–1368, 2022.

Timothée Hornek, Youngsub Lee, Sergio Potenciano Menci, and Ivan Pavić. The value of battery energy storage in the continuous intraday market: Forecast vs. perfect foresight strategies. In *2025 IEEE Kiel PowerTech*, pp. 1–7. IEEE, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, et al. Toto 2.0: Time series forecasting enters the scaling era. *arXiv preprint arXiv:2605.20119*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Tomislav Kovačević, Andro Merćep, Stjepan Begušić, and Zvonko Kostanjčar. Optimal trend labeling in financial time series. *IEEE access*, 11:83822–83832, 2023.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Sang-gil Lee, Heeseung Kim, Chaehun Shin, Xu Tan, Chang Liu, Qi Meng, Tao Qin, Wei Chen, Sungroh Yoon, and Tie-Yan Liu. Priorgrad: Improving conditional denoising diffusion models with data-dependent adaptive prior. *arXiv preprint arXiv:2106.06406*, 2021.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Rui Liang, Yang Deng, Yaohui Liu, Dafang Zhao, Ozan Baris Mulayim, Ittetsu Taniguchi, and Dan Wang. Thermostill: Distilling time series foundation model into thermal dynamics model for hvac model predictive control. In *Proceedings of the 17th ACM International Conference on Future and Sustainable Energy Systems*, pp. 636–651, 2026.

Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 6555–6565, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022a.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022b.

Ximeng Liu, Shizhen Zhao, Yong Cui, and Xinbing Wang. Figret: Fine-grained robustness-enhanced traffic engineering. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 117–135, 2024.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International journal of forecasting*, 38(4):1346–1364, 2022.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Fatma Ezzahra Salem, Tijani Chahed, Eitan Altman, Azeddine Gati, and Zwi Altman. Optimal policies of advanced sleep modes for energy-efficient 5g networks. In *2019 IEEE 18th international symposium on network computing and applications (NCA)*, pp. 1–7. IEEE, 2019.

Qiuhong Shen, Shihua Zhang, Yue Liao, Qi Li, Zhenxiong Tan, Shizun Wang, Shuicheng Yan, and Xinchao Wang. World action models: A survey. *arXiv preprint arXiv:2606.20781*, 2026.

Sean R Sinclair, Felipe Vieira Frujeri, Ching-An Cheng, Luke Marshall, Hugo De Oliveira Barbalho, Jingling Li, Jennifer Neville, Ishai Menache, and Adith Swaminathan. Hindsight learning for mdps with exogenous inputs. In *International Conference on Machine Learning*, pp. 31877–31914. PMLR, 2023.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Yifan Wang. Thermoforce: A physics-structured interventional world model for building hvac control. *arXiv preprint arXiv:2607.03942*, 2026.

Yunshi Wen, Wesley M Gifford, Chandra Reddy, Lam M Nguyen, Jayant Kalagnanam, and Anak Agung Julius. Revisiting the generic transformer: Deconstructing a strong baseline for time series foundation models. *arXiv preprint arXiv:2602.06909*, 2026.

Yahoo Finance. Yahoo finance historical data: Walmart inc. (WMT). Online financial data service, 2026.

Kang Yang, Walid A Hanafy, Prashant Shenoy, and Mani Srivastava. Fm-cac: Carbon-aware control for battery-buffered edge ai via time-series foundation models. In *ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 1–7, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Xiaobin Zhang, Lefei Shen, Mouxiang Chen, Zhuo Li, Hongkai Li, Han Fu, Jianling Sun, Xiaoxue Ren, and Chenghao Liu. Cloudcons: A comprehensive end-to-end benchmark for cloud resource consolidation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 10278–10289, 2026.

Shao Zhuang, Lei Lijun, Peng Wang, Liang Zheng, Dong Wenxuan, and Tengfei Li. Action-conditioned chronos-2 hybrid for sample-efficient mpc dynamics modeling: Industrial so2 validation. *Electronics*, 15(17):3894, 2026.

Table of Contents

A Experimental Details **14**
A.1 Baseline Implementations 14
A.2 Training and Inference Details 15
A.3 Model Configurations 16
A.4 Effect of History and Execution Lengths 16
A.5 Hyperparameter Sensitivity 17

B Details of the `Cassandra` Testbed **17**
B.1 States, Actions, Rewards, and Constraints 17
B.2 Battery Energy Arbitrage (`battery`) 17
B.3 Stock Market Timing (`economy`) 18
B.4 Base-Station Sleep Control (`telecom`) 19
B.5 Inventory Replenishment (`supply`) 20
B.6 Carbon-Aware Inference Scheduling (`carbon`) 20

C Visualizations across Datasets **21**

A Experimental Details

Settings. We train Time-π_0 separately on each of the five operational systems. Within each system, all trajectories generated from the same window, including those with different sampled initial states, are assigned to the same split, so that no window appears in more than one split. We use AdamW with learning rates of 10^{-3} for the action expert and state encoder and 10^{-4} for the LoRA adapters, with weight decay of 0 for the LoRA adapters and 0.01 for all other trainable parameters. Both parameter groups use $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and a constant learning rate. We use a batch size of 16 without gradient accumulation or gradient clipping. Training runs for up to 30 epochs with validation after each epoch, and we stop early if the validation loss does not improve for eight epochs and select the checkpoint with the lowest validation loss.

Implementation. We implement Time-π_0 with Python 3.10, PyTorch 2.5.1 (CUDA 12.1), Lightning 2.6.5, Transformers 4.56.0, and chronos-forecasting 2.2.2. Experiments are run on a workstation with two NVIDIA RTX 6000 Ada GPUs (48 GB each), with each training run on a single GPU. Training and validation use BF16 mixed precision, whereas testing uses FP32, and both validation and testing use a batch size of 8.

A.1 Baseline Implementations

Heuristic. For `battery`, the policy charges when the current price is below the 30th percentile of the training prices, discharges when it is above the 70th percentile, and holds otherwise. For `economy`, the thresholds are the 30th and 70th percentiles of the three-day price change. The policy invests when the current three-day price change exceeds the upper threshold, moves to cash when it falls below the lower threshold, and otherwise keeps its current position, including when the history is too short. For `telecom`, the policy selects sleep mode when the current exogenous demand is below 8% and the full-power mode otherwise. For `supply`, the policy orders the mean demand of the observation window every day. For `carbon`, a carbon intensity above 240 gCO_2/kWh triggers the low-accuracy mode without charging, powered by the battery if it has sufficient energy and by the grid otherwise. An intensity below 112 gCO_2/kWh triggers the high-accuracy mode with grid power and charging. Otherwise, the policy selects the middle mode, charges when the battery is less than half full, and uses the battery when the intensity exceeds 176 gCO_2/kWh and sufficient energy is available, and the grid otherwise.

Behavior Cloning. We use a conditional flow-matching policy. Each discrete action is decomposed into mixed-radix components (e.g., the mode, charging flag, and power source in `carbon`, and a single component elsewhere), and each component is mapped to $[-1, 1]$ by

$$y_{t,j} = \frac{2d_{t,j}}{b_j - 1} - 1, \tag{9}$$

where $d_{t,j}$ is the integer value of component j at step t and b_j is its number of possible values. The history and the state embedding are concatenated and passed through a two-layer MLP of width 256, and the flow time is encoded by 128 sinusoidal features followed by a width-256 MLP. The velocity network takes the flattened noisy action chunk, the time embedding, and the conditioning vector, and applies three width-256 hidden layers with SiLU activations and dropout 0.1. The model is trained with the flow-matching MSE loss. At inference, it integrates the velocity field with ten Euler steps from $\tau = 0$ to $\tau = 1$, and the outputs are quantized back to discrete actions.

Decision Transformer. At each step, the input concatenates the history, the initial-state representation, and a time-position encoding, where states are encoded by min–max scaling or one-hot encoding. The model is a causal Transformer with three layers, hidden size 128, one attention head, feed-forward width 512, and dropout 0.1. The action head has one tanh output per action component and is trained with MSE on the same normalized components as BC, using teacher forcing with oracle actions. At inference, actions are generated autoregressively, with each output quantized to a legal action and fed back as input.

DoubleDQN and DiscreteCQL. We implement DoubleDQN (Van Hasselt et al., 2016) with a Huber loss, and DiscreteCQL (Kumar et al., 2020) by adding the conservative penalty with coefficient λ_{CQL} to the DoubleDQN loss. The Q-network takes as input the standardized history, the state encoding, the standardized exogenous value of the previous step, and the normalized time step. The last transition of each window is treated as terminal, and the discount factor is $\gamma = 1$. Both methods are trained with Adam at a learning rate of 6.25×10^{-5}, and checkpoints are selected by the TD error on the validation set. To match the single-chunk setting, the Q-network predicts all actions of a window from the initial state s_0, with the exogenous input set to zero and only the time feature changing across positions, and selects the action with the highest Q-value at each position. We tune $\lambda_{\mathrm{CQL}} \in \{0.5, 1, 5, 10\}$ on the short horizon by the validation normalized reward, obtaining 0.5 for `battery` and `economy`, 5 for `telecom`, 1 for `supply`, and 5 for `carbon`.

MPC-Native and MPC-TSFM. An ensemble of five independently initialized MLPs predicts the next state and reward conditioned on the current state, initial state, action, exogenous input, and temporal features. Categorical next-state predictions are trained using cross-entropy loss. MPC-native forecasts future exogenous inputs by repeating the last seasonal block of the observation history, whereas MPC-TSFM uses Chronos2 to generate these forecasts.

A.2 Training and Inference Details

For each training example, we sample a single flow time τ shared across all positions of the action chunk, and the flow-matching MSE is averaged over the n positions and d_{code} dimensions. The state encoder produces a token for the prior network, which is excluded from the cross-attention context of the DiT.

Classifier-Free Guidance. During training, we replace the temporal context $\mathbf{C}_t$ with a learned null token $\varnothing$ with probability 0.1 per example, and this dropout is disabled when computing the validation loss. At inference, we combine the conditional and unconditional velocities with a guidance scale w:

$$\mathbf{v}_w(\mathbf{z}, \tau, \mathbf{C}_t) = \mathbf{v}_\theta(\mathbf{z}, \tau, \varnothing) + w\big[\mathbf{v}_\theta(\mathbf{z}, \tau, \mathbf{C}_t) - \mathbf{v}_\theta(\mathbf{z}, \tau, \varnothing)\big]. \tag{10}$$

Both branches share the same state-conditioned prior, and only the conditional branch is evaluated when $w = 1$.

Table 5: Shared configuration of Time-π_0. Dataset dimensions are determined from the generated trajectory metadata.

Component	Configuration
Backbone and Adaptation	
Feature dimension	768
LoRA rank / scaling	$r = 16$, $\alpha = 32$
LoRA dropout	0
Action Representation and Decoding	
Latent dimension d_a	128
Codebook	Fixed
Decoder hidden dimension	128
Classification	One $\lvert\mathcal{A}\rvert$-way categorical distribution per step
Diffusion Transformer (DiT)	
Hidden dimension / blocks	192 / 4
Attention heads / MLP ratio	6 / 4
Attention dropout	0.1
Prior	
Architecture	LayerNorm–Linear–SiLU–Linear
Hidden / output dimension	128 / 128
Gaussian scale	Fixed at 1
Conditioning and Losses	
Observation-condition dropout	0.1
Forecast-loss weight / normalizer	0.1 / 8
FM normalizer	2
Normalized CE weight	1

Numerical Integration and Action Selection. Starting from $\mathbf{z}^{(0)} = \mathbf{z}_0$, we integrate the guided velocity field with N_{step} Euler steps,

$$\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \frac{1}{N_{\text{step}}} \mathbf{v}_w \left(\mathbf{z}^{(i)}, \frac{i}{N_{\text{step}}}, \mathbf{C}_t \right), \qquad i = 0, \ldots, N_{\text{step}} - 1, \tag{11}$$

updating all positions jointly. The final latent is decoded by Dec_θ, and the action with the largest logit is selected at each position.

A.3 Model Configurations

As shown in Table 5, we use the pretrained Chronos-2 checkpoint with 768-dimensional encoder representations. For each variate, the encoder outputs at the historical patches, together with a summary token of the series, form the temporal context $\mathbf{C}_t$ for the action expert, whereas the outputs at the future positions are used only by the forecasting head. LoRA is applied to the query, key, value, and output projections of the temporal and group attention modules of the backbone. Each DiT block contains time-modulated self-attention, cross-attention to the projected backbone tokens, and a feed-forward network. The action positions use learned embeddings, and the adaptive normalization gates and the final velocity projection are initialized to zero. Continuous system states are encoded by a 128-dimensional sinusoidal encoding followed by an MLP that maps them to the backbone feature dimension.

A.4 Effect of History and Execution Lengths

As shown in Tables 7 and Tables 8, Time-π_0 shows strong performance across horizon settings in both open-loop and closed-loop evaluations. On `battery`, it achieves the best average accuracy and normalized reward, outperforming Decision Transformer in average reward by 9.9 and 10.4 percentage points, respectively. On `carbon`, although Decision Transformer achieves the highest average scores, Time-π_0 trails by only 0.6 and 0.7 percentage points in normalized reward and achieves the best reward under both s2l and l2s settings. These results support the adaptability of Time-π_0 across the evaluated horizons, particularly when transferring from shorter to longer horizons.

Table 6: Environment-specific inference settings and evaluation chain counts per training seed. Single-round and three-round evaluations use different chain constructions.

	Inference Settings		Evaluation Chains	
Environment	Euler steps K	CFG scale w	1-round	3-round
battery	2	1.0	111	24
carbon	1	3.0	110	36
economy	2	1.5	280	5
supply	10	1.0	301	5
telecom	2	3.0	52	8

Table 7: Open-loop evaluation on battery and carbon. Accuracy and normalized reward (%) of Time-π_0. Bold and underlined values indicate the best and second best results among the listed methods, respectively. Avg. denotes the arithmetic mean across the three settings.

Dataset	Horizon	Heuristic		MPC-native		MPC-TSFM		DiscreteCQL		DoubleDQN		BehaviorClone		DecisionTransformer		**Time-Pai0**	
		Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑
battery	s2m	48.9	30.4	53.0	55.1±0.2	52.3	54.9±0.3	37.9	5.0±2.5	45.5	10.2±0.6	51.5	45.1±2.1	64.7	68.1±6.1	**76.1**	**78.7**±0.6
	s2l	48.0	29.4	46.3	38.6±0.3	45.4	37.3±0.8	33.1	3.1±2.3	46.6	4.4±0.8	49.8	38.6±1.5	63.8	59.9±9.5	**76.0**	**77.6**±1.4
	l2s	45.0	37.3	68.0	84.9±0.4	70.1	**89.0**±0.8	52.7	27.1±1.6	53.6	28.6±0.0	62.7	71.6±4.2	71.5	79.5±4.4	**75.9**	81.1±2.0
	Avg.	47.3	32.4	55.8	59.5	55.9	60.4	41.2	11.7	48.6	14.4	54.7	51.8	66.7	69.2	**76.0**	**79.1**
carbon	s2m	34.8	71.0	17.5	45.8±1.0	17.9	45.9±0.8	20.1	41.4±6.5	3.1	-63.8±10.6	22.1	39.5±2.7	**55.0**	**88.7**±0.8	48.9	79.1±7.3
	s2l	36.1	70.8	13.7	29.7±0.5	13.7	29.5±0.6	18.6	23.7±6.9	0.9	-67.0±7.5	20.2	37.0±1.9	52.8	83.2±2.9	**56.3**	**87.1**±0.2
	l2s	33.3	70.9	29.0	70.1±0.9	30.3	71.9±2.0	20.7	42.1±7.8	1.7	-64.1±8.7	30.1	58.8±4.2	**52.0**	81.2±3.8	51.4	**85.3**±2.7
	Avg.	34.7	70.9	20.1	48.5	20.6	49.1	19.8	35.7	1.9	-65.0	24.1	45.1	**53.3**	**84.4**	52.2	83.8

A.5 Hyperparameter Sensitivity

Inference-Time Parameters. The inference study varies Euler steps over $K \in \{1, 2, 5, 10, 20\}$ and guidance scales over $w \in \{1, 1.5, 2, 3\}$ using the seed-0 checkpoint and validation-window closed-loop return. These parameters do not require retraining. The recorded protocol first selects K and then rechecks w at the chosen K; it is not an exhaustive joint grid search. The final settings in Table 6 are shared across the three training seeds and fixed before the corresponding test evaluation.

Training-Time Parameters. The development records also describe a seed-0 pilot that compares a larger training budget, constant versus cosine learning-rate schedules, a reduced action-module learning rate (3×10^{-4}), an increased adapter learning rate (3×10^{-4}), and forecasting weights $\lambda_{\mathrm{FC}} \in \{0, 0.1, 1\}$. Several variants are defined relative to the cosine-schedule pilot, rather than to the final constant-rate configuration. The retained recipe uses 30 epochs, patience eight, constant learning rates, and $\lambda_{\mathrm{FC}} = 0.1$. These development comparisons do not establish a multi-seed sensitivity result or the universal optimality of the selected hyperparameters.

B Details of the Cassandra Testbed

B.1 States, Actions, Rewards, and Constraints

We consider five finite-horizon, discrete-time decision-making environments driven by real world exogenous sequences. The exogenous input is unaffected by the agent's actions, whereas the endogenous state evolves according to the simulator dynamics. For each environment, we specify the data and decision resolution, state variables, actions, transitions, rewards, and constraints, including initial and terminal conditions.

B.2 Battery Energy Arbitrage (battery)

Data Source and Temporal Resolution. The exogenous input ρ_t is the hourly day-ahead electricity price in the French bidding zone, in EUR/MWh. The source contains 17,544 hourly observations from 2023–2024 and is used without normalization or smoothing; negative prices are retained. Each decision step has duration $\Delta t = 1$ hour.

Table 8: Close-loop evaluation on `battery` and `carbon`. Accuracy and normalized reward (%) of Time-π_0. Bold and underlined values indicate the best and second best results among the listed methods, respectively. Avg. denotes the arithmetic mean across the three settings.

Dataset	Horizon	Heuristic		MPC-native		MPC-TSFM		DiscreteCQL		DiscreteDQN		BehaviorClone		DecisionTransformer		**Time-Pai0**	
		Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑	Acc↑	NR↑
`battery`	s2m	47.7	31.9	53.0	51.8±0.5	52.1	51.4±0.9	38.8	5.0±1.1	46.3	7.1±0.5	51.1	46.0±1.4	63.5	67.5±4.2	**75.7**	**79.1**±1.6
	s2l	47.9	29.0	46.2	34.6±0.6	45.7	32.8±1.4	20.8	1.8±1.2	46.3	1.2±0.6	50.3	36.9±0.7	67.0	58.0±8.5	**76.7**	**77.8**±1.3
	l2s	45.1	36.8	68.4	82.0±0.3	70.7	**85.9**±0.3	47.5	13.8±1.1	47.5	13.2±0.0	60.5	71.5±6.3	**74.8**	79.4±0.7	72.3	79.3±1.2
	Avg.	46.9	32.6	55.9	56.1	56.2	56.7	35.7	6.9	46.7	7.2	54.0	51.5	68.4	68.3	**74.9**	**78.7**
`carbon`	s2m	34.6	71.5	17.3	47.3±0.2	17.7	47.2±0.3	20.6	42.2±8.5	5.4	-41.7±30.3	22.9	40.5±3.2	**54.2**	**89.6**±0.8	35.2	79.9±7.4
	s2l	35.5	70.2	14.2	30.5±0.4	14.0	30.2±0.6	18.7	28.4±7.0	13.3	-58.6±21.7	20.2	36.9±1.9	57.4	83.4±4.1	**60.0**	**87.5**±0.1
	l2s	33.3	69.8	29.6	70.5±1.7	31.2	72.2±0.9	18.2	42.8±7.5	5.4	-64.6±9.2	34.5	60.7±5.3	54.6	83.5±2.5	**57.2**	**87.0**±2.7
	Avg.	34.5	70.5	20.4	49.4	21.0	49.9	19.2	37.8	8.0	-55.0	25.9	46.0	**55.4**	**85.5**	50.8	84.8

States. The endogenous state is stored energy $E_t \in [0, E_{\max}]$, where $E_{\max} = 4.0$ MWh. Its normalized state of charge is $u_t = E_t / E_{\max}$. The current price ρ_t is the exogenous variable relevant to the step's cash flow.

Actions. The action identifiers 0, 1, and 2 denote charge, hold, and discharge, respectively. These three actions correspond to commanded powers

$$P_t^{\mathrm{cmd}} \in \{+P_{\max}, 0, -P_{\max}\}, \qquad P_{\max} = 0.5 \text{ MW}. \tag{12}$$

State Transitions. Storage is lossless, and commands are clipped at the energy bounds:

$$E_{t+1} = \mathrm{clip}(E_t + P_t^{\mathrm{cmd}} \Delta t, 0, E_{\max}), \tag{13}$$

$$\Delta E_t = E_{t+1} - E_t. \tag{14}$$

Here ΔE_t is the actual energy purchased from the grid (negative when energy is sold), in MWh.

Rewards. The step reward is trading cash flow in EUR:

$$r_t = -\rho_t \Delta E_t, \qquad \Phi_T = 0. \tag{15}$$

Charging at a negative price therefore yields a positive reward.

Constraints and Boundary Conditions. The hard bounds are $0 \le E_t \le 4.0$ MWh and $|P_t^{\mathrm{cmd}}| \le 0.5$ MW. All three commands remain available at the boundaries; clipping reduces the actual trade, and a fully saturated command trades no energy and incurs no additional penalty. Initial state of charge is enumerated over $u_0 \in \{0.01, 0.02, \ldots, 1.00\}$. Initial energy is not charged to the episode, and remaining energy has no terminal settlement. Thus, the initial energy endowment affects the interpretation of returns across initial states.

B.3 Stock Market Timing (`economy`)

Data Source and Temporal Resolution. The exogenous input p_t is Walmart's split- and dividend-adjusted daily closing price in USD. The series contains 753 trading days from 2022–2024. A trading action is executed at the corresponding decision-day close.

States. The endogenous state is $z_t = (n_t, B_t)$, consisting of the non-negative real-valued share holding n_t and cash balance B_t. The position indicator is derived as $\mathbf{1}\{n_t > 0\}$ rather than stored as an independent variable. The initial wealth $W_0 = n_0 p_0$ is retained as the reference for terminal profit and loss. Share count alone is insufficient to specify the simulator state because a purchase depends on cash.

Actions. The action space is $\mathcal{A} = \{0, 1\}$: 0 requests liquidation into cash, and 1 requests investment into shares. Actions specify target positions, so repeated same-side commands trade nothing. Fractional shares are allowed.

State Transitions. Let (n_t, B_t) denote the state before trading at the close p_t. Then

$$(n_{t+1}, B_{t+1}) = \begin{cases} (0, B_t + n_t p_t), & a_t = 0,\ n_t > 0, \\ (B_t/p_t, 0), & a_t = 1,\ n_t = 0, \\ (n_t, B_t), & \text{otherwise.} \end{cases} \tag{16}$$

The purchase uses all available cash without whole-share rounding. There are no transaction fees or slippage.

Rewards. The episode return is terminal profit and loss in USD:

$$R = B_T + n_T p_{T-1} - n_0 p_0. \tag{17}$$

An equivalent terminal-only reward representation is $r_t = 0$ for each decision step and $\Phi_T = R$. Remaining shares are marked to the final decision-day close. Buy-and-hold therefore earns $n_0(p_{T-1} - p_0)$. Under fractional-share trading, returns for a fixed price and action sequence scale linearly with the initial holding, up to numerical precision.

Constraints and Boundary Conditions. Short selling and borrowing are not allowed: $n_t \geq 0$ and $B_t \geq 0$. Episodes start fully invested with $B_0 = 0$ and one of 280 initial holdings,

$$n_0^{(i)} = \mathrm{round}_4\left(1 + \frac{27i}{279}\right), \qquad i = 0, \ldots, 279, \tag{18}$$

where round_4 rounds to four decimal places. The configured value 40 is a declared data-dependent share-count bound; the portfolio transition does not enforce a 40-share cap. There is no terminal adjustment beyond the final portfolio valuation.

B.4 Base-Station Sleep Control (telecom)

Data Source and Temporal Resolution. The exogenous input D_t is Internet traffic from cell 5262 of the Telecom Italia Milan dataset, normalized by the series peak to lie in $[0, 1]$. The series contains 8,928 ten-minute slots from November 2013 through January 2014.

States. The endogenous state immediately before the step is a switching-machine configuration $z_t = (\ell_t, v_t, g_t)$, where $\ell_t \in \{0, 1, 2, 3\}$ is a capped progress counter, v_t is the stored reference mode, and g_t is the current target mode. There are four mode choices and four progress values, giving $4 \times 4 \times 4 = 64$ encoded configurations; these are not all reachable, and stable states satisfy $\ell_t = 3$ and $v_t = g_t$. We use D_t for demand to avoid conflating it with the switch-progress variable.

Actions. Action identifiers $0, 1, 2, 3$ request modes $A1, A2, A3, A4$, respectively. Their service capacities and holding-power values are

$$\mathrm{cap}(A1{:}A4) = \left(0, \tfrac{1}{3}, \tfrac{2}{3}, 1\right), \qquad \mathrm{power}(A1{:}A4) = (0.08, 0.60, 0.80, 1.00). \tag{19}$$

Mode $A1$ is sleep and $A4$ is full power.

State Transitions. The simulator uses a progress threshold of three slots and updates the switching state before computing the current slot's service. Define the switch indicator $J_t = \mathbf{1}\{a_t \neq g_t\}$. The provisional update is

$$(\ell^+, g^+, \widehat{v}) = \begin{cases} (1, a_t, g_t), & J_t = 1, \\ (\min\{\ell_t + 1, 3\}, g_t, v_t), & J_t = 0. \end{cases} \tag{20}$$

The serving mode and next state are

$$v_t^{\mathrm{serve}} = \begin{cases} \widehat{v}, & \ell^+ < 3, \\ g^+, & \ell^+ = 3, \end{cases} \tag{21}$$

$$z_{t+1} = (\ell^+, v_t^{\mathrm{serve}}, g^+). \tag{22}$$

Thus, a new command issued at slot t and maintained thereafter begins serving at slot $t+2$, the third slot counting the command slot. A different command during preparation restarts the counter at one and uses the old target as the reference serving mode, even if that target had not yet become stable. The reference mode therefore need not be a previously stabilized mode after an interrupted switch.

Rewards. The step reward is the negative operating cost:

$$r_t = -\left[\text{power}(v_t^{\text{serve}}) + 4.0\max\{0, D_t - \text{cap}(v_t^{\text{serve}})\} + 0.25J_t\right]. \tag{23}$$

Constraints and Boundary Conditions. Mode selection is discrete and the preparation delay restricts when a requested capacity becomes available. Demand exceeding the serving capacity is allowed and incurs a soft unmet-demand penalty; there is no stated hard requirement to serve all traffic. Each replay window is initialized once in each stable mode, with $\ell_0 = 3$ and $v_0 = g_0 \in \{A1, A2, A3, A4\}$. The episode has zero terminal adjustment, $\Phi_T = 0$: a switch still in progress receives no additional completion cost or value beyond the decision horizon.

B.5 Inventory Replenishment (SUPPLY)

Data Source and Temporal Resolution. The exogenous input D_t is daily unit sales for FOODS_3_282 in the M5 dataset, aggregated across California stores and treated as demand. Removing the pre-launch all-zero prefix yields 1,029 days, with a maximum daily demand of 298 units. The action is selected from the state before that day's receipt and demand. The simulator then processes receipt of the previous order, demand fulfillment, cost accrual, and insertion of the selected order into the pipeline.

States. Let H_t denote inventory carried into day t before receipt and Q_t the outstanding order arriving that day. The endogenous state is (H_t, Q_t). Available inventory after receipt is $A_t = H_t + Q_t$. Including the pipeline distinguishes stock already on hand from stock committed for delivery.

Actions. The agent places an integer order

$$q_t \in \mathcal{A} = \{0, 1, \ldots, 299\}. \tag{24}$$

The action identifier equals the order quantity. Non-integer policy outputs are rounded and clipped to this grid before execution.

State Transitions. The deterministic lead time is one day. Lost demand is not backlogged, so

$$A_t = H_t + Q_t, \tag{25}$$
$$U_t = \max\{0, D_t - A_t\}, \tag{26}$$
$$H_{t+1} = \max\{0, A_t - D_t\}, \tag{27}$$
$$Q_{t+1} = q_t. \tag{28}$$

Here U_t denotes lost sales and H_{t+1} is inventory remaining after day-t demand. The previously selected order enters the pipeline after demand is served and cannot affect day-t fulfillment.

Rewards. The reward is the negative sum of holding and lost-sales costs:

$$r_t = -\left(0.01H_{t+1} + 0.02U_t\right), \qquad \Phi_T = 0. \tag{29}$$

There is no fixed ordering cost, and the stated step-cost formula includes no per-unit procurement charge.

Constraints and Boundary Conditions. Orders are bounded non-negative integers, inventory cannot become negative, and orders arrive after one day. Stockouts are allowed and penalized rather than forbidden. Initial conditions are $H_0 \in \{0, 1, \ldots, 300\}$ and $Q_0 = 0$. The upper endpoint 300 specifies the initial-state grid; it does not establish a warehouse-capacity constraint on subsequent stock. There is no warehouse-capacity cap, terminal salvage value, or terminal pipeline charge. An order placed on the final day arrives outside the episode and does not affect its return.

B.6 Carbon-Aware Inference Scheduling (CARBON)

Data Source and Temporal Resolution. The exogenous input c_t is CAISO grid carbon intensity in gCO_2/kWh. Fifteen-minute samples are averaged into hourly bins, and bins with missing hourly means are removed. The supplied series contains 26,286 observations over 2023–2025. Each simulator step represents one hour and executes exactly one inference job.

States. The endogenous state is battery energy $E_t \in [0, 15]$ MWh. The initial energy E_0 is retained as an episode parameter for terminal settlement. Emissions are charged when grid energy is drawn, so tracking the carbon intensity of stored energy is unnecessary.

Actions. An action consists of an inference mode $m_t \in \{0, 1, 2\}$, a charging flag $b_t \in \{0, 1\}$, and a source flag $g_t \in \{0, 1\}$, where 0 requests battery power and 1 requests grid power. Its scalar identifier is

$$a_t = 4m_t + 2b_t + g_t \in \{0, \dots, 11\}. \tag{30}$$

The mode accuracy α_m and energy requirement x_m are

$$(\alpha_m, x_m) = \begin{cases} (0.40, 0.5\ \text{MWh}), & m = 0 \quad \text{(low)}, \\ (0.75, 1.5\ \text{MWh}), & m = 1 \quad \text{(mid)}, \\ (0.95, 3.0\ \text{MWh}), & m = 2 \quad \text{(high)}. \end{cases} \tag{31}$$

State Transitions. Charging precedes inference. A 3 MWh charging request is accepted only if the entire amount fits before inference; otherwise no charging energy is drawn. Define

$$C_t = 3b_t\, \mathbf{1}\{E_t + 3 \le 15\}, \tag{32}$$

$$E_t^+ = E_t + C_t, \tag{33}$$

$$u_t = \mathbf{1}\{g_t = 0\}\, \mathbf{1}\{E_t^+ \ge x_{m_t}\}. \tag{34}$$

Here u_t indicates successful battery-powered inference. The next energy and total grid draw are

$$E_{t+1} = E_t^+ - u_t x_{m_t}, \tag{35}$$

$$G_t = C_t + (1 - u_t) x_{m_t}. \tag{36}$$

If a battery-power request cannot be satisfied after charging, the entire inference job automatically falls back to grid power. No partial discharge occurs, and the selected mode's accuracy α_{m_t} is retained. Energy charged in the current step can be used immediately for that step's inference.

Rewards. For energy x in MWh and intensity c in gCO_2/kWh, define emissions in tonnes of CO_2 as

$$\mathrm{em}(x, c) = \frac{xc}{1000}. \tag{37}$$

Using the default weights $w_c = 1.5$ and $w_a = 1.0$, the step reward and terminal adjustment are

$$r_t = -\left[1.5\, \mathrm{em}(G_t, c_t) + (0.95 - \alpha_{m_t})\right], \tag{38}$$

$$\Phi_T = -1.5\, \mathrm{em}(E_0 - E_T, c_{\mathrm{ref}}), \tag{39}$$

$$c_{\mathrm{ref}} = 182.489330\ \mathrm{gCO_2/kWh}. \tag{40}$$

The reference intensity is a fixed configuration constant derived from the full-series mean. It is shared across windows, not recomputed from the current decision window. Grid emissions are billed once at draw time; successful battery discharge has no additional emissions charge. Automatic grid fallback incurs its actual grid emissions and no separate failure penalty.

Constraints and Boundary Conditions. The hard storage bounds are $0 \le E_t \le 15$ MWh. Charging is lossless and all-or-nothing, and the 12 commands remain available at every state because unavailable charging is skipped and unavailable battery power is replaced by grid power. Exactly one job is completed per step; no deferral action or hard minimum accuracy requirement is imposed. Accuracy loss is a soft cost. Initial state of charge is $E_0/15 \in \{0.01, 0.02, \dots, 1.00\}$. Terminal settlement charges net depletion and credits net accumulation relative to E_0, without requiring $E_T = E_0$. Different initial states can still yield different attainable returns because their available energy and charging headroom differ.

C Visualizations across Datasets

Figure 5 shows example decision sequences from the five environments. Each row corresponds to one environment, and the three columns show short, medium, and long decision horizons, respectively. For each example, we report the oracle decisions computed by a policy that has access to the entire future time series. The figure therefore provides a reference for understanding what effective decisions look like in each environment and how the decision sequence changes as the horizon becomes longer. Detailed descriptions of the five datasets are provided below.

Battery. The Battery environment models electricity and price arbitrage through three actions: charging, holding, and discharging. In the short horizon example, the electricity price initially rises from approximately 50 to $175\,\mathrm{EUR/MWh}$. The policy charges during the relatively low-price portion at the beginning of the decision window and increasingly selects discharging actions around the subsequent price peaks, producing a total reward of 328.82 EUR. The medium-horizon trajectory contains several additional price valleys and peaks. The policy repeatedly switches between charging at lower prices and discharging at higher prices, increasing the total reward to 972.81 EUR. This behavior becomes more pronounced in the long horizon example, where the policy reacts to multiple price cycles over approximately one week and obtains a total reward of 2361.13 EUR. These trajectories show that a longer horizon requires the policy to coordinate a substantially larger number of charge-discharge cycles.

Carbon. The Carbon environment combines a time-varying carbon-intensity signal with the state of charge (SoC) of an energy-storage system. In the short-horizon example, carbon intensity rises to about $250\,\mathrm{gCO_2/kWh}$ before dropping to roughly $130\,\mathrm{gCO_2/kWh}$. Over the same period, the storage system undergoes several charging and discharging phases, with its SoC reaching approximately 0.9. The medium- and long-horizon examples show a recurring pattern of high- and low-carbon periods. The SoC follows a similar cyclical pattern, moving between nearly empty and nearly full, with most charging occurring during low-carbon periods or near their onset. The total rewards for the short, medium, and long horizons are -13.81, -42.52, and -103.49, respectively. Because these rewards are accumulated over different numbers of time steps, their absolute values are not directly comparable across horizons.

Economy. The Economy environment involves a binary portfolio choice between holding a risky asset (*in*) and holding cash (*out*). In the short-horizon example, the policy enters the market near a local price minimum and exits after the price rebounds. The portfolio value increases from approximately 45.66 to 46.31 USD, resulting in a reward of 0.65 USD. In the medium-horizon example, the asset price rises from about 42.9 to 44.5 USD before declining. The policy holds the asset for most of the upward movement and switches to cash during several downward movements. As a result, the portfolio value rises from 43.30 to approximately 45.00 USD. In the long-horizon example, the price follows several upward and downward trends, with an overall increase from approximately 42.4 to 46.6 USD. By switching between the asset and cash, the policy raises the portfolio value to about 52 USD, yielding a total reward of 9.52 USD.

Supply. The Supply environment considers inventory replenishment under changing demand. The upper panel shows the observed and future demand, the middle panel shows the order quantities, and the lower panel shows the on-hand inventory and shortages. All three examples start with no inventory and therefore experience a shortage at the beginning of the decision period. The initial shortages are approximately 84, 150, and 120 units for the short, medium, and long horizons, respectively. After the first step, the order quantities generally follow the demand pattern, including spikes above 200 units in the medium- and long-horizon examples. Inventory remains close to zero, with no sustained shortages after the initial step. This pattern suggests that the policy meets demand while holding little excess stock. The total costs are 1.68, 3.04, and 2.42 for the three horizons, respectively.

Telecom. The Telecom environment considers segment-routing traffic engineering across high-, medium-, and low-demand origin–destination tiers. The policy chooses between the shortest path and several alternative paths. Network congestion is measured by the maximum link utilization (MLU), where a value above one indicates that at least one link is overloaded. In the short-horizon example, the MLU starts at approximately 1.21 and falls below one as demand decreases. Only one path change occurs during this period. The medium-horizon example is more congested: the MLU reaches approximately 1.47 and remains above one for much of the decision period before eventually dropping below the threshold. Its mean MLU is 1.216. In the long-horizon example, the MLU initially peaks near 1.58 and later stays close to one, with a mean of 1.076. The policy uses the shortest path for the high-demand tier and an alternative path for the medium-demand tier. The low-demand tier is temporarily rerouted during the congested period and later returned to the shortest path. This produces three path changes, reflecting a balance between reducing congestion, limiting detours, and avoiding frequent routing changes.



(a) Short horizon (b) Medium horizon (c) Long horizon

(d) Short horizon (e) Medium horizon (f) Long horizon

(g) Short horizon (h) Medium horizon (i) Long horizon

(j) Short horizon (k) Medium horizon (l) Long horizon

(m) Short horizon (n) Medium horizon (o) Long horizon

Figure 5: Qualitative decision trajectories across the `battery`, `economy`, `telecom`, `supply` and `carbon` datasets under short, medium, and long decision horizons. In each figure, the gray line represents the observed time series, while the colored parts show the future time series and the corresponding oracle trajectory generated with access to future time series information.