EVIDENCE-GUIDED ACTION WORLD MODEL FOR TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series forecasting aims to predict future system states from historical observations and supports decision-making in applications such as energy management and retail planning. While modern forecasting methods effectively model endogenous temporal dynamics, real-world systems are also influenced by external factors that can alter their future evolution. Existing approaches typically incorporate such information as additional predictive inputs, directly learning mappings from external evidence to future observations. However, this treatment leaves the underlying dynamic response implicit. Importantly, the same external evidence may induce different responses depending on the current system state, making it difficult for existing methods to fully exploit external evidence. We therefore argue that forecasting should explicitly model the dynamic response implied by external evidence and characterize how this response changes system dynamics. Motivated by this perspective, we formulate forecasting under external evidence as an action-conditioned world-model problem, where latent actions explicitly represent dynamic responses to external influences and modulate future state transitions. We propose the *Evidence-Guided Action World Model* (EGA-WM), which learns a latent action space through a primitive bank, infers state-dependent actions from external evidence, and incorporates them into latent dynamics while modeling their persistent effects. Experiments on electricity-market and retail datasets demonstrate the effectiveness of EGA-WM across diverse forecasting scenarios.

1 INTRODUCTION

Time-series forecasting (TSF) supports decision-making by predicting future system states from historical observations, and has become important in many real-world applications such as energy management and retail planning. Many modern time-series forecasting methods primarily model endogenous dynamics from historical observations (Nie et al., 2023; Stitsyuk & Choi, 2025). However, future evolution is shaped not only by the system's internal dynamics but also by external influences that can alter these dynamics and, consequently, the future trajectory. For example, a scheduled promotion may reshape demand over the following days, causing the future trajectory to deviate from what historical dynamics alone would suggest. This poses a fundamental challenge to these methods, limiting their effectiveness when external influences reshape future dynamics.

To address this limitation, a growing line of work incorporates external information, such as known future covariates or exogenous variables, as additional predictive inputs (Wang et al., 2024; Qiu et al., 2026). However, these methods mainly learn direct associations between external evidence and future observations, without explicitly characterizing how the evidence changes the underlying system dynamics. Modeling these dynamic effects is important, because the response to external evidence is not fixed: the same evidence may induce different responses under different system states (Figure 1). For example, the same promotion schedule may produce different demand responses depending on current demand conditions. Explicitly modeling these responses allows a forecasting model to better capture how external evidence shapes future dynamics. However, existing methods leave this response implicit in the evidence–outcome mapping, which may limit their ability to accurately adapt their forecasts to external evidence under different conditions. To better account for external influences, a forecasting model should determine the system's response from the available evidence and current state, and model how that response shapes future evolution.

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.

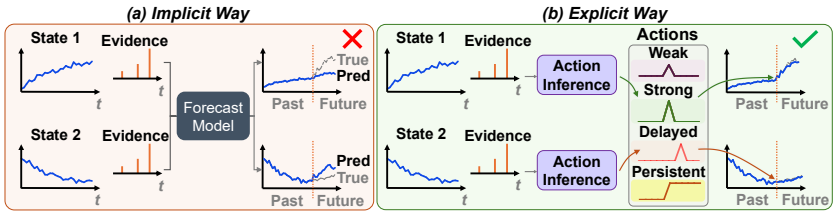


Figure 1: **State-dependent responses to external evidence.** The same evidence may correspond to responses with different timing and strength under State 1 and State 2. (a) Direct forecasting jointly maps history and evidence to future observations, leaving the response implicit. (b) State-conditioned action inference explicitly represents the response as a latent action. Different actions capture different response characteristics, and an appropriate action is inferred from the current state and available evidence to guide future evolution and forecasting.

This motivates an action-conditioned world-model formulation for time-series forecasting, in which future evolution is modeled through state transitions conditioned on actions (Hafner et al., 2020). In our setting, a latent action represents the dynamic response to an external influence and is determined from the available evidence and current system state. The world model uses this action to modulate future state transitions, allowing the response to propagate through subsequent states and shape the future trajectory. Although promising, applying this formulation to forecasting raises three fundamental questions. First, *how to learn the action space?* Since actions are neither predefined nor annotated in forecasting data, the model must learn representations of dynamic responses from observed trajectories. Second, *how to determine the action from external evidence?* The appropriate action depends not only on the evidence itself, but also on the current system state, and may vary in its timing, duration, and strength. Third, *how to translate the action into future dynamics?* The action should modify the system's state evolution rather than merely provide an additive correction to the final prediction, while allowing its consequences to persist beyond its direct temporal support.

To address these questions, we propose an *Evidence-Guided Action World Model* (EGA-WM) for forecasting under external evidence. EGA-WM constructs a latent action space through a learnable primitive bank, where each primitive represents a way that an external influence can change system dynamics. The bank serves as a basis for constructing an action through a soft combination of these primitives. To determine how the primitives should be combined under different conditions, a state-conditioned action inference network interprets the external evidence in the context of the current system state derived from the observed history. It then infers the corresponding action, together with its onset, duration, strength, and temporal support. The inferred action is then injected into an action-conditioned latent transition model, which separately modulates how the action enters the latent state and how its induced effect persists over time, allowing the action's consequences to persist as its activation fades. Together, the primitive bank, action inference network, and action-conditioned dynamics enable the model to translate external evidence into dynamic responses and incorporate their effects into future evolution for improved forecasting.

Our contributions are four-fold: (1) **Learnable action representation.** We construct a latent action space through a learnable primitive bank, where different primitives capture different ways of changing system dynamics and can be softly combined to form actions. (2) **State-conditioned action inference.** We develop an action inference network that interprets external evidence in the context of the current system state and infers the corresponding action together with its onset, duration, strength, and temporal support. (3) **Action-conditioned dynamics.** We incorporate the inferred action into latent state transitions and separately model how the action affects the latent state and how its induced effect persists over time, allowing its consequences to shape subsequent state evolution. (4) **Extensive empirical evaluation.** We evaluate EGA-WM on five electricity markets and two retail datasets, demonstrating its effectiveness across diverse forecasting scenarios.

2 Related Work

TSF. TSF research has developed diverse ways to capture temporal dependencies and evolving dynamics from historical observations (Nie et al., 2023; Shu & Lampos, 2026; Rangapuram et al., 2018; Liu et al., 2023). These approaches improve the extraction and extrapolation of historical patterns, but target history alone may not reveal upcoming external conditions. To account for such

information, covariate-aware forecasters incorporate known future covariates and exogenous variables alongside target history (Lim et al., 2021; Das et al., 2023; Wang et al., 2024; Qiu et al., 2026). The existing methods mainly learn a predictive mapping from historical observations and external evidence to future outcomes. Such mappings can capture interactions between the current state and external conditions, enabling forecasts to incorporate available evidence. However, the corresponding dynamic response generally remains implicit: its timing and persistence are embedded in the joint predictive mapping rather than explicitly represented and propagated through future state transitions. This makes it difficult to capture how the effect of external evidence unfolds over the forecast horizon.

World models and latent actions. Action-conditioned world models predict future state evolution under actions and support behavior learning through imagined trajectories (Hafner et al., 2020). Related work further learns latent actions from observed transitions for imitation learning and interactive generation (Edwards et al., 2019; Bruce et al., 2024). While these methods use actions to represent controllable behavior or interactions, our latent actions represent the predicted dynamic responses induced by external evidence. The central challenge is therefore to determine how the same evidence translates into different responses under different system states. EGA-WM addresses this challenge by inferring both the response and its temporal activation from the observed history and available evidence, and using the resulting action to modulate future state transitions. This formulation explicitly models when an action is active and how its effects propagate over time.

3 Methodology

3.1 Problem Definition

Let $\{x_\tau\}_{\tau=1}^{T}$ denote a TS sample, where $x_\tau \in \mathbb{R}$ is the value at time τ. At forecasting time t, the task is to predict the next H observations from the preceding L observations, together with available external evidence. We denote the historical window and prediction target by $X_t = (x_{t-L+1}, \ldots, x_t) \in \mathbb{R}^{L\times 1}$, $Y_t = (x_{t+1}, \ldots, x_{t+H}) \in \mathbb{R}^{H\times 1}$. We define external evidence as information available at prediction time about external events or conditions that may affect the future evolution of the target series. Here, the evidence is provided as time-indexed numerical features. Specifically, $E_t^- \in \mathbb{R}^{L\times d_e}$ denotes the evidence associated with the historical window, while $E_t^+ \in \mathbb{R}^{H\times d_e}$ denotes the evidence for the prediction window, where d_e is the number of evidence features. Given the input $\mathcal{I}_t = (X_t, E_t^-, E_t^+)$, we aim to learn a forecasting function F_θ such that $\widehat{Y}_t = F_\theta(\mathcal{I}_t)$.

3.2 Overall Framework

Figure 2 presents the overall framework, including three core components. To represent different dynamic responses, we first construct a learnable primitive bank that defines an action space, where each primitive is defined by trainable parameters and specifies a change in latent state transitions. Given a forecast window, the state-conditioned action inference network determines an action from the primitive bank. It first encodes the observed history and a reference forecast produced by a pretrained history-only forecaster to characterize the current system state. A learnable action head maps the state and evidence representations to soft weights over the primitives, and also predicts temporal activation parameters to determine when and how strongly the action is applied. These outputs define an action that specifies the dynamic response to the evidence under the current system state. The action-conditioned dynamics model uses the inferred action to evolve latent states over the forecast horizon. Starting from the state representation, each transition incorporates an action-dependent input and adjusts the retention of the previous latent state. The updated state is propagated to subsequent transitions, allowing action-induced changes to persist even as activation fades. The resulting latent states are decoded to determine the action-dependent modification to the forecast.

3.3 Learnable Action Representation

In this part, we define an action space through a learnable primitive bank. Each primitive serves as a building block to specify how to introduce changes into the latent state and adjust how much of the previous state is retained. These primitives are shared across forecast windows, providing reusable building blocks for dynamic responses.

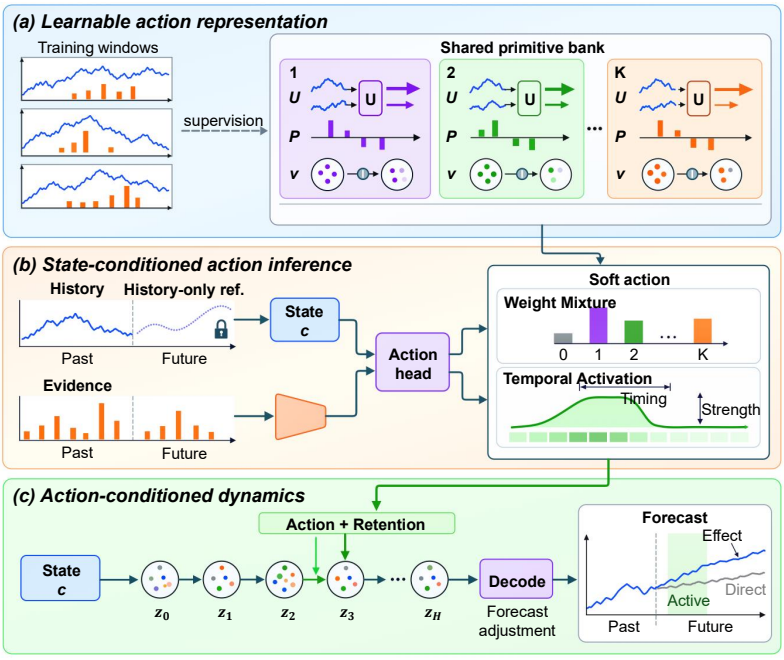


Figure 2: **Overall framework of EGA-WM.** (a) A learnable primitive bank defines a shared action space for representing different dynamic responses. (b) Observed history and a history-only reference forecast characterize the current system state. Together with external evidence, this state determines an action defined by a soft primitive mixture and its temporal activation, specifying the response, its timing, and its strength. (c) The inferred action modulates latent state inputs and retention, allowing its effects to propagate through subsequent states even as activation fades. The resulting latent states are decoded to produce an action-dependent forecast adjustment.

Specifically, we construct a bank containing K primitives, with each primitive j defined by trainable parameters $\{U_j, P_j, v_j\}$. Here, U_j describes how the input change depends on the current system state, P_j describes its variation across forecast steps, and v_j specifies adjustments to the retention of the previous latent state. Together, these parameters define two components of a primitive: an input change $f_{j,h}(c_t)$ and a state-retention change parameterized by v_j. To allow the input component to depend on the current state and vary across forecast steps, we express it as the sum of a state-dependent term and a step-dependent term:

$$f_{j,h}(c_t) = \tanh(W_c c_t + b_c)^\top U_j + P_{j,h}, \tag{1}$$

where c_t is the current-state representation defined in Section 3.4, and W_c and b_c are the trainable weight and bias of a projection shared across primitives. The primitive parameters have dimensions $U_j \in \mathbb{R}^{r\times d_z}$, $P_j \in \mathbb{R}^{H\times d_z}$, and $v_j \in \mathbb{R}^{d_z}$, where r and d_z denote the state-feature and latent dimensions, respectively. The vector $P_{j,h}$ is the h-th row of P_j.

In Eq. (1), the shared projection first extracts nonlinear features from the current state, and these features are then mapped by U_j to the input contribution of primitive j, while $P_{j,h}$ adds a learned offset for forecast step h. Thus, the same primitive can adapt its input contribution to different system states and vary it over the forecast horizon, even though c_t remains fixed within a window.

With this bank, the next module determines the actions based on current states and available evidence.

3.4 State-Conditioned Action Inference

In this part, we determine which primitives to combine for actions and when and how strongly their combination should act for a given forecast window. Since the response to external evidence depends on the current system conditions, we first construct representations of the state and evidence. A learnable action head uses these representations to predict the primitive weights and temporal activation parameters, which together specify the action.

State context. To characterize the conditions under which external evidence is interpreted, we use both the observed history and a history-only reference forecast. The history describes the system's observed behavior, while the reference summarizes the continuation suggested by that history. Specifically, a pretrained, frozen forecaster B_{hist} produces the reference R_t, and two encoders map the history and reference to a common state context:

$$R_t = B_{\text{hist}}(X_t), \qquad c_t = [h_t \parallel r_t] = [\Phi_x(X_t) \parallel \Phi_r(R_t)]. \tag{2}$$

Here $R_t \in \mathbb{R}^{H\times 1}$, and h_t and r_t are the encoded history and reference, respectively. Each encoder captures both local changes and periodic patterns in its input sequence, providing the action head with information about the system's recent behavior and periodic context (Appendix B). Their concatenation c_t characterizes the current state using only observed history and its predicted continuation, without access to future target values. This context remains fixed within the forecast window.

Evidence context. To determine an action, we consider not only the upcoming external conditions but also how they differ from recent conditions. To provide this distinction for action inference, we encode both the evidence and provide an explicit comparison between upcoming and recent conditions. Specifically, we encode the full historical evidence E_t^- to capture the external conditions preceding the forecast window. For the comparison, we take the last H steps of E_t^- as a recent reference $\bar{E}_t$, assuming $L \geq H$. This gives a reference of the same length as the horizon evidence E_t^+, allowing us to compute their elementwise difference $E_t^+ - \bar{E}_t$. We then separately encode $[E_t^+ \parallel \bar{E}_t \parallel (E_t^+ - \bar{E}_t)]$, which explicitly presents the upcoming conditions, the recent reference, and their difference. We fuse the two encodings to obtain u_t^e, which provides the action head with both the historical background and the upcoming conditions relative to the recent window, helping it determine the action together with the current-state representation (See Eq. (13) for details).

Inferring the soft action. With the state and evidence contexts, we infer a soft primitive mixture through a learnable action head. Specifically, the head outputs weights $\pi = (\pi_0, \pi_1, \ldots, \pi_K) \in \Delta^K$, where $\pi_1, \ldots, \pi_K$ specify the contributions of the K primitives and π_0 corresponds to no modulation. These weights are used to combine both the input and state-retention components of the primitives, allowing an action to incorporate multiple primitives rather than selecting only one. Here Δ^n denotes the simplex of $n+1$ nonnegative weights summing to one.

To determine these weights from the current state and evidence, the action head first forms a shared representation:

$$w_t = \mathcal{H}([h_t \parallel r_t \parallel \sigma(\gamma) u_t^e]), \tag{3}$$

where σ is the sigmoid function and the trainable scalar γ controls the contribution of the evidence representation. The primitive weights are then computed as

$$\pi = \text{softmax}(W_k w_t / T_a), \tag{4}$$

where T_a is the softmax temperature. These weights are used directly during training and forecasting, without selecting a single primitive. Thus, the primitive combination is determined by the evidence in the context of the current state, rather than assigning a fixed combination to each evidence pattern.

Temporal activation. Alongside the primitive weights, the head predicts temporal activation parameters to determine when and how strongly the combination acts over the forecast horizon. The onset weights $\alpha \in \Delta^{H-1}$ and duration weights $\beta \in \Delta^{|\mathcal{D}|-1}$ jointly describe its soft temporal support, where $\mathcal{D}$ is a fixed grid of candidate durations. The strength $s \in [0, s_{\max}]$ controls the overall activation magnitude, while the mask $m \in [0,1]^H$ allows activation to vary across individual forecast steps. Using the same representation w_t, we separately predict these temporal parameters in parallel with the primitive weights:

$$\alpha = \text{softmax}(W_\tau w_t / T_a), \beta = \text{softmax}(W_d w_t / T_a), s = s_{\max}\sigma(W_s w_t), m = \sigma(W_m w_t). \tag{5}$$

Together, the primitive weights and temporal parameters define the action $a = (\pi, \alpha, \beta, s, m)$. The onset and duration weights are also used directly during training and forecasting, without selecting a single onset or duration.

We combine these temporal parameters into a stepwise activation profile, which controls how strongly the primitive mixture modifies the latent state transition at each forecast step:

$$g_h(a) = s\, m_h \sum_{\tau=1}^{H} \sum_{d\in\mathcal{D}} \alpha_\tau \beta_d \, \mathbf{1}\{\tau \leq h < \tau + d\}. \tag{6}$$

The indicator marks the steps covered by an interval starting at τ and lasting d steps. Their weighted sum provides soft temporal support rather than a single hard interval, while s and m_h scale the activation globally and at each step, respectively. This separates which transition changes are combined, specified by π, from when and how strongly they act, specified by $g_h(a)$. Since activation is nonnegative, the signs of the changes come from the primitive components. The next module uses the mixture and activation profile to modify future state transitions.

3.5 Action-Conditioned Dynamics

Given the inferred action, this module applies it to successive latent state transitions and decodes the resulting states to refine the forecast. The action modifies the input contribution and state retention at each step, while recurrent updates allow these changes to influence subsequent states.

State initialization and forecast input. Since each latent state is updated from the preceding state, we initialize the recurrence using the current-state representation c_t:

$$z_0^a = \tanh(W_0 c_t + b_0).$$

Here $z_0^a \in \mathbb{R}^{d_z}$ is the initial latent state, allowing the state evolution to start from the conditions represented by the observed history and its predicted continuation.

To condition subsequent state updates on the forecast being refined, we use $b_t = B_{\mathrm{dir}}(\mathcal{I}_t) \in \mathbb{R}^{H\times 1}$, produced from the observed history and available evidence by a pretrained, frozen forecaster. Unlike the history-only reference R_t used to construct c_t, b_t supplies a prediction for each forecast step through the latent input:

$$u_h = W_b b_{t,h} + b_b + \eta_h.$$

Here $u_h \in \mathbb{R}^{d_z}$ provides the update with the corresponding baseline prediction, while the learned horizon embedding η_h identifies the forecast step.

Applying the inferred action. The inferred action modifies each transition through an additional input and a change in state retention. Specifically, the primitive weights π combine the input components $f_{j,h}(c_t)$ and retention components v_j, while temporal activation $g_h(a)$ controls their influence at step h. Let ρ denote the learned retention vector controlling how much of the previous latent state is retained without action modulation. With a fixed modulation scale $\lambda_\rho > 0$ and retention bounds $0 < \rho_{\min} < \rho_{\max} < 1$, we compute

$$u_h^{act}(a) = g_h(a) \sum_{j=1}^{K} \pi_j f_{j,h}(c_t), \tag{7}$$

$$\rho_h^a = \mathrm{clip}\left(\rho + \lambda_\rho g_h(a) \tanh\left(\sum_{j=1}^{K} \pi_j v_j\right), \rho_{\min}, \rho_{\max}\right). \tag{8}$$

Here $u_h^{act}(a)$ determines the additional input introduced by the action, and ρ_h^a determines how much of each coordinate of the previous latent state is retained. This bounded modulation allows retention to increase or decrease relative to ρ while keeping its coefficients below one. Parameterization and numerical settings are given in Appendix B.

These quantities are used directly in the state update:

$$z_h^a = \rho_h^a \odot z_{h-1}^a + u_h + u_h^{act}(a). \tag{9}$$

Here $\odot$ denotes elementwise multiplication. At each step, the model retains part of the previous state, incorporates the baseline input, and adds the action-dependent input. The action therefore changes the evolving latent state through both what enters it and what is carried forward.

Propagation of action effects. Each updated state becomes the input state of the next transition, allowing action-induced changes to influence later states. Temporal activation controls the modulation applied at the current step; it does not reset changes already accumulated in the state. If $g_h(a) = 0$ at a later step, the additional input vanishes and retention returns to ρ, but earlier action effects can still be carried forward by the recurrence. Consequently, the duration of activation need not coincide with the duration of its influence on subsequent predictions.

Obtaining the final forecast. A shared decoder D maps each latent state to a scalar output. To compute the action-dependent modification, we also evaluate an unmodulated state sequence with the same initialization and baseline inputs: $z_0^0 = z_0^a$ and $z_h^0 = \rho \odot z_{h-1}^0 + u_h$. The difference between the decoded outputs is added to the baseline forecast:

$$\widehat{Y}_t = b_t + \big[D(z^a) - D(z^0)\big]. \tag{10}$$

Here D is applied at each forecast step. This construction preserves b_t exactly when no action modulation is applied throughout the horizon. Appendix C gives the proof and the recurrence describing how action-induced state differences propagate.

For training, we compare the predicted values with the observed future targets and use the resulting error to jointly learn the primitive bank, action inference network, and action-conditioned dynamics (Appendix D). At test time, the model leverages the learned primitive bank to construct an action for each new forecast window based on the current state and available evidence, and uses this action to guide future state evolution and generate evidence-informed forecasts.

4 Experiments

We evaluate EGA-WM through comparisons with representative forecasters, ablations of its three components, and analyses of the size and learned responses of the primitive bank.

4.1 Datasets and Evaluation Protocol

We use EPF (Lago et al., 2021), Rossmann (Rossmann and Kaggle, 2015), and M5 (Makridakis et al., 2022) to evaluate forecasting with two forms of external evidence: *forecast-based evidence*, which provides forecasts of exogenous variables, and *event-based evidence*, which provides records or known schedules of external events. EPF comprises five electricity markets: Belgium, Germany, France, Nord Pool, and PJM. Each market uses 168 hours of history to predict the next 24 hourly prices, with two supplied day-ahead forecast channels describing load, generation, or wind. Rossmann contains 934 daily store series; we use 84 days of history to forecast 42 days, with promotion, opening, and holiday schedules as evidence. M5 contains 30,490 bottom-level item–store series; we use 112 days of history to forecast 28 days, with calendar, SNAP, and price information. These datasets cover different temporal resolutions, forecast horizons, and forms of external evidence. Chronological splits and preprocessing are detailed in Appendix A.

Metrics. We report mean squared error (MSE) and mean absolute error (MAE) on standardized targets, together with mean absolute scaled error (MASE) (Hyndman & Koehler, 2006) on the original scale. MASE uses the training-period weekly seasonal-naive error for normalization, as specified in Appendix A. Lower values indicate better performance for all metrics.

4.2 Comparisons with State-of-the-Art Forecasters

We compare EGA-WM with two categories of forecasting methods: (1) methods not specifically designed to incorporate external evidence, including xPatch (Stitsyuk & Choi, 2025), SimpleTM (Chen et al., 2025), and TimeMosaic (Ding et al., 2026); and (2) methods designed to incorporate external evidence, including TiDE (Das et al., 2023), TimeXer (Wang et al., 2024), XLinear (Chen et al., 2026), Sonnet (Shu & Lampos, 2026), and DAG (Qiu et al., 2026), which are provided with the same external evidence as EGA-WM for a fair comparison.

Table 1 summarizes the forecasting results, where EPF is reported as the macro-average over its five markets, with full results provided in Appendix E. Overall, methods specifically designed to incorporate external evidence generally outperform those without dedicated evidence modeling, demonstrating the value of external information for forecasting. However, these methods primarily incorporate the evidence into the predictive mapping without explicitly characterizing how it changes the underlying system dynamics, limiting their ability to fully exploit its dynamic effects. By explicitly modeling such effects, EGA-WM achieves the lowest mean errors among the compared methods, demonstrating the effectiveness of EGA-WM in leveraging external evidence for forecasting.

Table 1: Forecasting results averaged over three seeds. EPF is the macro-average over five markets.

	EPF			Rossmann			M5		
Method	MSE	MAE	MASE	MSE	MAE	MASE	MSE	MAE	MASE
TiDE	0.3892	0.3413	0.8895	0.2859	0.3488	2.3177	1.5783	0.8983	1.6323
TimeXer	0.2466	0.2448	0.6410	0.1051	0.2029	1.1839	1.5318	0.9044	1.6400
xPatch	0.3780	0.3491	0.9078	0.2084	0.2772	1.6322	1.5313	0.8959	1.6281
SimpleTM	0.4871	0.4123	1.0722	0.2282	0.2872	1.7602	1.5464	0.8925	1.6211
XLinear	0.3176	0.2419	0.6315	0.0586	0.1726	1.5247	1.5344	<u>0.8846</u>	<u>1.6096</u>
Sonnet	0.2416	0.2368	0.6184	<u>0.0194</u>	<u>0.0823</u>	<u>0.6814</u>	<u>1.5246</u>	0.9008	1.6350
TimeMosaic	0.2473	0.2432	0.6350	0.2089	0.2737	1.7091	1.5429	0.8919	1.6198
DAG	<u>0.2188</u>	<u>0.2281</u>	<u>0.5973</u>	0.0727	0.1418	1.1835	1.5719	0.9031	1.6384
EGA-WM	**0.2087**	**0.2193**	**0.5734**	**0.0084**	**0.0453**	**0.3891**	**1.5220**	**0.8834**	**1.6067**

4.3 Ablation Study

To examine our core contributions, we construct six variants and include the evidence-informed direct forecast without the action branch as *Baseline*. For learnable action representation (LAR), *w/o Primitive bank* removes the primitive bank and directly infers the action vector from state and evidence to evaluate the role of primitives in constructing effective actions. For state-conditioned action inference (SCAI), *w/o Temporal activation* removes activation parameters and applies actions with constant activation throughout the horizon, aiming to evaluate the importance of inferring when actions should act. *w/o Evidence conditioning* removes external evidence from action inference and uses only the current state to infer actions, aiming to evaluate the role of external evidence in determining effective actions. Similarly, *w/o State conditioning* uses only external evidence without state representations to infer actions, aiming to evaluate the importance of adapting actions to the current system state. Notably, these two variants remove evidence and state information, respectively, only from the action inference network, while still retaining their use in other components. For action-conditioned dynamics (ACD), *w/o State propagation* removes recurrent state propagation and directly maps inferred actions to forecast adjustments, aiming to evaluate the importance of propagating action effects through future states. *w/o Retention modulation* removes action-dependent retention changes and uses the learned retention for all actions, aiming to evaluate the importance of adapting how action effects persist over time.

Table 2: The ablation study. EPF is the five-market macro-average.

		EPF			Rossmann			M5		
Component	Variant	MSE	MAE	MASE	MSE	MAE	MASE	MSE	MAE	MASE
	Baseline	0.2398	0.2337	0.6093	0.0105	0.0655	0.5794	1.6513	0.9573	1.7167
LAR	w/o Primitive bank	0.2330	0.2276	0.5924	0.0099	0.0629	0.5474	1.5994	0.9282	1.6789
SCAI	w/o Temporal activation	0.2176	0.2199	0.5812	0.0092	0.0540	0.4601	1.5497	0.9017	1.6228
	w/o Evidence conditioning	0.2323	0.2289	0.5985	0.0101	0.0621	0.5650	1.5904	0.9204	1.6612
	w/o State conditioning	0.2354	0.2309	0.6028	0.0104	0.0654	0.5646	1.6289	0.9421	1.6929
ACD	w/o State propagation	0.2294	0.2252	0.5907	0.0097	0.0629	0.5368	1.5893	0.9295	1.6592
	w/o Retention modulation	0.2207	0.2231	0.5869	0.0094	0.0599	0.5288	1.5687	0.9201	1.6305
	Full	**0.2087**	**0.2193**	**0.5734**	**0.0084**	**0.0453**	**0.3891**	**1.5220**	**0.8834**	**1.6067**

From Table 2, we can find the effective contributions of the three components. First, for LAR, removing the primitive bank consistently degrades performance, showing the benefit of constructing actions from shared learnable primitives. Second, for SCAI, removing either the evidence or the state conditioning from action inference component causes clear performance drops, with *w/o State conditioning* generally leading to the largest degradation, e.g., increasing MSE by 12.79% on EPF, highlighting the importance of state context in interpreting external evidence. Removing temporal activation also degrades performance, demonstrating its effectiveness in modeling when actions take effect. Third, for ACD, both state propagation and retention modulation improve forecasting, while the larger degradation without state propagation indicates its important role in modeling how action effects evolve through future states.

4.4 Sensitivity to Primitive Bank Size

The bank size K determines the number of primitives in the action bank. To examine its effect, we evaluate $K = 2, 4, 8, 16, 32$ on PJM with all other settings fixed. Figure 3 reports the percentage change in error relative to the default setting $K = 8$. The MSE decreases as the bank size increases from two to eight, remains nearly unchanged at sixteen, and increases when the bank is further enlarged. These results suggest that a moderate bank size provides sufficient response diversity to learn effective actions, whereas adding more primitives does not necessarily improve forecasting performance.

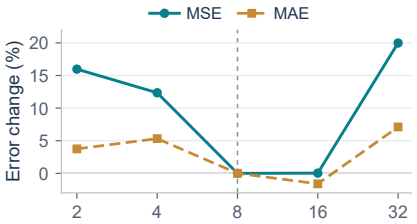


Figure 3: Sensitivity to K on PJM.

4.5 Visualizing the Learned Dynamic Responses

To examine the learned dynamic responses, we visualize four selected primitives and their inferred mixture weights for a forecasting window on the PJM dataset, where the task is to predict day-ahead electricity prices. Figure 4 shows the distinct response patterns of these primitives, each activated separately to illustrate its adjustment to the direct forecast. In this visualization, the external evidence includes day-ahead forecasts of ComEd zonal electricity load, with the mean predicted load over the next 24 hours being 5.3% higher than the preceding day. Primitive 1, which exhibits a sharp rise followed by a decline, receives the largest mixture weight, followed by Primitive 2, consistent with the higher electricity load. This example illustrates how the learned bank provides diverse dynamic responses and how action inference assigns different weights to these responses based on the current state and external evidence.

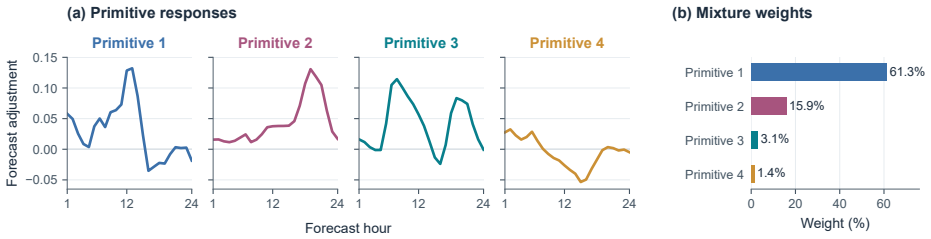


Figure 4: **Learned responses under load-increase evidence for electricity price forecasting on PJM.** (a) Responses of four selected primitives in a window with higher ComEd zonal load relative to the preceding day. Each curve shows the change to the direct forecast when an individual primitive is activated. (b) The mixture weights. Primitive 1, which exhibits a sharp rise followed by a decline, receives the largest mixture weight, consistent with the higher electricity load.

5 Conclusion

We formulated time-series forecasting with external evidence as an action-conditioned world-model problem and proposed EGA-WM to explicitly model the state-dependent dynamic responses implied by such evidence. EGA-WM learns a latent action space through a primitive bank, infers a soft combination of primitives together with their temporal activation from the current state and available evidence, and uses the resulting action to modulate latent state transitions. Together, the primitive bank, action inference network, and action-conditioned dynamics enable the model to translate external evidence into dynamic responses and propagate their effects through future state evolution. Experiments on electricity-market and retail datasets demonstrate improved forecasting performance over the compared methods under both forecast-based and event-based evidence, while ablation studies verify the contributions of the three components. These results demonstrate the benefit of explicitly representing state-dependent responses to external evidence and modeling how their effects shape future dynamics.

AI USE STATEMENT

We used generative AI tools to polish the language of this manuscript and improve its clarity and readability. The authors take full responsibility for the accuracy and integrity of the final manuscript.

REFERENCES

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktaschel. Genie: Generative interactive environments. In *International Conference on Machine Learning*, volume 235, pp. 4603–4623, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Hui Chen, Viet Luong, Lopamudra Mukherjee, and Vikas Singh. SimpleTM: A simple baseline for multivariate time series forecasting. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/27c546ab1e4f1d7d638e6a8dfbad9a07-Abstract-Conference.html.

Xinyang Chen, Huidong Jin, Yu Huang, and Zaiwen Feng. XLinear: A lightweight and accurate mlp-based model for long-term time series forecasting with exogenous inputs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(24):20325–20335, 2026. doi: 10.1609/aaai.v40i24.39121.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023. URL https://arxiv.org/abs/2304.08424.

Kuiye Ding, Fanda Fan, Chunyi Hou, Zheya Wang, Lei Wang, Zhengxin Yang, and Jianfeng Zhan. TimeMosaic: Temporal heterogeneity guided time series forecasting via adaptive granularity patch and segment-wise decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 20790–20798, 2026. doi: 10.1609/aaai.v40i25.39218. URL https://ojs.aaai.org/index.php/AAAI/article/view/39218.

Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *International Conference on Machine Learning*, pp. 1755–1763, 2019. URL https://proceedings.mlr.press/v97/edwards19a.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.01603.

Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. doi: 10.1016/j.ijforecast.2006.03.001.

Jesus Lago, Grzegorz Marcjasz, Bart De Schutter, and Rafal Weron. Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy*, 293:116983, 2021. doi: 10.1016/j.apenergy.2021.116983. URL https://arxiv.org/abs/2008.08004.

Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021. doi: 10.1016/j.ijforecast.2021.03.012. URL https://arxiv.org/abs/1912.09363.

Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/28b3dc0970fa4624a63278a4268de997-Abstract-Conference.html.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022. doi: 10.1016/j.ijforecast.2021.11.013. URL https://doi.org/10.1016/j.ijforecast.2021.11.013.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Xiangfei Qiu, Yuhan Zhu, Zhengyu Li, Xingjian Wu, Bin Yang, and Jilin Hu. DAG: A dual correlation network for time series forecasting with exogenous variables. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2509.14933.

Syama Sundar Rangapuram, Matthias Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, 2018. URL https://www.amazon.science/publications/deep-state-space-models-for-time-series-forecasting.

Rossmann and Kaggle. Rossmann store sales. Kaggle competition dataset, 2015. URL https://www.kaggle.com/competitions/rossmann-store-sales/data.

Yuxuan Shu and Vasileios Lampos. Sonnet: Spectral operator neural network for multivariable time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25419–25427, 2026. doi: 10.1609/aaai.v40i30.39736. URL https://ojs.aaai.org/index.php/AAAI/article/view/39736.

Artyom Stitsyuk and Jaesik Choi. xPatch: Dual-stream time series forecasting with exponential seasonal-trend decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20601–20609, 2025. doi: 10.1609/aaai.v39i19.34270. URL https://ojs.aaai.org/index.php/AAAI/article/view/34270.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. TimeXer: Empowering transformers for time series forecasting with exogenous variables. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0113ef4642264adc2e6924a3cbbdf532-Abstract-Conference.html.

A Data and Evaluation Details

Chronological splits. EPF uses three training years, one validation year, and two test years, with 364-day benchmark years. Rossmann retains 934 stores with complete daily grids from the labeled competition data and uses 774 training days, 84 validation days, and 84 test days; evaluation origins are seven days apart. M5 uses days 1–1885 for training, 1886–1913 for validation, and 1914–1941 for testing. Table 3 gives series counts, lookbacks, horizons, and channels.

Table 3: Forecasting protocols. Channels count the encoded exogenous inputs.

Dataset	Series	Lookback	Horizon	Channels	Evidence
EPF	5 markets	168	24	2/market	Load / generation / wind
Rossmann	934	84	42	8	Opening / promotions / holidays
M5	30,490	112	28	39	Calendar / SNAP / prices

Seasonal error scale. For series i, with training length T_i and evaluated origin–horizon pairs $\mathcal{E}_i$, we compute

$$q_i = \frac{1}{T_i - m} \sum_{t=m+1}^{T_i} |y_{i,t} - y_{i,t-m}|, \qquad \mathrm{MASE}_i = \frac{1}{|\mathcal{E}_i|} \sum_{(o,h)\in\mathcal{E}_i} \frac{|y_{i,o+h} - \widehat{y}_{i,o+h|o}|}{q_i}. \tag{11}$$

The seasonal lag is $m = 168$ for hourly EPF and $m = 7$ for daily retail data. Both numerator and denominator use original target units, making MASE dimensionless.

Preprocessing. EPF targets are standardized using the mean and standard deviation computed from the training data. For Rossmann and M5, we first apply a $\log(1 + x)$ transformation to sales and then standardize each series using statistics from its training period. MSE and MAE are computed in the standardized space, while MASE is evaluated in the original target scale after inverting the transformations.

B Implementation and Configurations

Table 4 summarizes the action-model settings.

Dynamics parameterization and initialization. The mappings defining z_0^a and u_h use trainable projections W_0 and W_b with biases b_0 and b_b, respectively. We parameterize the baseline retention as

$$\rho = 0.6 + 0.3\sigma(v_\rho), \qquad v_\rho \in \mathbb{R}^{d_z},$$

where v_ρ is trainable. In Eq. (8), we fix $\lambda_\rho = 0.1$, $\rho_{\min} = 0.55$, and $\rho_{\max} = 0.99$ across all datasets. These bounds contain the retention coefficients, so zero action modulation leaves retention unchanged. We zero-initialize the decoder's final affine layer so that learning starts from the direct forecast b_t.

Encoder and evidence definitions. Each encoder fuses temporal and spectral representations as

$$\Phi(V) = \mathrm{GELU}\big(\mathrm{LN}\big(W[\Phi_{\mathrm{time}}(V) \,\|\, \Phi_{\mathrm{freq}}(V)] + b\big)\big), \tag{12}$$

where LN is layer normalization. With $\bar{E}_t$ the last H historical evidence entries, the evidence context is

$$u_t^e = A_e\big([\Phi_e^-(E_t^-) \,\|\, \Phi_e^+([E_t^+ \,\|\, \bar{E}_t \,\|\, (E_t^+ - \bar{E}_t)])]\big). \tag{13}$$

Here A_e maps the two encoded sequences to the common feature width. All inputs are available at the forecast cutoff.

Encoders and decoder. The temporal encoder first constructs five groups of temporal features: levels, first differences, cadence-lag differences, local means, and local volatility. The cadence lag is set to 24 for hourly EPF data and 7 for daily retail data, and is capped by the encoded sequence length minus one. These features are processed by a convolutional network consisting of a width-5

Table 4: Action-model configuration.

Setting	EPF	Rossmann	M5
Feature / latent / head width	64 / 64 / 128	32 / 32 / 64	32 / 32 / 64
Primitives K	8	8	8
Primitive rank r	16	8	8
Cadence lag	24	7	7
Duration grid	1,3,6,12,24	1,3,7,14,28,42	1,3,7,14,28
History / profile / mask harmonics	16 / 10 / 8	16 / 10 / 8	16 / 10 / 8
Learning rate	10^{-3}	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
Batch size	64	256	512
Action temperature T_a	0.8	0.8	0.8
Maximum strength $s_{\max}$	4	4	4
Evidence-gate initial logit	-1.5	-1.5	-1.5

convolution followed by two width-3 dilated convolutions with dilation rates 2 and 4, together with a residual connection. Average, maximum, and terminal features are then pooled and projected to obtain the temporal representation. The spectral encoder extracts DC, cosine, sine, and harmonic-amplitude features, concatenates them, and processes the resulting representation using two affine layers with a GELU activation and layer normalization after the first layer. The number of harmonics is capped at $\lfloor (\ell - 1)/2 \rfloor$, where ℓ denotes the encoded sequence length. With feature width d_f, the resulting representations satisfy $h_t, r_t, u_t^e \in \mathbb{R}^{d_f}$ and $c_t \in \mathbb{R}^{2d_f}$, yielding an action-head input of dimension $3d_f$. The action head consists of two affine layers with GELU and layer normalization after the first layer. Finally, the shared state decoder applies layer normalization, projects the latent state from d_z to $d_z/2$, and uses GELU followed by a scalar output layer.

Pretrained forecasters. We use Sonnet (Shu & Lampos, 2026) for both B_{hist} and B_{dir}. The history-only forecaster B_{hist} uses only target history, while B_{dir} additionally receives historical and horizon evidence. Both forecasters are pretrained separately on the training data and kept frozen during action-model training.

C Null Identity and Temporal Propagation

Proposition 1 (Exact null forecast)**.** *If $\pi_0 = 1$, or $g_h(a) = 0$ throughout the horizon, then $\widehat{Y}_h(a) = b_h$ for every forecast step.*

Proof. If $\pi_0 = 1$, the simplex constraint implies $\pi_j = 0$ for all $j \geq 1$. Hence $u_h^{act}(a) = 0$ and $\rho_h^a = \rho$ at every step, since every coordinate of ρ lies within the clipping interval. The same equalities hold when $g_h(a) = 0$ throughout the horizon. The two trajectories share the initial state and baseline inputs, so Eq. (9) and the unmodulated recurrence yield $z_h^a = z_h^0$ for every h by induction. Their shared decoder outputs therefore cancel in Eq. (10), giving $\widehat{Y}_h(a) = b_h$. □

We implement the output as $b + [D(z^a) - D(z^0)]$, subtracting before adding the base. Each evaluated checkpoint is checked for exact numerical equality by explicitly setting the action to null. This is an identity of the architecture, not a requirement that its learned softmax attain a one-hot distribution.

Let $\delta z_h = z_h^a - z_h^0$ denote the action-induced state difference. Subtracting the action-conditioned and unmodulated recurrences gives

$$\delta z_h = \rho_h^a \odot \delta z_{h-1} + (\rho_h^a - \rho) \odot z_{h-1}^0 + u_h^{act}(a). \tag{14}$$

The three terms respectively propagate the accumulated difference, account for the change in retention, and introduce the additional input. When $g_h(a) = 0$, the last two terms vanish and the difference follows $\delta z_h = \rho \odot \delta z_{h-1}$.

Response after activation. If $g_h(a) = 0$ for all $h > h_*$, then Eq. (14) gives

$$\delta z_h = \rho^{\odot(h-h_*)} \odot \delta z_{h_*}, \qquad h > h_*. \tag{15}$$

The power is elementwise. A nonzero accumulated response can therefore remain after modulation stops, while its latent magnitude decays because every coordinate of ρ lies below one. This statement concerns latent state differences; the nonlinear decoder need not preserve monotonic decay in target space. In ordinary inference, the soft onset/duration weights and sigmoid mask need not become exactly zero, so the strict post-activation identity is a limiting or controlled case rather than a hard boundary imposed on every forecast.

For a diagonal recurrence $z_h = \rho_h \odot z_{h-1} + u_h$, define $P_h = \prod_{i=1}^{h} \rho_i$ elementwise. Then

$$z_h = P_h \odot \left(z_0 + \sum_{i=1}^{h} u_i \oslash P_i \right). \tag{16}$$

The two trajectories use this parallel scan at the experimental horizons of 24, 42, and 28 steps. The current implementation restricts this FP32 scan to horizons at most 64; the algebraic expression alone does not establish numerical stability for arbitrarily long horizons. Equation (14) follows by subtracting the two recurrences.

D Training and Forecasting Procedures

Training uses observed future targets to update the primitive bank, inference network, and dynamics. Forecasting uses the same action and transition computations with only the inputs available at the forecast cutoff.

Learning from training time series. For each training window, the model infers an action, evolves the paired latent trajectories, and produces a forecast. We compare this forecast with the observed future target using mean squared error:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|H} \sum_{i \in \mathcal{B}} \sum_{h=1}^{H} \left(\widehat{Y}_{i,h}(\mathcal{I}_i; \theta) - Y_{i,h} \right)^2. \tag{17}$$

Here $\mathcal{B}$ is a minibatch, and θ contains the state and evidence encoders, action head, primitive bank, transition parameters, and shared decoder. Backpropagating the forecast error through the decoder and transitions updates the primitive parameters and action inference network without action annotations. Forecast error alone does not ensure distinct primitive meanings or identify physical mechanisms.

Forecasting at a new window. We encode state and evidence, infer the soft action, and evolve the two latent trajectories. The forecast is obtained using Eq. (10). The algorithms below summarize the training and forecasting procedures.

Algorithm 1 Learning evidence-guided actions

Require: Training windows (X, E^-, E^+, Y) and the reference inputs R and b defined in Section 3.

1: Initialize sequence encoders, action head, and primitive transition bank.
2: Zero-initialize the final shared-decoder affine map.
3: **for** each training minibatch **do**
4: Encode the state $c = [\Phi_x(X) \,\|\, \Phi_r(R)]$.
5: Encode evidence with Eq. (13).
6: Infer $(\pi, \alpha, \beta, s, m)$ and form $g_h(a)$.
7: Mix the primitive forcing and persistence parameters.
8: Evaluate paired trajectories and $\widehat{Y} = b + [D(z^a) - D(z^0)]$.
9: Update the action-model parameters using Eq. (17).
10: **end for**

Algorithm 2 Forecasting at cutoff t

Require: Target history, historical evidence, available horizon evidence, and reference inputs R_t and b_t.

1: Encode state and evidence; infer the structured action.
2: Compute temporal activation and mix the primitive operators.
3: Roll out base and action-conditioned states from the same initial state.
4: **return** $b_t + [D(z^a) - D(z^0)]$.

E Detailed Electricity Forecasting Results

Table 5 reports the results for each of the five electricity markets. The EPF column in Table 1 averages the market-level results with equal weights.

Table 5: Results on the five EPF markets, averaged over three seeds. Best results are in bold and second-best results are underlined.

	Belgium (BE)			Germany (DE)			France (FR)		
Method	MSE	MAE	MASE	MSE	MAE	MASE	MSE	MAE	MASE
TiDE	0.6539	0.3971	1.1269	0.5335	0.4657	0.8702	0.3577	0.2976	0.8297
TimeXer	0.4842	0.3021	0.8573	0.2749	0.3172	0.5927	0.2252	0.1958	0.5459
xPatch	0.5727	0.3819	1.0838	0.5635	0.4781	0.8935	0.3208	0.2998	0.8358
SimpleTM	0.7286	0.4549	1.2909	0.7220	0.5678	1.0611	0.4275	0.3572	0.9956
XLinear	0.4530	0.2928	0.8309	0.2689	0.3188	0.5958	0.6287	0.2032	0.5665
Sonnet	0.5183	0.3098	0.8790	0.2533	0.3066	0.5730	0.2280	0.1940	0.5408
TimeMosaic	0.4758	0.2975	0.8442	0.2891	0.3270	0.6110	0.2208	0.1877	0.5232
DAG	**0.4207**	**0.2732**	**0.7752**	0.2292	0.3004	0.5613	0.2358	0.1885	0.5253
EGA-WM	0.4469	0.2887	0.8193	**0.2159**	**0.2875**	**0.5372**	**0.2006**	**0.1865**	**0.5199**

	Nord Pool (NP)			PJM		
Method	MSE	MAE	MASE	MSE	MAE	MASE
TiDE	0.2865	0.3250	1.0366	0.1146	0.2211	0.5841
TimeXer	0.1768	0.2339	0.7460	0.0716	0.1753	0.4631
xPatch	0.2921	0.3263	1.0408	0.1410	0.2592	0.6849
SimpleTM	0.3807	0.3873	1.2355	0.1769	0.2944	0.7779
XLinear	0.1650	0.2225	0.7096	0.0725	0.1721	0.4548
Sonnet	0.1424	0.2050	0.6540	0.0659	0.1684	0.4449
TimeMosaic	0.1833	0.2371	0.7564	0.0676	0.1667	0.4404
DAG	0.1494	0.2285	0.7290	0.0590	0.1498	0.3958
EGA-WM	**0.1304**	**0.1974**	**0.6297**	**0.0499**	**0.1366**	**0.3609**