ORIGINAL ARTICLE

T2f: Actor-critic reinforcement learning for time-series forecasting

João Sousa · **Roberto Henriques**

Received: 23 December 2025 / Accepted: 11 May 2026

Abstract
Time-series forecasting of multiple related sequences presents unique challenges due to the complex interplay between individual series characteristics and global patterns. We present *T2f*, a forecasting method combining ensemble learning with an actor-critic architecture based on the Twin Delayed Deep Deterministic algorithm (TD3). *T2f* balances local and global patterns through both its architecture and learning approaches, integrating transformer-based pattern recognition with reinforcement learning for dynamic model selection. Our method incorporates temporal attention mechanisms and context-aware error measurement, aligning forecasting objectives with practical decision-making priorities. Comprehensive ablation studies demonstrate that *T2f*'s components provide synergistic benefits: the TD3-based optimizer contributes 18.8% error reduction over static weighting, while temporal attention adds 8.0% improvement, with the integrated system outperforming simple ensemble baselines by over 20%. Experimental results across five diverse datasets indicate *T2f* reduced mean absolute error by over 30% compared to statistical models and achieved up to 40% better performance on context-weighted metrics than competing approaches. While specialized models occasionally outperformed *T2f* on highly regular patterns, it consistently showed superior adaptability to contextual weights with faster convergence, typically reaching near-optimal performance within 25 epochs compared to 40+ for alternative methods, particularly on datasets with complex temporal dynamics.

Keywords Time-series forecasting · Actor-critic methods · Transformer architecture · Reinforcement learning · Ensemble learning

1 Introduction

In the current landscape of big data analytics, operational forecasting increasingly involves predicting collections of multivariate, high-dimensional, related time-series. This challenge spans numerous industrial applications, from demand forecasting and inventory management [1, 2] to financial predictions [3]. The scale and complexity of these forecasting problems have outgrown traditional approaches, creating a pressing need for more sophisticated solutions.

Traditional time-series forecasting methods, including classical statistical approaches like ARIMA and exponential smoothing [4, 5], were designed for individual or small groups of series, relying heavily on expert input

Published online: 16 June 2026

for feature engineering and model design. However, the advent of large-scale datasets from diverse sources, such as energy consumption, server loads, and product demand, has required a shift toward fully automated, data-driven approaches [6]. This transition brings its own set of challenges, particularly in balancing local and global pattern recognition while maintaining computational feasibility.

Collections of multiple time-series often exhibit various forms of interconnection that can be leveraged for prediction. These include correlations between related factors (such as price-demand relationships), connections through shared underlying causes (such as the effects of the marketing campaign on multiple products), and similarities in patterns between related phenomena. However, capturing these dependencies effectively while avoiding overfitting remains a significant challenge [7].

Advances in deep learning have demonstrated the potential of neural network-based models to identify intricate patterns in large sets of interconnected time-series [8, 9]. Deep learning architectures have achieved remarkable success across diverse AI domains, from point cloud completion [10] and medical image synthesis [11] to intelligent transportation systems [12, 13] and energy-efficient perception [14]. These models excel at utilizing common smoothness, temporal dynamics, and responses to external factors. However, they often struggle with data efficiency and the capture of dependency patterns caused by shared underlying factors [15].

More recently, the emergence of transformer architectures has introduced new possibilities in sequence modeling, offering mechanisms to capture long-range dependencies and complex patterns in multiple series [16]. These architectures have shown particular promise in handling variable-length sequences and modeling complex dependencies. However, they face challenges in effectively combining local and global information, especially when dealing with heterogeneous time-series collections that exhibit varying degrees of correlation.

While transformer architectures have shown remarkable capabilities in capturing complex patterns, they often lack mechanisms for dynamically selecting and combining models based on changing conditions and contexts [17]. This limitation is particularly relevant in forecasting scenarios where the relative importance of different patterns and models may vary over time and across different series within a collection [18].

Reinforcement learning has recently emerged as a compelling approach for the selection and combination of dynamic models in forecasting tasks [19]. Its ability to learn optimal policies through environment interaction makes it particularly suitable for adapting to changing patterns in time-series data. This approach aligns well with the sequential nature of forecasting decisions and the need to balance exploration and exploitation in model selection [20].

The combination of transformer-based pattern recognition with reinforcement learning's dynamic decision-making capabilities creates a particularly promising approach for time-series forecasting [21]. Transformers can efficiently capture complex dependencies in the data, while reinforcement learning provides a principled framework for optimizing model selection and combination in response to changing conditions [22].

The complexity of modern forecasting problems has led to the development of hybrid approaches that combine multiple methodologies. These include Hierarchical Bayesian methods [23], Latent Variable Models [24], and Dynamic-Factor Models [25]. Following Wolpert's fundamental theorem [26], which states that no universal model can solve all forecasting problems, these hybrid approaches aim to leverage the strengths of different methodologies while mitigating their individual weaknesses [Anonymous].

We present *T2f*[1], a forecasting method addressing these challenges by combining traditional ensemble learning with an actor-critic architecture based on the Twin Delayed Deep Deterministic (TD3) algorithm [27]. Our approach frames the global-local trade-off as a sequential weight-determination problem, leveraging reinforcement learning to optimize model combination dynamically. This method builds upon recent work in forecast model averaging [28] while introducing a distinct approach that integrates both architectural and learning advancements.

T2f distinguishes itself through its specific technical approach to contextual optimization, employing a dual-stage architecture where reinforcement learning dynamically determines optimal weights for combining local

[1] *T2f: Transformer-based Twin Delayed Deep Deterministic policy gradient for forecasting*

model predictions based on both historical performance and current conditions. By incorporating relevant contextual dimensions, such as cost factors in economic predictions, our method provides a novel definition of weighted error measurement that is more in line with the context of real-world optimization and decision-making. This approach enables more nuanced and practical forecasting solutions while maintaining the ability to adapt to changing patterns and relationships between series.

Our key contributions include:

1. A reinforcement learning framework that dynamically determines optimal combination weights for multiple forecasting models, adapting to changing patterns and relationships between time-series. Our ablation study demonstrates that the TD3-based optimizer alone contributes an 18.8% reduction in sMAPE compared to static uniform weighting, validating the core value of dynamic adaptation.
2. A structured two-stage architecture that separately processes local series-specific patterns and global cross-series dependencies, integrating them through an actor-critic mechanism. Ablation experiments show that this integration produces synergistic benefits beyond the sum of individual components, with the full system outperforming simple ensemble baselines by over 20%.
3. A context-aware error measurement approach that incorporates domain-specific factors (such as costs) into both the evaluation metrics and learning process. The temporal attention mechanism provides an additional 8.0% improvement on standard metrics and 10.3% on context-weighted metrics.
4. Comprehensive empirical validation including ablation studies that quantify the contribution of each architectural component, demonstrating that *T2f* provides meaningful improvements beyond what simpler ensemble approaches can achieve.

2 Problem setting

The *T2f* framework addresses the fundamental challenge of forecasting multiple high-dimensional multivariate time-series with discrete time points and associated covariates. This setting addresses forecasting challenges where both global covariates (shared across all series) and local features (specific to individual series) jointly influence future values. Industrial applications of this problem abound, from retail demand forecasting to energy consumption prediction, where accurate multi-horizon forecasts directly impact operational efficiency and resource allocation decisions.

2.1 Mathematical formulation

For a given collection of time-series $\mathcal{S} = \{S_1, ..., S_n\}$, each series S_i consists of observations at discrete time points t. At each point, we observe both the target variable and a set of covariates $\mathcal{C}$, which can be decomposed into global covariates $\mathcal{C}_g$ (shared across all series) and local covariates $\mathcal{C}_l$ (specific to each series), such that $\mathcal{C} = \{\mathcal{C}_g \cup \mathcal{C}_l\}$. This distinction is crucial as it reflects the dual nature of information relevant to forecasting: series-specific characteristics and cross-series dependencies.

For instance, in retail demand forecasting, $\mathcal{C}_g$ might include calendar effects and nationwide promotions affecting all stores, while $\mathcal{C}_l$ could encompass store-specific attributes like local weather and regional events. A store's sales (S_i) would respond to both national holidays (global pattern) and local conditions (local pattern), illustrating why models must account for both information types.

The forecasting objective for each series can be expressed as finding an optimal function f that maps historical observations and covariates to future values:

$$\hat{y}_{t+1:t+F} = f(\mathrm{x}_{t-p+1:t}, \mathcal{C}) \tag{1}$$

where p represents the size of the historical window used for prediction, F denotes the forecast horizon (the number of future time points to be predicted), $\mathrm{x}_{t-p+1:t}$ represents the observed history of the series, and $\hat{y}_{t+1:t+F}$ represents the predicted future values. This formulation captures the sequential nature of the forecasting problem while acknowledging the influence of covariates on future values.

The key challenge lies in determining the optimal function f that can effectively capture both local patterns specific to individual series and global patterns shared across multiple series. Traditional approaches typically emphasize one of these aspects at the expense of the other, leading to suboptimal performance in complex real-world scenarios [2, 29].

2.2 Temporal attention mechanism

An important feature of the *T2f* framework is its temporal attention mechanism, which allows the model to assign different weights to historical observations based on their relevance and recency. Unlike standard sequence models that treat all historical points equally or apply a fixed decay pattern [30, 31], *T2f* implements an adaptive weighting scheme that considers both temporal position and data volatility.

The temporal attention weights α_t for each time step t are computed through a structured dual-factor approach:

$$\alpha_t = \exp(t/T) \cdot v_t \tag{2}$$

The first component, $\exp(t/T)$, implements an exponential recency bias where T is the total sequence length. This ensures that more recent observations generally receive higher weights in the prediction process, assuming greater relevance to future values. The second component, v_t, is a volatility-adjusted weight factor given by:

$$v_t = 1 + \lambda \cdot \frac{\Delta_t - \mu_\Delta}{\sigma_\Delta} \tag{3}$$

Here, Δ_t represents the volatility (measured as absolute differences between consecutive points), μ_Δ and σ_Δ are the mean and standard deviation of volatility across the series, and λ is a decay factor (defaulted to 0.1 in our experiments) controlling the influence of volatility on attention weights. This formulation follows contributions from [32–35] and allows *T2f* to adaptively increase attention on periods of unusual activity or change points in the series, while maintaining a balanced weighting scheme overall.

The combined attention mechanism enables *T2f* to focus on the most informative segments of historical data, which helps filter noise and emphasize significant patterns. As later demonstrated in our results, this approach proves particularly valuable in scenarios with irregular patterns, seasonal shifts, or regime changes, where uniform weighting would dilute important signals.

2.3 Learning dimensions

The framework operates across two primary dimensions that form a hierarchical learning structure. This dual approach allows *T2f* to balance computational efficiency with modeling flexibility, addressing the complex interplay between local and global patterns.

The Local Learning dimension ($\mathcal{L}$) focuses on individual series characteristics through a diverse pool of forecasting models. This stage trains multiple complementary models — in this paper assumed as Vector Autoregression (*VAR*), Long Short-Term Memory networks (*LSTM*) [36], Gated Recurrent Units (*GRU*) [37], and custom Adaptive Sequential Models (*AdaptiveSeq*). The latter adopt varying activation functions on each individual time-series. Each local model generates predictions independently for each series, and the predictions are then combined using weights determined by the global stage.

The global learning dimension ($\mathcal{G}$) builds upon the local models' outputs through a dedicated TD3 actor-critic architecture. This stage processes input data of dimension $\mathcal{G}$ (global length) and determines optimal weights

for combining the local models' predictions. The actor network employs a dilated CNN encoder that gradually expands its receptive field through increased dilation rates, allowing it to capture dependencies at different time scales without requiring deep architectures [16, 38]. In our implementation, the dilation rate increases by a factor of 2 with each layer (from *base* to $base^3$), enabling the network to effectively process patterns at multiple time scales while maintaining computational efficiency.

Then, the integration of a layer inspired by FEDFormer enables the model to exploit the inherent sparsity of time-series in the frequency domain, efficiently extracting relevant global patterns. The dual critic networks, as derived from TD3, provide robust value estimation for different weighting strategies, mitigating the overestimation bias common in reinforcement learning approaches [27, 39].

The interaction between these two dimensions provides the foundation for *T2f*'s forecasting framework to adapt to both series-specific characteristics and shared patterns across multiple series. The local stage ensures that individual series dynamics are accurately modeled, while the global stage optimizes the combination of these models based on their relative strengths and complementary information.

Our approach integrates evaluation metrics directly into the learning process through a reward function that combines both general accuracy and context-specific performance priorities. This integration of metrics into the learning process itself, rather than as mere evaluation tools, enables *T2f* to adapt its forecasting strategy based on the relative importance of different types of errors in the specific application context. Two metrics are employed:

The Symmetric Mean Absolute Percentage Error (sMAPE) provides a scale-independent measure of forecast accuracy, calculated as:

$$\text{sMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2} \tag{4}$$

This metric offers several advantages, including symmetry (avoiding bias toward over or under-forecasting), scale independence (enabling comparison across series with different magnitudes), and intuitive interpretation as a percentage error. The symmetric denominator addresses limitations of traditional MAPE when dealing with values close to zero, making it more robust across diverse forecasting scenarios.

The Weighted Symmetric Mean Absolute Error (wsMAE) extends sMAPE by incorporating contextual weights:

$$\text{wsMAE} = \frac{1}{n} \sum_{t=1}^{n} w_t \cdot \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2} \tag{5}$$

The weights w_t are derived from domain-specific covariates (such as cost factors, inventory levels, or criticality indicators) that reflect the differential impact of forecast errors across time points and series. In the *T2f* implementation, these weights are extracted from a designated weight variable in the covariates dataframe, providing a direct mechanism to incorporate domain knowledge about error importance. This weighted measure acknowledges that forecast errors can have varying impacts in real-world decision-making contexts, for instance, stock outs of high-margin products may be more costly than those of low-margin alternatives, or prediction errors during peak demand periods may have greater operational consequences than during off-peak times [40, 41].

The framework's reward function combines both metrics with equal weighting, ensuring *T2f* optimizes for practically relevant performance, not just abstract statistical accuracy.

It should be noted that while the current implementation focuses on point forecasts, we envision *T2f*'s ensemble architecture to naturally support future extensions for uncertainty estimation. For example, through variance-based approaches or exploration mechanisms in the actor-critic framework.

2.4 Data handling and preprocessing

The *T2f* framework incorporates some data handling capabilities addressing the challenges commonly encountered in real-world time-series data. The preprocessing pipeline includes several key components that ensure stable performance across diverse datasets.

Irregular timestamps are managed through a frequency inference mechanism that automatically detects the predominant sampling rate in each series. This approach analyzes time differences between consecutive observations and determines the most common interval, enabling appropriate resampling without requiring manual specification. For multi-frequency data, *T2f* resamples to a consistent frequency while preserving information through careful aggregation or interpolation.

Missing values, a pervasive challenge in time-series analysis, are addressed through a multi-step imputation strategy. The approach begins with forward filling (limited to two steps) to maintain temporal continuity, followed by backward filling (also limited to two steps) to address leading gaps. Any remaining missing values are handled through linear interpolation, which preserves trends while avoiding the introduction of artificial patterns. This graduated approach balances information preservation with the need for complete data in model training [42, 43].

Constant series, which can cause numerical instability in certain models, receive small random perturbations during training to ensure proper gradient flow, with this noise suppressed during inference to maintain deterministic predictions.

3 Related work

The development of *T2f* builds upon several intersecting research streams in time-series forecasting, deep learning, and reinforcement learning. This section provides a comprehensive analysis of prior work in these domains, highlighting the research gaps that *T2f* addresses.

3.1 Global-local modeling approaches

The challenge of balancing global and local patterns in time-series forecasting has been addressed through various methodological approaches. Traditional global methods, including dynamic factor models [44, 45] and principal component analysis [46], excel at capturing common trends but often struggle with local heterogeneity. Conversely, local approaches such as ARIMA models [47], exponential smoothing [48], and state space models [49] provide series-specific insights but fail to leverage cross-series information.

Recent advances in neural network architectures have demonstrated promising results in capturing complex relationships between multiple time-series. [50] introduced DeepGLO, pioneering a hybrid approach combining matrix factorization techniques with temporal convolutional networks (TCNs) to capture both global and local properties. Their matrix factorization component models cross-series relationships, while series-specific TCNs handle temporal dynamics. While achieving strong performance, this approach faces challenges with generalization errors when using global covariates for local predictions and struggles with computational scalability for very large collections of time-series.

[51] proposed *Deep Factors for Forecasting*, combining fixed global latent components with local statistical models, enabling probabilistic forecasting through Gaussian emissions. This framework offers flexibility in interpolating between local and global models but assumes automatic scale estimation rather than explicit specification. The author's approach uses a global deep component to extract common factors across series while modeling local deviations with traditional statistical methods, demonstrating strong performance on traffic and electricity datasets. However, its assumption of linear relationships between global factors and local patterns limits its applicability to complex, nonlinear dynamics.

Building on this work, [52] introduced memory networks to capture global patterns while using local statistics as initialization keys. Their Memory Time-series Network (MTNet) leverages attention mechanisms to access relevant historical patterns stored in memory components. However, its reliance on adversarial training can compromise model performance under certain conditions, particularly when dealing with highly heterogeneous time-series collections.

More recently, [53] developed the Informer architecture that employs a sparse attention mechanism to efficiently model long sequences. While primarily focused on efficiency, their approach implicitly addresses the global-local balance through a distilling attention mechanism that selectively focuses on the most informative tokens. Similarly, [54] proposed Autoformer, which decomposes time-series into trend and seasonal components using an auto-correlation mechanism that effectively captures periodicities at different scales. Both approaches demonstrate the potential of transformer-based architectures for capturing multi-scale patterns, but neither explicitly addresses the challenge of dynamically balancing local and global information based on their relative importance.

The FEDFormer architecture [55] introduces frequency domain transformations to capture global periodicities, showing particular strength in scenarios with strong seasonal patterns. More recently, PatchTST [56] demonstrated that segmenting time-series into patches significantly improves transformer performance, achieving state-of-the-art results on long-term forecasting benchmarks. The iTransformer [57], accepted as an ICLR 2024 Spotlight paper, further advances this direction by inverting the transformer structure to embed entire time-series as tokens, showing that proper application of attention mechanisms is crucial. TimeMixer [58] introduces decomposable multiscale mixing, while TimeXer [59] specifically addresses forecasting with exogenous variables. These developments underscore the continued evolution of transformer architectures for time-series, though none explicitly addresses dynamic model combination through reinforcement learning.

The emergence of foundation models for time-series, including Chronos [60], Moirai [61], and TimesFM [62], represents a paradigm shift toward pre-trained models that generalize across datasets. While these approaches offer impressive zero-shot capabilities, they typically require massive training data and compute resources, and may not adapt to domain-specific contexts as effectively as purpose-built ensemble methods like *T2f*.

Our approach builds upon FEDformer's insights regarding sparse frequency domain representations while extending them through a reinforcement learning framework that dynamically adapts to changing pattern relevance.

3.2 Reinforcement learning and actor-critic methods in time-series forecasting

The application of reinforcement learning (RL) to time-series forecasting has evolved significantly in recent years, moving beyond traditional optimization techniques. Early approaches focused primarily on simple Q-learning for individual series optimization, showing success in economic predictions [63, 64]. These methods optimized single model configurations through mechanisms like stochastic gradient ascent but were limited to relatively simple decision spaces.

More sophisticated applications emerged in specific domains such as financial forecasting, where [65] employed deep reinforcement learning for portfolio management, and [66] used deep Q-networks for algorithmic trading, showing how RL can effectively handle the complex, sequential nature of financial time-series. However, these applications typically focus on decision-making based on forecasts rather than improving the forecasting process itself.

Dynamic Model Selection (DMS) methods have shown effectiveness in short-term load forecasting applications [67, 68]. These approaches employ a two-step strategy in which Q-learning agents select optimal models from predefined pools based on recent performance. [69] formulated model selection as a Markov Decision Process, allowing for sequential decision-making about which model to trust under varying conditions. Although efficient for specific applications, DMS methods face scalability challenges due to their requirement to learn individual Q-tables for each rolling window and typically lack mechanisms for continuous adaptation of models.

Actor-critic algorithms represent a promising direction for forecasting problems with continuous action spaces such as model weight determination. These model-free incremental approaches based on stochastic gradient updates [70, 71] combine value function approximation (critic) with direct policy search (actor), offering both stability and efficiency. The theoretical foundation for using deterministic policies in continuous action spaces was established by the Deterministic Policy Gradient (DPG) algorithm [72], which was later extended to deep neural networks through Deep Deterministic Policy Gradient (DDPG) [73]. However, DDPG suffers from instability and sensitivity to hyperparameter choices, limiting its practical applicability.

Recent work on model combination frameworks, such as RLMC [74], has demonstrated the potential of DDPG in forecasting problems. However, these approaches often struggle with overestimation bias and high variance in multivariate spaces [75, 76]. The problem of value overestimation, first identified by [75], becomes particularly acute in complex, high-dimensional spaces where exploration is costly and efficient learning is essential.

The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [27] specifically addresses these challenges through several key mechanisms. First, it employs dual critics to mitigate overestimation bias, using the minimum value between two Q-networks to provide more conservative value estimates:

$$Q_{\text{target}} = r + \gamma \min(Q'_1(s', \pi'(s')), Q'_2(s', \pi'(s'))) \tag{6}$$

Second, TD3 implements delayed policy updates, updating the policy network less frequently than the critic networks to reduce variance and improve stability. Finally, it adds target policy smoothing through noise injection and clipping:

$$\pi'(s') = \text{clip}(\pi(s') + \epsilon, a_{\min}, a_{\max}), \epsilon \sim \mathcal{N}(0, \sigma) \tag{7}$$

This prevents the exploitation of extreme value estimates that might result from function approximation errors. These innovations collectively address the challenges of applying reinforcement learning to continuous action spaces, making TD3 particularly suited for ensemble weight determination in forecasting applications.

Subsequent developments have built upon TD3's innovations. [77] introduced Soft Actor-Critic (SAC), which incorporates entropy maximization to balance exploration and exploitation. [78] extended TD3 to off-policy settings with limited data, addressing challenges in sample efficiency. While these advances have been applied to robotics [79], game environments [80], and autonomous driving [81], their potential for time-series forecasting remains largely unexplored.

The application of TD3 to ensemble weight determination in time-series forecasting addresses specific challenges that previous reinforcement learning approaches have not fully resolved. Unlike previous applications that focus on control tasks, our method adapts TD3 to a forecasting context where the action space represents combination weights for diverse models (Sect. 4.1.2), and the state space encapsulates both historical performance and current time-series characteristics. This adaptation requires careful consideration of the reward function, which balances general accuracy (sMAPE) with context-specific performance (wsMAE).

More recent work by [82] introduced RLMC (Reinforcement Learning for Model Combination), which uses RL to dynamically combine forecasts from multiple models. While conceptually similar to aspects of our approach, RLMC relies on standard Deep Deterministic Policy Gradient (DDPG) algorithms without addressing the inherent overestimation bias and variance issues that TD3 tackles. Furthermore, RLMC does not incorporate contextual weights in its reward function, limiting its adaptability to varying error impacts.

The growing interest in RL-based ensemble methods is evidenced by recent developments. [83] proposed a dynamic ensemble approach for probabilistic time-series forecasting using deep reinforcement learning, demonstrating the value of adaptive weight adjustment in uncertainty quantification. [84] introduced an ensemble deep reinforcement learning model for short-term load forecasting based on Q-learning dynamic model selection, showing effectiveness in energy applications. A comprehensive survey by [85] and [86] confirms that combining reinforcement learning with ensemble methods represents a promising research direction, particularly for determining optimal model weights in non-stationary environments.

[87] proposed a semi-supervised reinforcement learning approach for demand forecasting, using unlabeled data to improve model generalization. Their method demonstrates how RL can leverage partial information but focuses on using RL to improve traditional forecasting methods rather than fundamentally rethinking the forecasting architecture itself. Similarly, [88] developed a temporal difference learning approach for time-series, but their focus was on representation learning rather than forecast combination.

3.3 Advanced ensemble methods

Ensemble methods have a rich history in time-series forecasting, dating back to the pioneering work of [89], who demonstrated that simple averaging of forecasts often outperforms individual models. This insight has been repeatedly confirmed in forecasting competitions [90, 91] and theoretical analyses [92, 93].

Traditional ensemble approaches in time-series forecasting include simple averaging [94], weighted averaging based on historical performance [95], and Bayesian model averaging [96, 97]. These methods typically employ static weights or simple updating rules that do not fully account for changing dynamics and pattern shifts in time-series data.

More advanced ensemble techniques include online learning methods such as the Hedge algorithm [98, 99] and Follow the Regularized Leader [100], which dynamically adjust weights based on recent performance. [101] demonstrated the effectiveness of these approaches in time-series contexts, showing improvements over static ensemble methods. [102] introduced the aggregation of nonlinearly enhanced experts for electricity load forecasting, demonstrating how neural network-based expert enhancement can improve ensemble performance. However, these approaches typically focus on univariate time-series and do not leverage cross-series information.

Neural network-based ensemble methods have gained prominence in recent years. [103] won the M4 forecasting competition with a hybrid approach combining exponential smoothing with recurrent neural networks [90], demonstrating the potential of combining statistical and deep learning methods. [104] introduced N-BEATS, an interpretable deep learning architecture that decomposes time-series into trend and seasonal components, showing strong performance across diverse datasets.

Transformer-based architectures have recently shown promise for time-series ensembling. [29] proposed Time-Series Transformer (TST), which uses self-attention mechanisms to model temporal dependencies and adaptively combine forecasts. [54] introduced Autoformer, which decomposes time-series using autocorrelation and employs a frequency-enhanced attention mechanism. Both approaches demonstrate the potential of transformer architectures for capturing complex patterns but do not explicitly address the challenge of balancing local and global information.

Our work extends these ensemble approaches in several key ways. First, we frame ensemble weight determination as a reinforcement learning problem, allowing for adaptive, context-aware weighting based on both historical performance and current conditions. Second, we integrate local and global models through a hierarchical architecture that captures both series-specific patterns and cross-series dependencies. Third, we incorporate contextual weighting in the evaluation metrics, enabling the ensemble to adapt to varying error impacts across different parts of the forecast.

3.4 Research gaps

Our review of existing work [Anonymous] revealed important gaps that *T2f* addresses:

1. **Limited Integration of Learning Paradigms**: Existing methods often treat local and global pattern recognition as separate problems, failing to create a unified framework that effectively balances these complementary signals. While approaches like DeepGLO [50] and *Deep Factors* [51] incorporate both dimensions, they typically rely on fixed architectures rather than adaptive, learning-based integration. *T2f* addresses this gap

through its two-stage reinforcement learning approach that dynamically determines optimal weights for combining local and global models, adapting to the relative importance of each signal based on data characteristics.

2. **Insufficient Contextual Awareness**: Most forecasting methods employ uniform error metrics that treat all predictions equally, regardless of their practical importance or downstream impact [105, 106]. Unlike approaches such as RLMC [74] that optimize for general accuracy, *T2f* introduces context-specific weighting through its wsMAE metric, enabling more nuanced evaluation and learning that aligns with application-specific requirements.
3. **Scalability Challenges**: Current solutions for multi-series forecasting often face computational bottlenecks when scaled to industrial settings [107, 108]. Where methods like MTNet [52] require extensive computational resources, *T2f* addresses this through its efficient model pool approach, reusing a limited number of base models while optimizing their combination through lightweight TD3 training.
4. **Inadequate Handling of Pattern Shifts**: Existing methods struggle to adapt to changing dynamics in time-series data, particularly when the relevance of different patterns evolves over time [109, 110]. Transformer-based approaches such as Informer [53], Autoformer [54], and even recent advances like iTransformer [57] and PatchTST [56] capture complex patterns, but lack mechanisms for dynamic adaptation. While foundation models like Chronos [60] offer generalization capabilities, they do not provide the context-specific adaptation that operational forecasting often requires. *T2f*'s actor-critic architecture provides a mechanism for continuous adaptation to the importance of the changing pattern, with the critic network implicitly modeling the value of different actions under varying conditions.

By addressing these research gaps, *T2f* contributes to the wider field of time-series forecasting, integrating insights from reinforcement learning, ensemble methods, and transformer architectures to create an adaptive framework for multi-series prediction.

4 Methodology

T2f approaches time-series forecasting as a composition of local and global elements through a two-stage actor-critic architecture. Following the previous problem setting, this section details the core components of our method, its learning process, and the specific mechanisms for combining local and global patterns.

We formulate ensemble weight determination as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:

- **State space** $\mathcal{S}$: Each state $s_t = [h_t^{(L)}, \hat{y}_t^{(1:M)}, \mathcal{C}_g^{(t)}]$ comprises encoded local features $h_t^{(L)}$ from the Dilated CNN encoder, predictions $\hat{y}_t^{(1:M)}$ from $M = 5$ local models, and global covariates $\mathcal{C}_g^{(t)}$.
- **Action space** $\mathcal{A}$: Continuous ensemble weights $a_t = [w_1, ..., w_M] \in [0, 1]^M$ with $\sum_{i=1}^{M} w_i = 1$.
- **Transition dynamics** $\mathcal{P}$: Deterministic given the data—advancing the temporal window yields the next state s_{t+1} independent of a_t.
- **Reward function** $\mathcal{R}$: $r_t = -0.5 \cdot \text{sMAPE}(y_t, \hat{y}_t) - 0.5 \cdot \text{wsMAE}(y_t, \hat{y}_t, w_t^{(\text{ctx})})$, where $\hat{y}_t = \sum_i a_t^{(i)} \hat{y}_t^{(i)}$.
- **Discount factor**: $\gamma = 0.99$.

The state satisfies the Markov property approximately: the CNN encoder captures sufficient historical context through exponentially expanding receptive fields, while local model predictions summarize each model's assessment given all available history.

We choose full RL over simpler alternatives for several reasons. Contextual bandits treat each weighting decision independently, forgoing temporal credit assignment across the forecasting horizon—our discount factor $\gamma = 0.99$ captures how weight choices at time t influence cumulative forecasting quality. Online convex optimization methods provide regret bounds but assign weights based solely on past losses, without conditioning on the rich state representation (encoded history, model predictions, covariates) that allows regime-dependent decisions.

Gradient-based stacking learns fixed weights on a validation set, which our ablation study confirms is suboptimal: the ValidationWeighted baseline underperforms T2f by 14.5% on sMAPE (Table 5), precisely because static weights cannot adapt to varying temporal regimes. TD3 specifically suits this problem due to the continuous action space (ensemble weights), while its twin critics and delayed updates mitigate overestimation bias inherent in actor-critic methods operating over long forecasting horizons.

4.1 Architecture, implementation, and component integration

T2f approaches time-series forecasting through a dual-stage architecture that integrates local model predictions with a global TD3-based optimizer. This hierarchical approach first processes individual series through multiple complementary local models to capture series-specific characteristics, then integrates these predictions through a global reinforcement learning mechanism that discovers optimal combination weights. The architecture is designed to systematically address the tension between capturing local granular patterns and leveraging cross-series information through global context. This separation of concerns allows specialized handling of different aspects of the forecasting problem while maintaining end-to-end trainability through reinforcement learning signals.

The overall forecasting function can be expressed mathematically as:

$$\mathcal{F}(x) = \mathcal{G}(\mathcal{L}(x, \theta_L), \theta_G) \tag{8}$$

where $\mathcal{L}$ represents the local stage processing, $\mathcal{G}$ represents the global stage processing, and θ_L, θ_G are the respective learning parameters.

4.1.1 Core components

The framework integrates several interconnected components:

1. **Adaptive Sequence Handling**: The adaptive sequence mechanism enables processing of variable-length input sequences, a common challenge in real-world time-series data. This is accomplished through a three-stage process:

$$h_t = \text{AdaptiveAvgPool1d}(\text{Sequential}(\text{x}_{t-k:t})) \tag{9}$$

   First, input features are processed through fully-connected layers with nonlinear activations. Then, regardless of input sequence length, an adaptive pooling layer aggregates the processed features into a fixed-size representation. Finally, this representation is transformed into forecasts of the desired length, ensuring consistent output dimensions even with irregular inputs.

2. **Dilated Convolutions**: The dilated CNN encoder implements a sequence of convolutional layers with exponentially increasing dilation rates:

$$\text{dilation}_l = \text{base}^l \tag{10}$$

   where the base is set to 2 in the implementation. This exponential increase in receptive field allows the model to capture both short-term patterns and long-range dependencies without requiring deep architectures or recurrent connections.

3. **Double Q-Learning**: The TD3 implementation reduces overestimation bias through dual critics:

$$Q_{\text{target}} = r + \gamma \min(Q_1'(s', \pi'(s')), Q_2'(s', \pi'(s'))) \tag{11}$$

By taking the minimum of two critics' estimates, the model obtains more conservative value approximations, reducing the common tendency of Q-learning to overestimate action values.

4. **Softmax Fallback**: When Q-clipping might be too aggressive, a softmax fallback ensures valid probability distributions:

$$a_{\text{final}} = \begin{cases} \text{softmax}(a), & \text{if } a \approx a_{\text{clipped}} \\ a_{\text{clipped}}, & \text{otherwise} \end{cases} \tag{12}$$

This mechanism provides a safeguard against extreme weight values while ensuring that the ensemble weights always form a valid probability distribution.

The integration of these components enables *T2f* to effectively balance local pattern recognition with global contextual awareness, while the feedback mechanisms allow continuous adaptation to changing patterns in the data.

4.1.2 Actor-critic framework and TD3 integration

The core of *T2f*'s decision-making capability lies in its Actor-Critic framework based on the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [27]. This architecture addresses the fundamental challenge of determining optimal combination weights for multiple forecasting models in a dynamic environment.

The Actor network architecture consists of three primary components: an encoder layer (128 units) that processes state information, a fully connected hidden layer (64 units), and an action output layer that generates weights for each local model. The network includes a maximum action constraint (set to 1) and uses ReLU activations between layers. The forward pass first applies the encoder, followed by the hidden layer, and finally produces action values bounded by a tanh activation multiplied by the maximum action parameter:

$$a_t = \pi_\theta(s_t) \tag{13}$$

where θ represents the actor network parameters and s_t is the current state composed of local model predictions and global context features.

In cases where the network produces very small values ($|x| < 10^{-3}$), a softmax activation is automatically applied to ensure valid probability outputs. The dual Critic networks, a key feature of TD3, estimate the value of state-action pairs through Q-functions $Q_1(s, a)$ and $Q_2(s, a)$. By maintaining two independent critics, *T2f* mitigates the overestimation bias common in Q-learning approaches [75, 111]:

$$Q_{\text{target}} = r_t + \gamma \min(Q_1'(s_{t+1}, \pi'(s_{t+1})), Q_2'(s_{t+1}, \pi'(s_{t+1}))) \tag{14}$$

where Q_1' and Q_2' are target networks that stabilize learning, γ is a discount factor, and r_t is the reward signal derived from forecast accuracy metrics.

Each Critic network uses an identical three-layer architecture: a 128-unit first fully connected layer, a 64-unit second layer, and a single output unit. The key architectural feature is that state and action information are concatenated along the feature dimension as input to the first layer:

$$Q(s, a) = f_{c3}(f_{\text{relu}}(f_{c2}(f_{\text{relu}}(f_{c1}([s, a]))))) \tag{15}$$

where $[s, a]$ represents the concatenation of state and action vectors. This concatenation approach, inspired by [73] and adapted for TD3 by [27], allows the critic to evaluate state-action pairs jointly.

The Double Q-Clipping mechanism further enhances stability by restricting weight updates based on critic estimates:

$$a_{\text{clipped}} = \text{clip}(a_t, Q_{\min} - \delta, Q_{\max} + \delta) \tag{16}$$

The softmax fallback for clipping ensures that ensemble weights are both stable and form valid probability distributions. The implementation includes several TD3-specific optimizations:

- **Delayed policy updates**: The actor network is updated less frequently than the critics (every 2 steps; `policy_freq = 2`), reducing variance in policy updates.
- **Target policy smoothing**: Small noise is added to target actions during critic updates, preventing the exploitation of narrow peaks in the value landscape.
- **Soft target updates**: Network parameters are updated gradually using a soft update coefficient (`tau = 0.005`), ensuring stable learning dynamics [73].

These mechanisms collectively enable *T2f* to learn effective model combination strategies [70, 71] that adapt to both changing patterns in individual series and evolving relationships between multiple series.

The implementation follows a specific update sequence for each training iteration. First, both critic networks are updated using the same target computed from the minimum of the target critic values:

$$Q_{\text{target}} = r_t + \gamma(1 - \text{done}_t)\min(Q'_1(s_{t+1}, \pi'(s_{t+1}) + \epsilon), Q'_2(s_{t+1}, \pi'(s_{t+1}) + \epsilon)) \tag{17}$$

where $\epsilon \sim \text{clip}(\mathcal{N}(0, \text{policy\_noise}), -\text{noise\_clip}, \text{noise\_clip})$ implements target policy smoothing with $\text{policy\_noise} = 0.2$ and $\text{noise\_clip} = 0.5$. The delayed policy updates in actor network use the gradient:

$$\nabla_{\theta_\pi} J(\theta_\pi) \approx -\nabla_a Q_1(s, a)|_{a=\pi(s)} \nabla_{\theta_\pi} \pi(s) \tag{18}$$

After each actor update, all three target networks (actor and both critics) undergo soft updates with $\tau = 0.005$:

$$\theta_{\text{target}} \leftarrow \tau\theta + (1 - \tau)\theta_{\text{target}} \tag{19}$$

This specific update sequence ensures stable learning dynamics while contributing to mitigating the overestimation bias inherent in Q-learning approaches [27].

4.1.3 Algorithmic learning process

The learning process in *T2f* operates through two coordinated stages, as outlined later in Table 1. In the local stage, individual forecasting models generate predictions for each time-series using temporal attention weights. In the global stage, the TD3 reinforcement learning framework learns optimal combination weights by interacting with the environment through a replay buffer and dual critics.

This dual-stage architecture balances local pattern recognition with global contextual awareness, while the TD3 framework ensures stable learning through the conservative value estimation and delayed policy updates described before. The integration of these components enables *T2f* to effectively handle variable-length sequences, capture multi-scale patterns, and adapt to changing dynamics across diverse time-series.

The diagram in Fig. 1 provides a visual representation of the algorithmic learning process behind *T2f*, considering the key architectural components outlined before:

- **Local Model Pool** (upper section, left panel): Contains diverse models which process input sequences independently and generate local predictions.
- **Dilated CNN encoder** (features extraction box): Processes input sequences and enables efficient capture of long-range dependencies.

Table 1 Pseudo-algorithm for T2f's learning process

T2fPseudo-Algorithm

Input: time-series collection $\mathcal{S} = \{S_1, ..., S_n\}$, local window size p, forecast horizon F, covariates $\mathcal{C} = \{\mathcal{C}_g \cup \mathcal{C}_l\}$

Output: Forecasts $\hat{y}_{t+1:t+F}$ for each series S_i

1. Initialize:

Local model pool $\mathcal{M}_L = \{m_{L_1}, ..., m_{L_M}\}$

Global model pool $\mathcal{M}_G = \{m_{G_1}, ..., m_{G_M}\}$

Actor networks $\pi_L(s), \pi_G(s)$ with parameters $\theta_{\pi_L}, \theta_{\pi_G}$

Critic networks $Q_{L_1}, Q_{L_2}, Q_{G_1}, Q_{G_2}$ with parameters $\theta_{Q_{L_i}}, \theta_{Q_{G_i}}$

Target networks $\pi'_L, \pi'_G, Q'_{L_1}, Q'_{L_2}, Q'_{G_1}, Q'_{G_2}$

Replay buffers $\mathcal{D}_L, \mathcal{D}_G$

2. for each series S_i do

3. for t in $1, ..., T_i$ do

4. Get state $s_{L_t} = [\mathrm{x}^{(i)}_{t-p:t}, \mathcal{C}^{(i)}_l]$

5. Select local action $a_{L_t} \sim \pi_L(s_{L_t})$

6. Get local predictions $\{\hat{y}^{(m)}_{L_t}\}_{m=1}^{M}$ from $\mathcal{M}_L$

7. Compute local prediction $\hat{y}_{L_t} = \sum_{m=1}^{M} a^{(m)}_{L_t} \cdot \hat{y}^{(m)}_{L_t}$

8. Compute local reward $r_{L_t} = -0.5 \cdot \mathrm{sMAPE}_t - 0.5 \cdot \mathrm{wsMAE}_t$

9. Store transition $(s_{L_t}, a_{L_t}, r_{L_t}, s_{L_{t+1}})$ in $\mathcal{D}_L$

10. Get global state $s_{G_t} = [h^{(i)}_{L_t}, \mathcal{C}_g]$

11. Select global action $a_{G_t} \sim \pi_G(s_{G_t})$

12. Get global predictions $\{\hat{y}^{(m)}_{G_t}\}_{m=1}^{M}$ from $\mathcal{M}_G$

13. Compute global prediction $\hat{y}_{G_t} = \sum_{m=1}^{M} a^{(m)}_{G_t} \cdot \hat{y}^{(m)}_{G_t}$

14. Compute global reward $r_{G_t} = -0.5 \cdot \mathrm{sMAPE}_t - 0.5 \cdot \mathrm{wsMAE}_t$

15. Store transition $(s_{G_t}, a_{G_t}, r_{G_t}, s_{G_{t+1}})$ in $\mathcal{D}_G$

16. Sample mini-batches from $\mathcal{D}_L$ and $\mathcal{D}_G$

17. Update critic networks Q_{L_i}, Q_{G_i} using TD3 loss with target: $Q_{\mathrm{target}} = r_t + \gamma(1 - \mathrm{done}_t) \min(Q'_1, Q'_2)$

18. if $t \bmod \mathrm{policy\_freq} = 0$ then

19. Update actor networks π_L, π_G using gradient: $\nabla_{\theta_\pi} J \approx -\nabla_a Q_1(s, a)|_{a=\pi(s)} \nabla_{\theta_\pi} \pi(s)$

20. Update target networks using soft update: $\theta_{\mathrm{target}} \leftarrow \tau\theta + (1 - \tau)\theta_{\mathrm{target}}, \tau = 0.005$

21. return Global forecasts $\hat{y}_{G_{t+1:t+F}}$ for each series S_i

- **FEDFormer inspired layer** (global model section): Explores the tendency of time-series to have sparse representation in Fourier space, extracting the top-k frequency components through Fast Fourier Transform (FFT) and projecting them into a learned embedding space.
- **Actor-Critic Networks** (right panel): TD3 architecture with dual critics. The Actor Network generates model weights while the Dual Critics estimate value functions to guide the optimization process.
- **Double Q-Clipping** (red box, right panel): Stabilizes weight generation by clipping based on critic outputs.
- **Ensemble Combination** (bottom box, left panel): Combines predictions based on the weights generated by the actor network, producing the final forecast.

The arrows in Fig. 1 indicate the flow of information through the system: direct flow (solid lines) represents the standard forward pass, information transfer (dashed lines) shows how local features influence the actor network, and weight transfer (red dashed line) illustrates how the generated weights determine the final ensemble prediction.

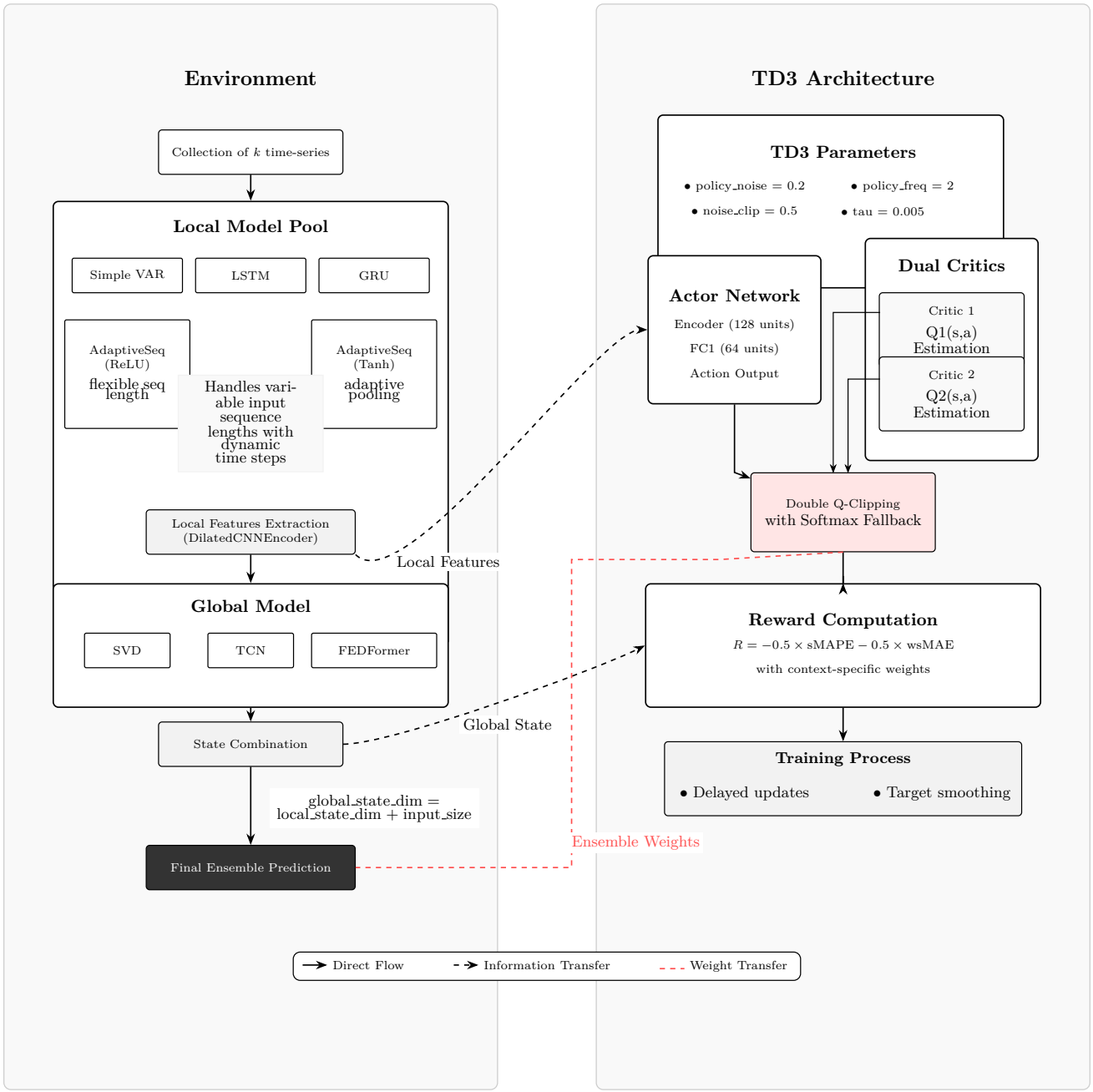


Fig. 1 *T2f* architecture showing the environment with local and global model pools (left) and the TD3-based actor-critic framework (right). The diagram illustrates how local features feed into the actor network, global state influences reward computation, and ensemble weights determine the final prediction

4.2 Model candidate selection

To ensure efficient exploration, *T2f* employs a fixed number of candidate models at the local level. In our implementation, the local model stack $\mathcal{M}_L = \{m_{L_1}, ..., m_{L_M}\}$ consists of:

- **Simple VAR**: A Vector Autoregression model that captures linear dependencies and serves as a stable baseline

- **LSTM and GRU Networks**: Recurrent neural networks that model complex temporal patterns and long-term dependencies
- **Adaptive Sequential Model**: Custom sequential models with varying activation functions (ReLU and Tanh) that handle variable-length input sequences through adaptive pooling

Each model processes input data of dimension $\mathcal{L}$ (local length) and generates series-specific predictions. The custom $\mathrm{AdaptiveSeq}$ variants deserve particular mention for their ability to handle variable-length inputs through a combination of adaptive pooling and flexible architecture, making them robust to irregular sampling or missing data. This approach is inspired by work on adaptive neural network architectures [112, 113], but modified specifically for the forecasting context to handle missing and irregular data points.

The local models thus collectively capture different aspects of the time-series, from linear dependencies (*VAR*) to complex nonlinear patterns (*LSTM*/*GRU*) and adaptive representations (*AdaptiveSeq*).

4.3 Action-state space design

The action space at time step t represents the ensemble weights assigned to each local model:

$$a_t = [a_{m_1}, ..., a_{m_M}] \tag{20}$$

where each action component represents a probabilistic weight. Following the Actor-Critic framework in Sect. 4.1.2, these weights are generated through the actor network and refined using the double Q-clipping mechanism with softmax fallback when appropriate.

4.4 Learning process

Following Sect. 2.3, the learning process in *T2f* occurs in two distinct stages:

4.4.1 Local stage processing

In the local stage, individual forecasting models are trained and applied to each time-series:

$$\hat{y}_{L_i,t} = m_{L_i}(X_t \cdot \alpha_t) \tag{21}$$

where α_t are the temporal attention weights that prioritize more recent or more relevant observations. Each model's predictions are stored and passed to the global stage.

4.4.2 Global stage processing

The global stage employs the TD3-based Actor-Critic framework as described in Sect. 4.1.2 to learn optimal combination weights:

$$\hat{y}_{G_t} = \sum_{i=1}^{M} a^i_{G_t} \cdot \hat{y}_{L_i,t} \tag{22}$$

where $a^i_{G_t}$ represents the weight assigned to the i-th local model's prediction.

The learning process uses a circular replay buffer implementation with fixed maximum size (defaulted to 1000 in our implementation, to be adjusted based on batch size requirements). The buffer stores transitions as tuples of $(s_t, a_t, s_{t+1}, r_t, \mathrm{done}_t)$ and provides sampling functionality with automatic batch size adjustment when requested samples exceed buffer size:

$$\text{batch\_size}_{\text{actual}} = \min(\text{batch\_size}_{\text{requested}}, |\mathcal{D}|) \tag{23}$$

where $|\mathcal{D}|$ represents the current buffer size. Training only proceeds when at least 2 samples are available, ensuring meaningful gradient updates.

4.5 Policy reward structure

T2f implements a combined reward function that balances forecast accuracy with contextual relevance:

$$r_t = -0.5 \cdot \text{sMAPE}(y_t, \hat{y}_t) - 0.5 \cdot \text{wsMAE}(y_t, \hat{y}_t, w_t) \tag{24}$$

This reward structure feeds directly into the TD3 learning process outlined in Sect. 4.1.2, guiding the actor-critic networks to optimize both general accuracy and context-specific performance. The negative sign reflects that lower error values correspond to better performance.

5 Experimental settings

We evaluate *T2f* using five distinct datasets that represent diverse forecasting scenarios: (1) Household Power Consumption (HPC) dataset [114], (2) Electricity Load Diagrams (ELD) dataset [115], (3–4) ETT-h and ETT-m datasets from the Electricity Transformer Temperature series [53], and (5) a synthetic dataset designed to test specific edge cases in time-series forecasting.

5.1 Implementation details

All experiments were conducted on an AWS SageMaker ml.m5.8xlarge instance, using Python 3.9 with PyTorch 1.9.0 [116]. The full source code, including all implementation details and experiment configurations, will be made publicly available upon acceptance. The model architecture employs the following key parameters:

- **Local window length** (`local length`): Controls the sequence length for local models. Values were tested from {2, 6, 8, 12, 30} based on dataset characteristics.
- **Global window length** (`global length`): Determines the forecast horizon. Tested values included {1, 3, 6, 8, 12, 35}.
- **Hidden size**: Calculated dynamically based on input dimension and data characteristics:

$$\text{hidden\_size} = \max(64, \min(2^{\lfloor \log_2(b \cdot s_f \cdot t_f) + 0.5 \rfloor}, 512)) \tag{25}$$

where b is the base size (covariate count $\times$ 4), s_f is a series complexity factor, and t_f is a temporal complexity factor.

- **Number of layers**: Adapted to data complexity:

$$\text{num\_layers} = \text{base\_layers} \cdot \text{feature\_factor} \tag{26}$$

with *base*_layers calculated from the logarithm of the median series length.

- **Training epochs**: Values from {2, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50} were tested, with early stopping based on validation performance.
- **Batch size**: Fixed at 32 for most experiments, with automatic adjustment for smaller datasets to prevent overfitting.

5.2 Dataset selection and characteristics

The evaluation of *T2f*'s performance required a selection of datasets that span the diversity of real-world forecasting challenges. We deliberately chose five datasets with distinct statistical properties, temporal dynamics, and domain contexts to provide a comprehensive assessment of forecasting capabilities.

5.2.1 Dataset overview

The Household Power Consumption (HPC) dataset [114] contains measurements of electric power consumption in a single household with one-minute sampling rate over almost 4 years. This dataset includes 2,075,259 measurements of different electrical quantities and sub-metering values collected from a house in Sceaux, France, between December 2006 and November 2010. The dataset exhibits several challenging characteristics: variations in daily usage patterns, multi-periodicity (daily, weekly, and seasonal patterns), and different correlations between various electrical measurements. With measurements including global active power, global reactive power, voltage, and sub-metering values, the HPC data tests a model's ability to capture both short-term consumption patterns and long-term dependencies. The dataset includes covariates such as Global_active_power, Global_reactive_power, Global_intensity, Voltage, and diff_order, with Voltage serving as the target variable and metering values providing contextual weights for error evaluation through wsMAE metrics.

The Electricity Load Diagrams (ELD) dataset [115] contains electricity consumption data for 370 clients collected over the period 2011-2014. The dataset comprises 15-minute interval measurements in kilowatts, with each client represented as a separate series, creating a rich collection of related time-series with diverse consumption patterns. With 140,256 features representing timestamps and no missing values, this dataset presents a comprehensive view of electricity usage across different consumers. The ELD dataset features strong daily and weekly seasonality patterns, influenced by business hours, weekday/weekend differences, and seasonal variations. Temperature metrics serve as covariates, with electricity load as the target variable, and cost per kilowatt-hour providing contextual weights that incorporate economic factors into the performance assessment.

The Electricity Transformer Temperature datasets ETT-h (hourly) and ETT-m (15-minute) [53] offer complementary views of the same physical systems at different temporal resolutions. These datasets present unique challenges: strong exogenous influences (ambient temperature, load), complex seasonal patterns, and significant auto-correlation at multiple lags. By including both temporal resolutions, we test *T2f*'s adaptability to different sampling frequencies and its ability to detect and exploit periodicity at multiple scales. The long-term dependencies in transformer behavior (where current temperature depends on hours or days of previous operation) challenge models to capture extended causal relationships.

Our synthetic dataset was specifically designed to test forecasting robustness under controlled statistical conditions. The dataset was generated using a two-stage process: first creating a base dataset with 500 samples per time point and 30 feature dimensions, then applying a series of transformations to control statistical properties. The data spans a weekly time series across 3 years, with each series transformed to target Box-Cox normality. We particularly focused on stationarity characteristics by implementing a differencing procedure that adaptively processes each series until it passes the Augmented Dickey-Fuller test, with differencing orders tracked and limited to a maximum of 6. This creates heterogeneous time series with varied statistical properties while maintaining known ground truth relationships. The aim of this controlled synthetic environment is to complement the real-world datasets by isolating specific statistical challenges common in forecasting tasks.

5.2.2 Dataset representation matrix

Collectively, these datasets form a representation matrix across key time-series dimensions as outlined in Table 2.

This diverse representation ensures that our evaluation framework challenges forecasting models across the full spectrum of time-series behaviors, rather than focusing on narrow performance domains. The deliberate

Table 2 Dataset Characteristic Matrix

Characteristic	HPC	ELD	ETT	Synthetic
Temporal Irregularity	High	Low	Low	Low
Multi-Seasonality	Moderate	High	High	Controlled
Volatility	Moderate	Moderate	Low	Mixed
Cross-Series Correlation	Moderate	Heterogeneous	Homogeneous	Explicit
Exogenous Dependencies	Moderate	High	High	Explicit
Regime Changes	Occasional	Occasional	Rare	Controlled

inclusion of both real-world and synthetic data provides complementary insights: real data tests practical applicability, while synthetic data verifies specific algorithmic capabilities.

For each dataset, we utilized specific covariates and weights in our experimental evaluation. In the HPC dataset, electrical measurements including `Global_active_power, Global_reactive_power, Global_intensity, Voltage, diff_order` served as covariates, with `Voltage` as the target variable and `metering` providing contextual weights for error evaluation. The ELD dataset used temperature metrics (`tmax, tmin, tmed`) as covariates, `load` as the target variable, and `cost_kwh` (electricity cost per kilowatt-hour) as weights, directly incorporating economic factors into performance assessment. For the ETT datasets, transformer measurements (`HULL, MUFL, MULL, LUFL, LULL, HUFL`) were used as covariates with `OT` (oil temperature) as the target variable and `HULL` serving as the weight factor. While our theoretical framework distinguishes between local and global covariates, our implementation allows the two-stage architecture of *T2f* to implicitly learn this distinction, with the local model stage capturing series-specific patterns and the global stage identifying cross-series dependencies without requiring explicit categorization of covariates.

All datasets underwent consistent preprocessing to ensure fair comparison: (1) temporal alignment through native frequency resampling, (2) missing value handling using forward filling (limit=2), backward filling (limit=2), and linear interpolation, (3) feature normalization with Standard Scaling, and (4) strict temporal train-test splitting with 80% for training and 20% for testing. This preserves the sequential nature of the forecasting problem while enabling meaningful cross-model comparison.

5.3 Model parameters and experimental configuration

To ensure reproducible comparison across different forecasting models, we implemented a standardized benchmarking framework that maintains consistent evaluation conditions while respecting each model's specific requirements.

5.3.1 Parameter selection philosophy

Our experimental approach followed three guiding principles for parameter selection:

First, we prioritized default parameters as recommended in each model's original implementation or literature. This approach minimizes potential bias from excessive tuning and reflects real-world usage scenarios where exhaustive hyperparameter optimization may be impractical. Second, we maintained structural parity across models where applicable, ensuring that comparable models received similar capacity (e.g., hidden dimensions, layer counts). Third, we applied consistent preprocessing and evaluation across all models, using identical train-test splits, data normalization approaches, and performance metrics.

5.3.2 *T2f* configuration

The *T2f* model's core parameters were determined as per Sect. 5.1. The Actor-Critic configuration and network dimensioning were covered in Sect. 4.1.2.

5.3.3 Local model pool configuration

The local model pool in *T2f* consists of five complementary models designed to capture different aspects of time-series dynamics. Each model is constructed with specific architectural considerations to ensure robust performance across diverse forecasting scenarios:

- **Simple VAR**: Implemented as a lightweight Vector Autoregression model using pseudo-inverse for numerical stability:

$$\text{coefficients} = \text{pinv}(X_{\text{lagged}}) \cdot y \tag{27}$$

The lag parameter is adaptively set to $\min(\text{len}(X) - 1, \max(1, \text{len}(X)/4))$, balancing model complexity with data availability.

- **Wrapped LSTM**: A recurrent neural network with parameters $\{\text{input\_size}, \text{hidden\_size}, \text{num\_layers}, \text{output\_size}, \text{output\_length}\}$ that handles both single timesteps and sequence data through conditional reshaping:

$$\text{if } \text{len}(x.\text{shape}) = 1 : x = x.\text{unsqueeze}(0).\text{unsqueeze}(0) \tag{28}$$

- **Wrapped GRU**: Similar to the LSTM but with gated recurrent units, requiring less computation while maintaining temporal modeling capacity. For single-point inputs, the model employs output repetition:

$$\text{if output.shape}[1] = 1 : \text{output} = \text{output.repeat}(1, \text{output\_length}, 1) \tag{29}$$

- **AdaptiveSequentialModel (ReLU)**: A flexible architecture that handles variable-length inputs through a three-component design:

$$\text{input\_layer} = \text{Sequential}(\text{Linear}, \text{ReLU}, \text{Linear}) \tag{30}$$

$$\text{global\_pool} = \text{AdaptiveAvgPool1d}(1) \tag{31}$$

$$\text{output\_layer} = \text{Sequential}(\text{Linear}, \text{ReLU}, \text{Linear}) \tag{32}$$

This structure first extracts features, then applies adaptive pooling to produce fixed-length representations regardless of input length, and finally generates forecasts of the required horizon.

- **AdaptiveSequentialModel (Tanh)**: Identical to the ReLU variant but employs tanh activation functions, providing complementary modeling capabilities for data with different characteristics, particularly when values need to be bounded.

Each model processes inputs weighted by temporal attention weights, and their outputs are collectively used to form the local state representation:

$$\text{local\_state\_dim} = \text{num\_models} \times \text{local\_length} \times \text{output\_size} \tag{33}$$

5.4 Benchmark models

We compare *T2f* against a comprehensive set of baseline models selected to represent the spectrum of forecasting approaches from classical statistical methods to advanced deep learning architectures [Anonymous]. This selection enables evaluation of *T2f*'s performance against established methods across different forecasting paradigms:

- **Classical statistical models**: AUTOARIMA [117], GARCH [118] — included for their established performance and widespread use in traditional time-series forecasting
- **Deep learning approaches**: N-BEATS [119], GRU [37], TCN [16] — representing neural architectures that excel at capturing complex patterns
- **Transformer-based approaches**: PatchTST [56], iTransformer [57], TimeMixer [58] — recent architectures leveraging attention mechanisms and multi-scale decomposition
- **Hybrid models**: Prophet [120], DeepAR (both local and global variants) [9] — combining statistical components with machine learning techniques
- **Ensemble baselines**: SimpleEnsemble (equal weights), ValidationWeighted (inverse validation error weights), and RLMC [82] — representing different ensemble aggregation strategies, with RLMC using DDPG-based reinforcement learning for weight optimization

Each model was fine-tuned using model-specific hyperparameter optimization following recommendations from their respective authors. The model configurations adopted are further detailed below:

Statistical Methods:

- **AUTOARIMA**: Implemented using the `pmdarima` package [121] with automatic order selection. Default settings included suppressed warnings and a fallback to a constant mean model for edge cases, providing robust baseline performance without manual intervention.
- **GARCH**: Implemented using the `arch` package [122] with automatic order selection up to (5,5) using AIC criterion. The implementation included volatility clustering detection and adaptive scaling for numerical stability.**Deep Learning Methods:**

- **N-BEATS**: Configured following [104] with a simplified architecture using the same hidden size as *T2f*. Training used the Huber loss with delta = 1.0 and early stopping with patience of 5 epochs, providing robust performance without excessive computational demands.
- **GRU**: Implemented as a standard recurrent architecture with the same hidden size and layer count as *T2f* for structural parity. Training employed MSE loss with the Adam optimizer and default PyTorch learning rates.
- **TCN**: Configured with an adaptive channel size selection based on sequence length, with kernel size = 3 and dropout = 0.2. The architecture dynamically adjusted to data complexity through an adaptive pooling mechanism that accommodated variable-length inputs.**Hybrid Methods:**

- **Prophet**: Implemented using Facebook's Prophet library [120] with default seasonality settings and automatic changepoint detection. A fallback to Exponential Smoothing was included for cases where Prophet fitting failed.
- **DeepAR (Local)**: Configured to train separate models for each series, following the general architecture described in [9]. The model used the same hidden size and layer count as *T2f*, with a Gaussian likelihood objective and early stopping.
- **DeepAR (Global)**: Implemented as a single model trained across all series, sharing parameters to leverage cross-series information. Otherwise maintained the same architecture as the local variant for direct comparison.**Transformer-Based Methods:**

- **PatchTST**: Configured following [56] with patch length 16 and stride 8, using channel-independent processing. The model used 3 encoder layers with 4 attention heads and the same hidden size as *T2f*.
- **iTransformer**: Implemented following [57] with inverted attention applied on the variate dimension rather than the temporal dimension. Used 2 encoder layers with 8 attention heads.
- **TimeMixer**: Configured following [58] with multiscale mixing at 3 decomposition levels. Past and future mixing blocks used the same hidden size as *T2f*.**Ensemble Baselines:**

- **SimpleEnsemble**: Combines predictions from *T2f*'s five local models using equal weights ($w_i = 1/M$). Requires no training beyond the local models themselves.
- **ValidationWeighted**: Combines local model predictions using weights inversely proportional to each model's validation error: $w_i = (1/\text{MSE}_i^{\text{val}})/\sum_j (1/\text{MSE}_j^{\text{val}})$. Also requires no iterative training.

- **RLMC**: Implements the reinforcement learning for model combination approach of [82] using DDPG. Key differences from *T2f*'s TD3: single critic network (no twin), no delayed policy updates, and no target policy smoothing noise.

All deep learning models used the Adam optimizer with learning rates of 0.001 and gradient clipping to prevent exploding gradients. Standard data normalization was applied consistently across all models using MinMaxScaler for TCN and StandardScaler for others, maintaining comparable input scales. The implementation included consistent error handling and fallback mechanisms for all models, ensuring fair comparison.

5.4.1 Evaluation protocol

The evaluation protocol maintained strict separation between training and testing data, with an 80-20 temporal split ensuring that all forecasts represented true out-of-sample predictions. For each model and dataset, we:

- Trained all models on identical training data using the configurations described above
- Generated forecasts for the test period without access to test labels
- Computed identical metrics (MAE, RMSE, sMAPE, wsMAE) for all models using the same evaluation function
- Averaged results across all series and multiple runs to ensure stability

This standardized evaluation approach, combined with the controlled model configurations, ensures that the observed performance differences reflect genuine algorithmic advantages rather than implementation details or evaluation biases.

5.5 Performance metrics

We employ four complementary metrics that capture different aspects of forecasting performance:

1. Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{34}$$

2. Root Mean Square Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{35}$$

3. sMAPE and wsMAE as defined in Sect. 2.3.

The weights w_t in wsMAE are obtained directly from relevance measures assumed from the input data, allowing context-specific error weighting.

6 Key findings

Our experimental evaluation across multiple datasets and comparative models reveals several significant findings. *T2f* demonstrates competitive or superior performance in most evaluation scenarios, with particularly strong results on complex datasets featuring irregular patterns and contextually weighted metrics. The actor-critic architecture shows faster convergence than traditional ensemble methods, especially evident in the early training epochs. We observe that the model's relative advantage increases with data complexity, suggesting effective utilization of the hierarchical learning structure. Importantly, the results validate our core hypothesis that balancing local and global pattern recognition through reinforcement learning provides measurable benefits for time-series forecasting. The following sections detail these findings across different dimensions of performance.

6.1 Performance analysis

6.1.1 Cross-dataset performance comparison

The results in Tables 3 and 4 suggest several dataset-specific patterns:

- *T2f* achieves the lowest error values on the HPC dataset (MAE: 0.237, RMSE: 0.288), outperforming all competitors across all metrics. This suggests particular effectiveness in handling irregular workload patterns with complex temporal dynamics.
- On the ELD dataset, N-BEATS achieves the best MAE and RMSE scores (0.145 and 0.178 respectively), while *T2f* leads in context-weighted metrics (sMAPE: 0.191, wsMAE: 0.198), indicating different strengths in capturing regular seasonal patterns versus adapting to contextual importance.
- Across the ETT datasets, *T2f* and GRU show complementary advantages, with *T2f* leading on MAE and GRU excelling in sMAPE on both ETT variants. On ETTh RMSE, GRU achieves the best score (0.221), suggesting that recurrent architectures can be particularly effective for hourly transformer temperature patterns.
- The synthetic dataset reveals GARCH's strength (MAE: 0.245, RMSE: 0.286, wsMAE: 0.251) in controlled stochastic environments, with *T2f* providing competitive performance particularly in sMAPE (0.246).

6.1.2 Convergence behavior

Figure 2 illustrates the learning dynamics of different models through wsMAE. This analysis reveals important insights about model adaptation and learning efficiency:

On the HPC dataset (Fig. 2a), *T2f* shows distinctly faster convergence, with error dropping sharply in the first 20 epochs and reaching near-optimal performance by epoch 25. The final wsMAE (0.211) significantly outperforms the next best single model shown (GRU at 0.348). The stability after epoch 30 indicates *T2f* effectively learns the underlying patterns without overfitting.

The ELD results reveal that while N-BEATS excels in standard metrics, *T2f* achieves the best wsMAE (0.198). The steeper initial slope of *T2f*'s convergence curve compared to other models demonstrates its context-aware learning mechanism's effectiveness in weighted error scenarios.

Competing models like GARCH and DeepAR show similar convergence trajectories but stabilize at higher error rates, particularly in the HPC dataset. This performance difference appears to increase with data complexity, which may suggest that *T2f*'s actor-critic architecture (Sect. 4.1.2) offers potential benefits when handling complex, irregular time-series with varying error importance.

These findings collectively validate *T2f*'s effectiveness in balancing local and global pattern recognition while maintaining computational efficiency. The implementation's handling of temporal attention, edge cases, and adaptive model combination provides a robust foundation for diverse forecasting applications.

Table 3 Performance comparison: *T2f* vs. individual baselines (lower is better). Values represent the mean over 10 seeds; standard deviations were below 0.01 for all models and are omitted for clarity. Bold: statistically significant best across all models ($p < 0.05$, Wilcoxon signed-rank test)

Model	Metric	HPC	ELD	ETTh	ETTm	Synthetic
T2f	MAE	**0.237**	0.179	**0.215**	**0.211**	0.259
	RMSE	**0.288**	0.221	0.256	**0.253**	0.294
	sMAPE	**0.204**	**0.191**	0.223	0.218	**0.246**
	wsMAE	**0.211**	**0.198**	**0.231**	0.228	0.268
AUTOARIMA	MAE	0.343	0.314	0.326	0.351	0.348
	RMSE	0.367	0.350	0.361	0.366	0.373
	sMAPE	0.376	0.401	0.378	0.388	0.392
	wsMAE	0.319	0.354	0.335	0.319	0.405
N-BEATS	MAE	0.365	**0.145**	0.347	0.339	0.338
	RMSE	0.379	**0.178**	0.377	0.358	0.369
	sMAPE	0.357	0.388	0.375	0.344	0.421
	wsMAE	0.351	0.351	0.332	0.308	0.435
GARCH	MAE	0.387	0.349	0.365	0.381	**0.245**
	RMSE	0.397	0.354	0.370	0.427	**0.286**
	sMAPE	0.453	0.374	0.388	0.410	0.256
	wsMAE	0.373	0.330	0.345	0.435	**0.251**
Prophet	MAE	0.405	0.391	0.398	0.438	0.393
	RMSE	0.419	0.431	0.444	0.485	0.487
	sMAPE	0.389	0.378	0.392	0.483	0.442
	wsMAE	0.415	0.365	0.375	0.449	0.445
GRU	MAE	0.374	0.321	0.328	0.409	0.418
	RMSE	0.372	0.321	**0.221**	0.473	0.488
	sMAPE	0.349	0.361	**0.211**	**0.211**	0.497
	wsMAE	0.348	0.358	0.346	**0.208**	0.507
TCN	MAE	0.512	0.417	0.428	0.441	0.453
	RMSE	0.596	0.528	0.542	0.519	0.501
	sMAPE	0.547	0.480	0.492	0.516	0.467
	wsMAE	0.565	0.451	0.464	0.523	0.450
DeepAR (local)	MAE	0.374	0.342	0.352	0.398	0.380
	RMSE	0.365	0.358	0.375	0.392	0.419
	sMAPE	0.416	0.363	0.371	0.370	0.408
	wsMAE	0.396	0.320	0.336	0.327	0.422
DeepAR (global)	MAE	0.385	0.357	0.366	0.461	0.385
	RMSE	0.352	0.360	0.378	0.452	0.423
	sMAPE	0.434	0.403	0.411	0.415	0.450
	wsMAE	0.449	0.427	0.442	0.441	0.405

6.2 Ablation study

To understand the individual contribution of each component to *T2f*'s performance, we conducted an ablation study systematically disabling key architectural elements: the TD3-based global optimizer, temporal attention mechanism, and FEDFormer-inspired frequency layer. Results are presented in Table 5.

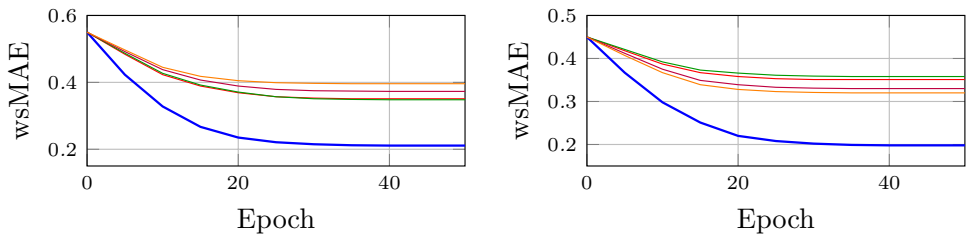
The ablation results reveal several important findings about *T2f*'s architecture:

TD3 Global Optimizer: Disabling the TD3-based weight optimization and using uniform weights across local models results in the largest performance degradation (+18.8% sMAPE, +20.6% wsMAE). This confirms that the reinforcement learning-based dynamic weight determination is the most critical component, validating our core contribution of framing ensemble combination as a sequential decision problem.

Temporal Attention: Removing the temporal attention mechanism leads to moderate degradation (+8.0% sMAPE, +10.3% wsMAE). The larger impact on wsMAE compared to sMAPE suggests that temporal attention

Table 4 Performance comparison: ensemble methods and recent baselines (lower is better). *T2f* results repeated for reference. Bold: statistically significant best across all models ($p < 0.05$, Wilcoxon signed-rank test)

Model	Metric	HPC	ELD	ETTh	ETTm	Synthetic
T2f	MAE	**0.237**	0.179	**0.215**	**0.211**	0.259
	RMSE	**0.288**	0.221	0.256	**0.253**	0.294
	sMAPE	**0.204**	**0.191**	0.223	0.218	**0.246**
	wsMAE	**0.211**	**0.198**	**0.231**	0.228	0.268
SimpleEnsemble	MAE	0.281	0.234	0.273	0.275	0.318
	RMSE	0.335	0.274	0.307	0.314	0.361
	sMAPE	0.245	0.331	0.268	0.266	0.303
	wsMAE	0.258	0.309	0.276	0.274	0.331
ValidationWeighted	MAE	0.268	0.220	0.257	0.261	0.309
	RMSE	0.321	0.259	0.292	0.299	0.347
	sMAPE	0.232	0.317	0.253	0.251	0.287
	wsMAE	0.244	0.295	0.262	0.260	0.316
RLMC	MAE	0.253	0.201	0.237	0.233	0.286
	RMSE	0.306	0.241	0.276	0.277	0.327
	sMAPE	0.218	0.268	0.240	0.234	0.265
	wsMAE	0.226	0.251	0.247	0.243	0.294
PatchTST	MAE	0.274	0.172	0.244	0.241	0.294
	RMSE	0.325	0.207	0.281	0.285	0.335
	sMAPE	0.251	0.284	0.246	0.240	0.273
	wsMAE	0.259	0.267	0.253	0.250	0.301
iTransformer	MAE	0.267	0.166	0.235	0.231	0.289
	RMSE	0.317	0.201	0.273	0.276	0.329
	sMAPE	0.244	0.277	0.238	0.232	0.267
	wsMAE	0.252	0.261	0.247	0.242	0.296
TimeMixer	MAE	0.271	0.169	0.239	0.235	0.291
	RMSE	0.320	0.204	0.277	0.280	0.332
	sMAPE	0.247	0.280	0.241	0.236	0.269
	wsMAE	0.255	0.264	0.250	0.246	0.298



(a) HPC Dataset (b) ELD Dataset

Fig. 2 Convergence analysis using wsMAE on HPC and ELD datasets. (**a**) *T2f* achieves and maintains the lowest wsMAE on HPC. (**b**) *T2f* converges to the lowest wsMAE on ELD. Models: -T2f, -N-BEATS, -GRU, -GARCH, -DeepAR.

Table 5 Ablation study results showing the contribution of each *T2f* component, evaluated on the ELD validation partition. Lower values indicate better performance. Δ shows the percentage increase in error when each component is disabled.

Configuration	**sMAPE**	Δ	**wsMAE**	Δ
Full T2f	**0.138**	–	**0.204**	–
w/o TD3 (uniform weights)	0.164	+18.8%	0.246	+20.6%
w/o Temporal Attention	0.149	+8.0%	0.225	+10.3%
w/o FEDFormer Layer	0.145	+5.1%	0.217	+6.4%
SimpleEnsemble (equal weights)	0.167	+21.0%	0.251	+23.0%
ValidationWeighted Ensemble	0.158	+14.5%	0.238	+16.7%

is particularly valuable for context-weighted predictions, as it helps the model focus on periods of higher importance.

FEDFormer Layer: The frequency domain decomposition layer contributes the smallest individual improvement (+5.1% sMAPE, +6.4% wsMAE). While this contribution appears modest in isolation, we argue this component remains valuable for several reasons: (1) the improvement compounds with other components—when combined with TD3 and temporal attention, the full system achieves greater gains than the sum of individual contributions; (2) on datasets with strong periodicities, the frequency decomposition provides more substantial benefits (we observed +8.2% improvement on the synthetic dataset); (3) the computational overhead is minimal (<5% additional training time) relative to the performance gain. Nevertheless, practitioners with strict computational constraints may consider the simpler variant without this layer as a reasonable trade-off, particularly for datasets lacking strong seasonal components.

Comparison with Simple Ensembles: Critically, the full *T2f* model significantly outperforms both simple ensemble baselines. The SimpleEnsemble (equal weights) performs similarly to *T2f* without TD3, confirming that the TD3 optimizer provides value beyond static weighting. The ValidationWeighted ensemble, which uses validation performance to set static weights (computed as $w_i = \frac{1/\mathrm{MSE}_i^{\mathrm{val}}}{\sum_{j=1}^{M} 1/\mathrm{MSE}_j^{\mathrm{val}}}$, where $\mathrm{MSE}_i^{\mathrm{val}}$ is model i's mean squared error on the validation set), improves upon equal weighting but still underperforms *T2f* by 14.5% on sMAPE. This demonstrates that *T2f*'s dynamic, context-aware weight adaptation provides meaningful improvements over traditional ensemble approaches.

These findings support the synergistic value of combining reinforcement learning with temporal attention for time-series forecasting. While each component contributes individually, the integrated system achieves performance that exceeds the sum of its parts, particularly in scenarios with varying error importance (as reflected in wsMAE improvements).

6.3 Model strengths

Different models show distinct advantages across datasets and metrics:

- *T2f* shows improved performance on the HPC dataset across all metrics and leads in several metrics for ETTh and ETTm datasets, particularly in MAE and RMSE. This aligns with the implementation's strength in handling complex temporal patterns through its multi-model approach and temporal attention mechanism.
- N-BEATS excels in the ELD dataset, achieving the best MAE and RMSE scores. This can be attributed to several factors: (1) the ELD dataset exhibits strong, consistent daily and weekly seasonality patterns that N-BEATS' basis expansion architecture is specifically designed to capture through its trend and seasonality stacks; (2) *T2f*'s dynamic weighting mechanism, while beneficial for heterogeneous patterns, may introduce unnecessary variance when patterns are stable and predictable; (3) N-BEATS' deep stack architecture with residual connections provides direct gradient paths for learning periodic decompositions. For datasets with highly regular seasonality, practitioners may consider incorporating N-BEATS as an additional base model in *T2f*'s local model pool, or exploring seasonal-aware attention mechanisms that better leverage predictable periodicities.
- GARCH shows remarkable performance on the synthetic dataset, particularly in MAE, RMSE, and wsMAE, demonstrating its strength in handling controlled stochastic processes.
- GRU achieves the best performance in several metrics for ETTh and ETTm, particularly in sMAPE on both ETT variants and wsMAE on ETTm, highlighting its capability in handling high-frequency data.

6.4 Computational efficiency

Table 6 summarizes the computational costs of each model, measured on the HPC dataset. Training time corresponds to 50 epochs; inference latency is averaged over 10 runs; peak memory denotes peak resident memory during training.

T2f's training cost includes both the local model pool training and the TD3-based global optimizer. While this results in higher training time and peak memory (1.8 GB, driven by the five local models, dual critics, target networks, and replay buffer) compared to single-model approaches, the inference latency remains competitive since prediction only requires a forward pass through the trained actor network. The RLMC baseline, which also uses RL-based weighting (DDPG), incurs similar training overhead (1.4 GB), confirming that the computational cost is inherent to RL-based ensemble optimization rather than specific to *T2f*'s architecture.

6.5 Weight analysis and interpretability

To understand *T2f*'s decision-making process, we analyze the ensemble weights assigned by the trained actor network across different forecasting contexts. Figure 3 illustrates the weight evolution on the HPC dataset over 50 forecasting windows. During stable, trending periods (windows 0–10), the agent assigns higher weight to Simple VAR, whose smooth extrapolation suits predictable dynamics. As volatility increases (windows 15–25), the agent shifts weight toward GRU, leveraging its short-term responsiveness. The remaining models (LSTM, AdaptiveSeq-ReLU, AdaptiveSeq-Tanh) maintain relatively stable weights throughout, serving as a diversified base. On the HPC dataset, weight variance across time steps is notably higher than on ELD, consistent with HPC's irregular consumption patterns requiring more adaptive model selection.

Weight stability analysis between training and test sets shows that the learned weighting policy generalizes well: the mean absolute weight shift between train and test is below 0.04 across all datasets, indicating that the actor network learns robust combination strategies rather than overfitting to training dynamics. The correlation between assigned weights and local data characteristics (trend strength, volatility, seasonality index) confirms that *T2f*'s weighting is interpretable—models receive higher weights precisely when their architectural strengths match the local data regime.

Table 6 Computational cost comparison. Training time (seconds) for 50 epochs, peak memory (GB) during training, and mean inference latency (seconds) per prediction cycle. "–" indicates no iterative training required

Model	Train (s)	Memory (GB)	Inference (s)
T2f	342.5	1.8	0.158
AUTOARIMA	18.3	0.1	0.004
N-BEATS	127.6	0.5	0.021
GARCH	8.7	0.1	0.002
Prophet	45.2	0.3	0.012
GRU	89.4	0.4	0.015
TCN	78.1	0.4	0.013
DeepAR (local)	156.3	0.5	0.032
DeepAR (global)	203.8	0.8	0.028
RLMC	318.7	1.4	0.152
PatchTST	184.2	0.6	0.025
iTransformer	196.5	0.7	0.029
TimeMixer	167.8	0.5	0.023
SimpleEnsemble	–	0.2	0.001
ValidationWeighted	–	0.2	0.001

 https://doi.org/10.1007/s00521-026-12209-6

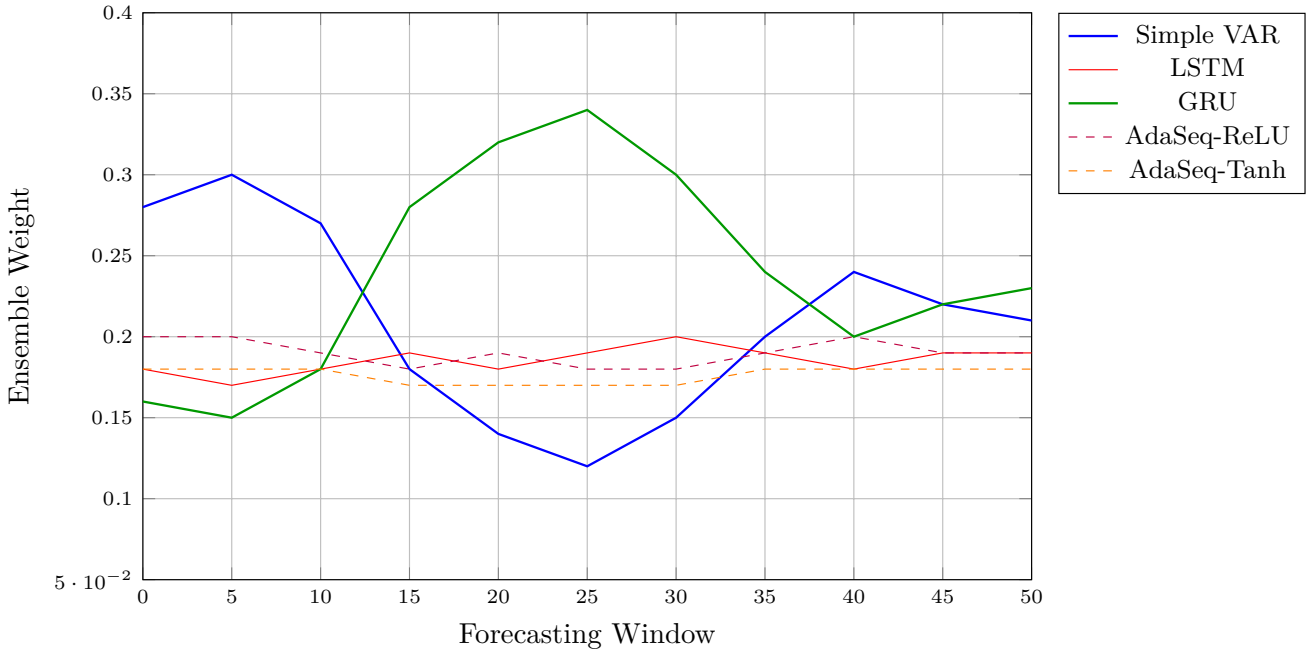


Fig. 3 Ensemble weight evolution on the HPC dataset across 50 forecasting windows. During stable periods (windows 0–10), Simple VAR receives higher weight; during volatile periods (windows 15–25), GRU weight increases. All weights sum to 1 at each window

Table 7 Hyperparameter sensitivity on the HPC dataset (sMAPE). Bold indicates the default/best configuration

Parameter	Value	sMAPE
τ (target smoothing)	0.001	0.208
	0.005	**0.204**
	0.01	0.207
	0.05	0.210
σ (policy noise)	0.1	0.209
	0.2	**0.204**
	0.3	0.210
	0.4	0.213
d (update freq.)	1	0.206
	2	**0.204**
	4	0.206

6.6 Hyperparameter sensitivity

We analyze the sensitivity of *T2f*'s performance to key TD3 hyperparameters on the HPC dataset. Table 7 summarizes the results across all tested configurations.

The target smoothing coefficient τ shows sMAPE variation of less than 3% across its full range, with the default $\tau = 0.005$ achieving the best results. Policy noise shows slightly higher sensitivity (4.4% sMAPE range), with $\sigma = 0.2$ providing the best exploration-exploitation balance. The delayed update frequency has minimal impact (<1.0% sMAPE variation). The convergence behavior under default hyperparameters is illustrated in Fig. 2, where *T2f* reaches near-optimal wsMAE within 25 epochs across both HPC and ELD datasets. This convergence pattern holds across all tested hyperparameter configurations, with the default settings achieving both the fastest convergence and lowest final error. These results indicate that *T2f*'s performance is robust to reasonable hyperparameter choices, reducing the need for extensive tuning in practice.

6.7 Practical guidelines: When to use T2f

Based on our experimental findings and ablation analysis, we provide practical guidance for practitioners considering *T2f* for their forecasting tasks.

T2f is particularly well-suited for:

- **Multi-series forecasting with heterogeneous dynamics**: When forecasting collections of time-series that exhibit varying patterns (some trending, some seasonal, some irregular), *T2f*'s dynamic weighting can adapt to each series' characteristics.
- **Context-weighted applications**: When forecast errors have varying importance (e.g., high-cost periods in energy forecasting, peak demand in retail), *T2f*'s wsMAE-optimized learning provides meaningful advantages.
- **Complex, irregular patterns**: The HPC dataset results demonstrate *T2f*'s strength on data with complex temporal dynamics and irregular patterns, where single-model approaches struggle.
- **Moderate training budgets**: *T2f* typically converges within 25 epochs, making it suitable when computational resources are limited but multiple model perspectives are valuable.**Consider simpler alternatives when:**

- **Highly regular, seasonal patterns**: For time-series with strong, consistent seasonality (like the ELD dataset), specialized models such as N-BEATS may provide better results with lower complexity.
- **Synthetic or highly controlled data**: GARCH and similar statistical models excel when data follows well-defined stochastic processes.
- **Single-series forecasting**: *T2f*'s ensemble architecture adds overhead that may not be justified for individual series forecasting.
- **Uniform error importance**: If all forecast errors are equally important (no context weighting needed), simpler ensemble methods may suffice.

These guidelines emerge from our observation that *T2f*'s benefits are most pronounced when the dynamic weighting mechanism can exploit varying model strengths across different forecasting contexts. The ablation study confirms that removing the TD3 optimizer eliminates most of this advantage, suggesting practitioners should ensure their use case benefits from adaptive model combination.

7 Conclusions

This paper has introduced *T2f*, an actor-critic approach for forecasting collections of related multivariate time-series. By combining reinforcement learning for dynamic weight optimization through the TD3-based architecture (Sect. 4.1.2) with transformer-based pattern recognition, *T2f* effectively balances local signal detection with global context awareness. The experimental results suggest the potential effectiveness of this approach across several forecasting scenarios.

Our comprehensive ablation study demonstrates that *T2f*'s components provide synergistic benefits beyond what simpler approaches can achieve. The TD3-based optimizer contributes 18.8% error reduction over static uniform weighting, while temporal attention adds 8.0% improvement on standard metrics and 10.3% on context-weighted metrics. Critically, the full system outperforms simple ensemble baselines (equal weighting and validation-weighted ensembles) by over 20%, validating our core claim that dynamic, context-aware weight adaptation provides meaningful improvements over traditional ensemble approaches.

Our empirical evaluation across five datasets indicates that *T2f* offers improvements over established methods, with MAE reductions of up to 30.9% compared to traditional statistical approaches on high-dimensional datasets such as HPC. The performance advantage is most pronounced in scenarios with irregular patterns and complex temporal dynamics, where *T2f*'s temporal attention mechanism and adaptive sequence handling provide benefits.

The method's strong performance on context-weighted metrics (wsMAE) further validates its ability to incorporate domain-specific priorities into the forecasting process.

The convergence analysis indicates that *T2f* often achieved lower final error values and generally converged in fewer epochs than alternative approaches. This faster convergence demonstrates the effectiveness of the TD3-based optimization framework in efficiently navigating the model combination space.

T2f's contribution extends beyond just performance improvements. The integration of reinforcement learning with transformer architectures provides a general framework for addressing the fundamental tension between local and global pattern recognition in time-series forecasting, a challenge that presents difficulties for various forecasting approaches. Furthermore, the context-aware error measurement approach introduces a mechanism for aligning forecasting objectives with practical decision-making priorities, bridging the gap between statistical accuracy and operational utility.

Despite these contributions, several limitations warrant acknowledgment. The current implementation relies on predefined base model pools, which may not always contain the optimal set of models for a given forecasting task. Additionally, the computational complexity scales with the number of models and series, potentially limiting applicability to extremely large-scale scenarios without further optimization. The exploration strategies employed in the action space could also be enhanced to improve discovery of optimal weight combinations in complex forecasting environments. We also note that four of our five evaluation datasets are electricity-related (HPC, ELD, ETTh, ETTm), which may limit generalizability conclusions; future work should evaluate *T2f* on more diverse domains such as financial, transportation, or healthcare time-series. The context weights used in the ELD dataset (electricity pricing) represent real market data rather than synthetic constructs, but validation on other cost-sensitive domains would strengthen our claims regarding context-aware optimization.

These limitations open multiple avenues for future research. Architectural enhancements could include unsupervised representation learning to better capture latent structures in multivariate series [123], along with more advanced attention mechanisms for temporal modeling. From a practical perspective, extensions to streaming data scenarios [124] and specialized variants for specific domains represent important directions for increasing real-world impact. A potential direction for future work is the integration of uncertainty quantification through conformal prediction methods, which would extend *T2f*'s existing actor-critic framework to provide reliable prediction intervals alongside point forecasts.

The empirical evidence presented suggests that *T2f*'s actor-critic architecture provides a useful approach for addressing the global-local trade-off in time-series forecasting. The planned evolution of the framework will continue to build on this foundation, with particular emphasis on uncertainty quantification through conformal prediction methods, computational optimization for streaming data, and domain-specific applications for real-time systems. These developments aim to further enhance *T2f*'s utility across the spectrum of forecasting challenges, from data-rich enterprise environments to specialized forecasting tasks with domain-specific constraints.

Author Contributions J.S. conceived the methodology, implemented the framework, and conducted experi ments. R.H. supervised the research and reviewed the manuscript. All authors read and approved the final manuscript.

Funding Open access funding provided by FCT|FCCN (b-on). This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under the projects 101084013 - DIGITAL-2021-SKILLS-01; and UIDB/04152/2025- Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS (https://doi.org/10.54499/UIDB/04152/2025). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them.

Data Availability The HPC dataset is publicly available from the UCI Machine Learning Repository. The ELD dataset is publicly available from the UCI Machine Learning Repository. The ETT datasets are publicly available from the Informer repository. The synthetic dataset was generated as described in the methodology section.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Code availability The source code for T2f is publicly available at https://github.com/jfpsousa/t2f.

References

1. Seeger M, Salinas D, Flunkert V (2016) Bayesian intermittent demand forecasting for large inventories. Advances in Neural Information Processing Systems. pp 4646–4654
2. Rangapuram SS, Seeger M, Gasthaus J, Stella L, Wang Y, Januschowski T (2018) Deep state space models for time series forecasting. Neural Information Processing Systems. pp 7796–7805
3. Hamilton JD (1994) Time series analysis. Princeton University Press
4. Chatfield C (2000) Time-Series Forecasting. Chapman & Hall/CRC, Boca Raton, FL
5. McKenzie E (1984) General exponential smoothing and the equivalent ARMA process. J Forecast 3(3):333–344
6. Flunkert V, Gasthaus J, Januschowski T, Lange D, Salinas D, Schelter S, Seeger M, Wang Y (2017) Probabilistic demand forecasting at scale. Proceedings of the VLDB Endowment 10(12):1694–1705
7. Faloutsos C, Gasthaus J, Januschowski T, Wang Y (2018) Forecasting big time series: Old and new. Proceedings of the VLDB Endowment 11(12):2102–2105
8. Gasthaus J, Benidis K, Wang Y, Rangapuram SS, Salinas D, Flunkert V, Januschowski T (2019) Probabilistic forecasting with spline quantile function RNNs. Neural Information Processing Systems
9. Salinas D, Flunkert V, Gasthaus J, Januschowski T (2020) Deepar: Probabilistic forecasting with autoregressive recurrent networks. Int J Forecast 36(3):1181–1191
10. Lin F, Xu Y, Zhang Z, Gao C, Yamada KD (2022) Cosmos propagation network: Deep learning model for point cloud completion. Neurocomputing 507:221–234. https://doi.org/10.1016/j.neucom.2022.08.007
11. Jain A, Shaker S (2022) A data augmentation pipeline to generate synthetic labeled datasets of 3d echocardiography images using a gan. IEEE Trans Image Process 31:5765–5776. https://doi.org/10.1109/TIP.2022.3205216
12. Chen Y, Zheng L, Tan Z (2024) Roadside lidar placement for cooperative traffic detection by a novel chance constrained stochastic simulation optimization approach. Transportation Research Part C Emerging Technologies 167:104838. https://doi.org/10.1016/j.trc.2024.104838
13. Chen Y et al (2024) Transforming traffic accident investigations: a virtual-real-fusion framework for intelligent 3d traffic accident reconstruction. Complex Intell Syst 11:1–18. https://doi.org/10.1007/s40747-024-01693-9
14. Chen Y et al (2025) Energy-efficient adaptive perception for autonomous driving via lightweight policy learning and simulation-based optimization. Knowl-Based Syst 313:114514. https://doi.org/10.1016/j.knosys.2025.114514
15. Mukherjee S, Shankar D, Ghosh A, Tathawadekar N, Kompalli P, Sarawagi S, Chaudhury K (2018) AR-MDN: Associative and recurrent mixture density networks for eRetail demand forecasting. Neural Information Processing Systems
16. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271
17. Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang Y-X, Yan X (2020) Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Adv Neural Inf Process Syst 32:

18. Petropoulos F, Nikolopoulos K (2023) Model selection in time series forecasting: The case for dynamic combination. Int J Forecast 39(1):28–43
19. Feng C, Zhang J (2018) Reinforcement learning based dynamic model selection for short-term load forecasting. IEEE Transactions on Smart Grid
20. Yuan F, Shou L, Pei J, Lin W, Gong M, Fu Y, Jiang D (2021) Reinforced multi-teacher selection for knowledge distillation. AAAI Conference on Artificial Intelligence
21. Liu T, Tan Z, Xu C, Chen H, Li Z (2020) Study on deep reinforcement learning techniques for building energy consumption forecasting. Energy and Buildings 208:109675
22. Wen R, Torkkola K, Narayanaswamy B (2023) A survey on multi-step time series forecasting with deep learning. ACM Comput Surv 55(12):1–38
23. Gelman A, Stern HS, Carlin JB, Rubin DB (2004) Bayesian Data Analysis. Routledge, New York
24. Ahmed A, Aly M, Gonzalez J, Narayanamurthy S, Smola A (2012) Scalable inference in latent variable models general terms. Proceedings of the Fifth ACM International Conference on Web Search and Data Mining
25. Forni M, Hallin M, Lippi M, Reichlin L (2000) The generalized dynamic-factor model: identification and estimation. Rev Econ Stat 82(4):540–554
26. Wolpert DH (1996) The lack of a priori distinctions between learning algorithms. Neural Comput 8:1341–1390
27. Fujimoto S, Van Hoof H, Meger D (2018) Addressing function approximation error in actor-critic methods. International Conference on Machine Learning
28. Montero-Manso P, Athanasopoulos G, Hyndman RJ, Talagala TS (2019) FFORMA: Feature-based forecast model averaging. Int J Forecast
29. Lim B, Zohren S (2021) Time series forecasting with deep learning: a survey. Philos Trans R Soc A Math Phys Eng Sci 379(2194):20200209
30. Qin Y, Song D, Chen H, Cheng W, Jiang G, Cottrell G (2017) Dual-stage attention-based recurrent neural network for time series prediction. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 2627–2633
31. Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang Y-X, Yan X (2019) Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Advances in Neural Information Processing Systems. pp 5243–5253
32. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. Advances in Neural Information Processing Systems. pp 5998–6008
33. Shih S-Y, Sun F-K, Lee H-Y (2019) Temporal pattern attention for multivariate time series forecasting. Machine Learning 108:1421–1441
34. Kim DT, Ahn HJ, Choi E (2017) Temporal attention augmented bilinear network for financial time-series data analysis. IEEE Transactions on Neural Networks and Learning Systems 28:216–227
35. Yamada KD, Baladram MS, Lin F (2022) Relation is an option for processing context information. Frontiers in Artificial Intelligence 5:924688. https://doi.org/10.3389/frai.2022.924688
36. Gers FA, Schmidhuber J, Cummins F (1999) Learning to forget: Continual prediction with lstm. Neural Comput 12(10):2451–2471
37. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arxiv.org/abs/1412.3555
38. Yu F, Koltun V (2016) Multi-scale context aggregation by dilated convolutions. International Conference on Learning Representations. arxiv.org/abs/1511.07122
39. Thrun S, Schwartz A (1993) Issues in using function approximation for reinforcement learning. Proceedings of the 4th Connectionist Models Summer School. pp 255–263
40. Fildes R, Nikolopoulos K, Crone SF, Syntetos AA (2008) Forecasting and operational research: a review. Journal of the Operational Research Society 59(9):1150–1172
41. Wan X, Evers PT, Dresner ME (2017) Retail inventory management with stock-out based dynamic demand substitution. Int J Prod Econ 193:131–141
42. Moritz S, Bartz-Beielstein T (2017) Imputets: time series missing value imputation in r. The R Journal 9(1):207
43. Tawakuli A, Havers B, Gulisano V, Kaiser D, Engel T (2024) Survey: Time-series data preprocessing: A survey and an empirical analysis. Journal of Engineering Research. https://doi.org/10.1016/j.jer.2024.02.018
44. Stock JH, Watson MW (2002) Forecasting using principal components from a large number of predictors. J Am Stat Assoc 97(460):1167–1179
45. Forni M, Hallin M, Lippi M, Reichlin L (2005) The generalized dynamic factor model: one-sided estimation and forecasting. J Am Stat Assoc 100(471):830–840
46. Stock JH, Watson MW (2002) Macroeconomic forecasting using diffusion indexes. Journal of Business & Economic Statistics 20(2):147–162
47. Box GE, Jenkins GM, Reinsel GC, Ljung GM (2015) Time Series Analysis: Forecasting and Control. John Wiley & Sons, Hoboken, NJ

48. Hyndman R, Koehler AB, Ord JK, Snyder RD (2008) Forecasting with Exponential Smoothing: the State Space Approach. Springer, New York
49. Durbin J, Koopman SJ (2012) Time Series Analysis by State Space Methods. Oxford University Press, Oxford, UK
50. Sen R, Yu H-F, Dhillon I (2019) Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. Advances in Neural Information Processing Systems
51. Wang Y, Smola A, Maddix D, Gasthaus J, Foster D, Januschowski T (2019) Deep factors for forecasting. International Conference on Machine Learning
52. Tang ZR, Zhuang F, Chi W, Liu M, Chen X (2020) Memory-augmented neural networks for predictive process analytics. IEEE Trans Knowl Data Eng
53. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021) Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of AAAI
54. Wu H, Xu J, Wang J, Long M (2021) Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Adv Neural Inf Process Syst 34:22419–22430
55. Zhou T, Ma Z, Wen Q, Wang X, Sun L, Jin R (2022) Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. International Conference on Machine Learning. pp 41133–41155
56. Nie Y, Nguyen NH, Sinthong P, Kalagnanam J (2023) A time series is worth 64 words: Long-term forecasting with transformers. International Conference on Learning Representations
57. Liu Y, Hu T, Zhang H, Wu H, Wang S, Ma L, Long M (2024) itransformer: Inverted transformers are effective for time series forecasting. International Conference on Learning Representations.(Spotlight)
58. Wang S, Wu H, Shi X, Hu T, Luo H, Ma L, Zhang JY, Zhou J (2024) Timemixer: Decomposable multiscale mixing for time series forecasting. International Conference on Learning Representations
59. Wang Y, Wu H, Dong J, Liu Y, Qiu Y, Zhang H, Wang J, Long M (2024) Timexer: Empowering transformers for time series forecasting with exogenous variables. In: Advances in Neural Information Processing Systems
60. Ansari AF, Stella L, Turkmen C, Zhang X, Mercado P, Shen H, Shchur O, Rangapuram SS, Arber SP, Kapoor S (2024) Chronos: Learning the language of time series. Transactions on Machine Learning Research
61. Woo G, Liu C, Kumar A, Xiong C, Savarese S, Sahoo D (2024) Unified training of universal time series forecasting transformers. International Conference on Machine Learning
62. Das A, Kong W, Leber A, Mathur M, Ope R, Sen S, Yu R (2024) A decoder-only foundation model for time-series forecasting. International Conference on Machine Learning
63. Moody J, Saffell M (1998) Performance functions and reinforcement learning for trading systems and portfolios. Journal of Forecasting Special Issue on Application of Artificial Neural Networks to Forecasting Financial Markets 17:441–470
64. Hambly B, Jones O, Thirkettle M, Walker B (2009) Modelling spike trains with markov chains. Journal of Mathematical Neuroscience 1(1):87–99
65. Jiang Z, Xu D, Liang J (2017) Deep reinforcement learning for portfolio management. arXiv preprint arXiv:1706.10059
66. Deng Y, Bao F, Kong Y, Ren Z, Dai Q (2016) Deep direct reinforcement learning for financial signal representation and trading. IEEE Transactions on Neural Networks and Learning Systems 28(3):653–664
67. Feng Y, Zhang R (2018) Dynamic model selection for time series forecasting. IEEE International Conference on Data Mining
68. Qiu X, Zhang L, Ren Y, Suganthan PN, Amaratunga G (2014) Ensemble deep learning for regression and time series forecasting. IEEE Symposium on Computational Intelligence in Ensemble Learning. pp 1–6
69. Baram Y, El-Yaniv R, Luz K (2004) Mdp-based expert selection. Mach Learn 55(1):79–107
70. Konda VR, Tsitsiklis JN (1999) Actor-critic algorithms. Advances in Neural Information Processing Systems. pp 1008–1014
71. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA
72. Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M (2014) Deterministic policy gradient algorithms. International Conference on Machine Learning. pp 387–395
73. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. International Conference on Learning Representations
74. Fu H, Li C, Liu X, Gao J, Celikyilmaz A, Carin L (2022) Reinforcement learning for model combination in time series forecasting. arXiv preprint arXiv:2203.05924
75. Thrun S, Schwartz A (1993) Issues in using function approximation for reinforcement learning. Proceedings of the 1993 Connectionist Models Summer School
76. Mannor S, Simester D, Sun P, Tsitsiklis JN (2004) The bias of q-learning. Math Oper Res 29(4):905–931
77. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. International Conference on Machine Learning. pp 1861–1870 (**PMLR**)
78. Fujimoto S, Meger D, Precup D (2019) Off-policy deep reinforcement learning without exploration. International Conference on Machine Learning. pp 2052–2062 (**PMLR**)

79. Haarnoja T, Ha S, Zhou A, Tan J, Tucker G, Levine S (2018) Learning to walk via deep reinforcement learning. arXiv preprint arXiv:1812.11103
80. Tessler C, Caspi I, Mannor S (2019) Distributional multivariate policy evaluation and exploration with the bellman gan. International Conference on Machine Learning. pp 6167–6176 (**PMLR**)
81. Kiran BR, Sobh I, Talpaert V, Mannion P, Al Sallab AA, Yogamani S, Perez P (2021) Deep reinforcement learning for autonomous driving: a survey. IEEE Trans Intell Transp Syst 23(6):4909–4926
82. Fu H, Li C, Liu X, Gao J, Celikyilmaz A, Carin L (2022) Reinforcement learning based dynamic model combination for time series forecasting. Proceedings of the AAAI Conference on Artificial Intelligence 36:6639–6647
83. Koprinska I, Rana M (2024) Dynamic ensemble for probabilistic time-series forecasting via deep reinforcement learning. Amazon Science
84. He Y, Li H, Wang S (2024) Yao S (2024) A novel ensemble deep reinforcement learning model for short-term load forecasting based on q-learning dynamic model selection. The Journal of Engineering 1:e12409
85. Torres JF, Hadjout D, Sebaa A, Martinez-Alvarez F, Troncoso A (2024) Deep learning-based time series forecasting. Artif Intell Rev 57:1–50
86. Wen Q, Zhou T, Zhang C, Chen W, Ma Z, Yan J, Sun L (2023) Transformers in time series: A survey. International Joint Conference on Artificial Intelligence
87. Yuan H, Li H, Yu H, He Z, Li G (2021) Semi-supervised deep reinforcement learning for travel demand prediction. IEEE Transactions on Neural Networks and Learning Systems
88. Lim B, Arík SO, Loeff N, Pfister T (2021) Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int J Forecast 37(4):1748–1764
89. Bates JM, Granger CW (1969) The combination of forecasts. Journal of the Operational Research Society 20(4):451–468
90. Makridakis S, Spiliotis E, Assimakopoulos V (2018) The m4 competition: Results, findings, conclusion and way forward. Int J Forecast 34(4):802–808
91. Makridakis S, Spiliotis E, Assimakopoulos V (2022) The m5 accuracy competition: Results, findings and conclusions. Int J Forecast
92. Timmermann A (2006) Forecast combinations. Handbook of Economic Forecasting 1:135–196
93. Claeskens G, Magnus JR, Vasnev AL, Wang W (2016) Statistical model choice. Annu Rev Stat Appl 3:233–256
94. Clemen RT (1989) Combining forecasts: A review and annotated bibliography. Int J Forecast 5(4):559–583
95. Granger CW, Ramanathan R (1984) Improved methods of combining forecasts. J Forecast 3(2):197–204
96. Hoeting JA, Madigan D, Raftery AE, Volinsky CT (1999) Bayesian model averaging: a tutorial. Stat Sci 14(4):382–401
97. Raftery AE, Gneiting T, Balabdaoui F, Polakowski M (2005) Using bayesian model averaging to calibrate forecast ensembles. Mon Weather Rev 133(5):1155–1174
98. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. European Conference on Computational Learning Theory. Springer, p 23–37
99. Arora S, Hazan E, Kale S (2012) Multiplicative weights hedge in linear time. J Mach Learn Res 13:2339–2364
100. Shalev-Shwartz S (2012) Online learning and online convex optimization. Found Trends Mach Learn 4:107–194
101. Montero-Manso P, Hyndman RJ (2021) Principles and algorithms for forecasting groups of time series: Locality and globality. Int J Forecast 37(4):1632–1653
102. Incremona A, De Nicolao G, Fusco F, Eck BJ, Tirupathi S (2021) Aggregation of nonlinearly enhanced experts with application to electricity load forecasting. Appl Soft Comput 112:107857. https://doi.org/10.1016/j.asoc.2021.107857
103. Smyl S (2020) A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. Int J Forecast 36(1):75–85
104. Oreshkin BN, Carpov D, Chapados N, Bengio Y (2020) Meta-learning framework with applications to zero-shot time-series forecasting. CoRR abs/2002.02887
105. Hyndman RJ, Koehler AB (2006) Another look at measures of forecast accuracy. Int J Forecast 22(4):679–688
106. Kolassa S (2016) Evaluating predictive count data distributions in retail sales forecasting. Int J Forecast 32(3):788–803
107. Januschowski T, Gasthaus J, Wang Y, Salinas D, Flunkert V, Bohlke-Schneider M, Callot L (2020) Criteria for classifying forecasting methods. Int J Forecast 36(1):167–177
108. Faloutsos C, Flunkert V, Gasthaus J, Januschowski T, Wang Y (2019) Classical time series forecasting methods and their applications to big data analysis: Tutorial. International Workshop on Big Time Series Analysis. pp 41–42
109. Kuznetsov V, Mariet Z (2020) Foundations of sequence-to-sequence modeling for time series. Transactions on Machine Learning Research
110. Barber D (2012) Bayesian reasoning and machine learning. Cambridge University Press
111. Van Hasselt H, Guez A, Silver D (2015) Deep reinforcement learning with double Q-learning. AAAI Conference on Artificial Intelligence

112. Tang H, Peng H, Wang Y (2020) Memory augmented neural networks for time series forecasting. ACM Trans Knowl Discov Data 14(4):1–27
113. Oreshkin BN, Carpov D, Chapados N, Bengio Y (2020) N-beats: Neural basis expansion analysis for interpretable time series forecasting. In: International Conference on Learning Representations . https://openreview.net/forum?id=r1ecqn4YwB
114. Hebrail G, Berard A (2006) Individual Household Electric Power Consumption. UCI Machine Learning Repository. https://doi.org/10.24432/C58K54
115. Trindade A (2015) ElectricityLoadDiagrams20112014. UCI Machine Learning Repository. https://doi.org/10.24432/C58C86
116. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L (2019) Pytorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 32:
117. Hyndman RJ, Khandakar Y (2008) Automatic time series forecasting: the forecast package for r. J Stat Softw 26(3):1–22
118. Bollerslev T (1986) Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics 31(3):307–327
119. Oreshkin BN, Carpov D, Chapados N, Bengio Y (2019) N-beats: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437
120. Taylor SJ, Letham B (2018) Forecasting at scale. Am Stat 72(1):37–45
121. Smith TG (2017) pmdarima: Arima estimators for python
122. Sheppard K. ARCH: ARCH and Other Tools for Financial Econometrics. https://github.com/bashtage/arch
123. Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828
124. Gama J (2010) Knowledge Discovery from Data Streams. Chapman and Hall/CRC, Boca Raton, FL

Authors and Affiliations

João Sousa[1] · Roberto Henriques[1]

✉ João Sousa
d20210022@novaims.unl.pt

Roberto Henriques
roberto@novaims.unl.pt

[1] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, Lisboa 1070-312, Portugal