Dynamic Ensemble for Probabilistic Time-series Forecasting via Deep Reinforcement Learning

Yuhao Ding*
IEOR, UC Berkeley

Youngsuk Park†
AWS AI Labs

Karthick Gopalswamy
AWS AI Labs

Hilaf Hasson
AWS AI Labs

Yuyang Wang
AWS AI Labs

Jun Huan
AWS AI Labs

ABSTRACT

Ensembles have long been a cornerstone in improving performance by integrating black-box base learners. The primary approach, "stacked generalization" is inherently static, lacking adaptability to changes in input data post-training. This limitation is more pronounced in time series forecasting, where these methods struggle to manage temporal correlations or non-stationarity. Given base learners may perform better under various data conditions and timeframes, combining them naively could hurt the desirable properties of probabilistic forecasters for subsequent tasks.

To address these challenges, we introduce a novel "dynamic ensemble policy." This policy employs a deep Reinforcement Learning framework to ensemble multi-horizon probabilistic forecasters. By incorporating intermediate ensembled samples as feedback to each base learner, we mitigate temporal error accumulation that plagues base auto-regressive learners. The resulting ensemble forecaster is dynamic, adapting to forecast start times and unseen data, while preserving the advantageous properties of auto-regressive models in terms of samplings and quantile estimates. To train this policy, we design a Markov Decision Process (MDP) by defining appropriate state, action, and transition dynamics. An environment, *TS-GYM*, simulates this MDP, enabled by the interactions between the agent that ensembles forecasters, offline datasets, and base forecasters. The effectiveness of the proposed framework is demonstrated on multiple synthetic and real-world experiments against several ensemble baselines.

KEYWORDS

Machine learning, Time-series, Ensemble, Forecasting, Robustness, Reinforcement Learning, Probabilistic Models

ACM Reference Format:
Yuhao Ding*, Youngsuk Park†, Karthick Gopalswamy, Hilaf Hasson, Yuyang Wang, and Jun Huan. 2023. Dynamic Ensemble for Probabilistic Time-series Forecasting via Deep Reinforcement Learning. In *Proceedings of 9TH SIGKDD INTERNATIONAL WORKSHOP ON MINING AND LEARNING FROM TIME SERIES (MileTS '23).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

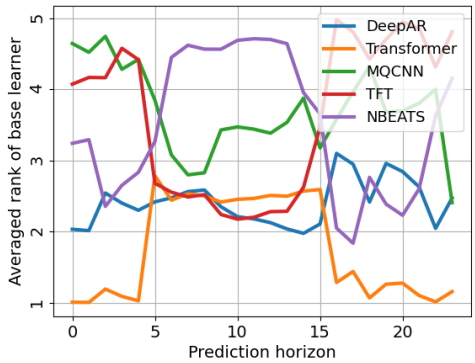
1 INTRODUCTION

Time series data occurs naturally in countless domains including supply chain optimization [6, 25], medical analysis [4, 14], financial analysis [3, 26], sensor network monitoring [7, 16], cloud computing [17, 19], optimal control [5, 18] and social activity mining [11–13]. Among the applications of ML-based time series analysis, forecasting is arguably one of the most sought-after, due to its importance in industrial, social, and scientific applications. For example, forecasting plays a key role in automating and optimizing operational processes in most businesses and enables data driven decision making. Forecasts of product supply and demand are used for optimal inventory management, staff scheduling and topology planning, and are more generally a crucial technology for most aspects of supply chain optimization. In order to make optimal decisions, predictive uncertainties need to be taken into account, making probabilistic forecast a desirable property of time series models [2].

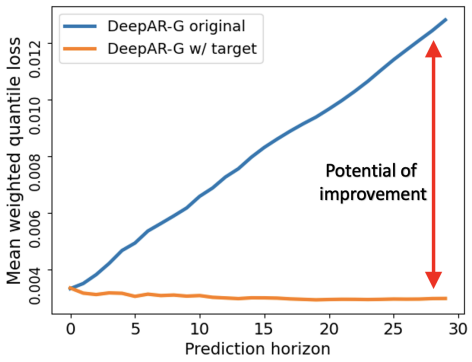
In practice, one often encounters complex time series, making it difficult to find a single best model that excels at short-term, mid-term, and long-term forecasting scenarios. In such cases, different forecasting models usually perform well on different data regimes at different time steps. As a motivating example, Figure 1a shows the relative ranking of the performances of 5 popular forecasting models on the *Solar* dataset; transformer excels at shorter and longer-term forecasts while DeepAR and TFT shine in the mid-term scenario. It is thus desirable to have an ensembling strategy that can have different weights at each time step. Therefore, the traditional ensembling strategy in time series forecast, which assumes that ensemble weights do not vary along the forecasting horizon is not sufficient to capture the non-stationary patterns of base learners' performance profile. Furthermore, popular auto-regression based models are known to have increasing prediction errors as the prediction horizon stretches further, and the performance degrades dramatically when the prediction horizon is sufficiently large [23]. As shown in the blue curve of Figure 1b, the prediction error increases for "DeepAR-G original" ("G" means using the Gaussian distribution as the output distribution and "original" means using the original auto-regressive feeding in DeepAR) over the prediction horizon on *exchange rate* dataset. On the other hand, if we can provide base learners such as DeepAR with ground-truth (future) target values as the auto-regressive input, denoted as "DeepAR-G target", the prediction error can be significantly decreased for the long horizon predictions, depicted as the orange curve in Figure 1b. The huge difference in the prediction error between these two cases show the potential to improve the auto-regression based models if we can provide more accurate intermediate estimation via ensembling along the way. However, none of the traditional ensemble methods utilize the ensemble predictions as the feedback to boost

MileTS '23, Aug 7th, 2023, Long Beach, CA
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

* Work done during the author's internship at AWS AI Labs. †Correspondence to: Youngsuk Park <pyoungsu@amazon.com>.



(a) The ranks of 5 base learners along the prediction horizon on Solar dataset. The ranks are based on the mean weighted quantile loss over the quantiles $[0.1, 0.5, 0.9]$ and averaged over all items in each dataset.



(b) The gap between the "DeepAR original" (fed with its own predicted sample) and "DeepAR w/ target" (fed with the ground-truth target) shows the potential improvement under more accurate auto-regressive samples.

Figure 1: Two motivations on the need of dynamic ensembles, beyond static ensembles.

the performance of the auto-regression based models. *Motivated by the above examples, the natural question arises whether we can develop a general dynamic ensembling approach that overcomes all the major limitations of the traditional static ensemble methods and further improve the prediction accuracy for the probabilistic time-series forecasting?*

To address the above mentioned challenges, in this work, we develop a general dynamic ensemble framework for probabilistic multi-horizon time series forecasting. Our contributions can be summarized as follows:

- This work is the first one that proposes a dynamic ensemble policy suitable for probabilistic time series forecasting with the properties of sequential weighting, being adaptive, and quantile ensemble.
- We formulate this as a Markov Decision Process (MDP) with a careful design of the rewards, transition dynamics, and ensemble action policy. In particular, the state evolution in our formulation depends on the ensemble strategy through our novel transition dynamics design.
- To solve this MDP problem, we design a time series gym (TS-GYM) environment which implements the interaction between the time series off-line dataset, base learners and ensemble agent. Through this interaction, actor-critic based deep RL method with our "random extreme point" exploration strategy can learn optimal ensemble policy.
- The extensive experiments show the advantages of our ensemble dynamic framework. In particular, we demonstrate that our general dynamic ensemble framework can (1) learn the optimal time-varying ensemble weights along the multi-horizon prediction, (2) be adaptive to any forecast start time, (3) boost the performance of the auto-regressive base learners, and (4) result in better performance than other potential variants on real-world datasets.

2 PRELIMINARIES

Notation. For a random variable $Z \in \mathbb{R}$, we denote $F_Z(z)$ as its cumulative distribution function (CDF). Then, the τ-quantile of Z is defined as

$$q_\tau(Z) := F_Z^{-1}(\tau) = \inf\{z \in \mathbb{R} : \tau \le F_Z(z)\},$$

When empirical CDF $\hat{F}_Z$ from samples, but not exact F_Z, is available, we use $\hat{F}_Z$ to approximate quantile.

2.1 Probabilistic time-series forecasting

Suppose we have a panel of n time series, where the i-th time series consists of observations $z_{i,t} \in \mathbb{R}$ with (optional) input covariates $\xi_{i,t} \in \mathbb{R}^d$, as t varies over time at fixed discrete intervals. For the i-th time series (often called i-th item), we wish to make predictions for the next H timesteps, namely of $z_{i,T+1:T+H}$ from the forecast start time $T+1$, given the history of that item's observations $z_{i,1:T}$ and (optional) the associated historical and future covariates $\xi_{i,1:T+H}$. In this paper we will focus on global forecasters, namely a single univariate model trained on all of the items together, and accepting only a single item at inference. To avoid cluttering the notation, we will drop the item index i and covariates $\xi_{i,t}$ unless explicitly stated. We now formally define an auto-regressive forecasting model as a set of random variable valued functions $\{f_h\}_{h=1}^H$ such that, for $h = 1, ..., H$

$$Z_{T+h} = f_h(z_{1:T}, \hat{z}_{T+1:T+h-1}), \tag{1}$$

where $\hat{z}_{T+1:T+h-1}$ is a sequence of previous prediction samples to be fed for the next prediction. In case of sequence-to-sequence (Seq2Seq), it does not take $\hat{z}_{T+1:T+h-1}$ prediction samples and output Z_{T+h} for $h = 1, \ldots, H$ at once. Refer to [1] for the detailed modeling. For notational simplicity, we express models in the form of equation 1.

Then, the associated τ-quantile predictions can be followed as

$$\hat{q}^\tau_{T+h} := q_\tau\left(Z_{T+h}\right)$$

2.2 Forecasting ensemble

For the m-th base learner, we denote $\hat{q}^{\tau_k,m}_{T+h}$ as the τ_k-quantile prediction at time step $T+h$ at a quantile level where $\tau_k \in \{\tau_k\}_{k=1}^K$. Then, $\left\{\hat{z}^{\tau_k,m}_{T+h}\right\}_{k=1,m=1}^{K,M}$ is denoted as a pool of quantile predictions at time step $T+h$ over M base learners and K quantile levels. A general ensemble predictions can be formally expressed as a (linear) weighted combination of predictions of the individual base models, at each prediction step $h = 1, \ldots, H$ and $\tau = \tau_1, \ldots, \tau_K$,

$$\hat{q}^{\tau,\text{es}}_{T+h} = \sum_{m=1}^{M} w^m_h \hat{q}^{\tau,m}_{T+h}, \tag{2}$$

where $w^m_h \geq 0$ with $\sum_{m=1}^M w^m_h = 1$ are the ensemble weights and $\hat{q}^{\tau,m}_{T+h}$ may depend on previous ensemble weights $\{w^m_{1:h-1}\}_{m=1}^M$. For example if $w^m_h = 1/M$ or indicator on median index m, it is mean ensemble or median ensemble respectively.

2.3 Reinforcement learning

Reinforcement learning (RL) is usually formulated as a Markov Decision Process (MDP), which can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, H)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor and $H > 0$ is the horizon length of each episode. At each state $s \in \mathcal{S}$, the RL agent takes an action $a \in \mathcal{A}$, transits to the next state $s' \in \mathcal{S}$ under the dynamics $\mathcal{P}$ and receives a reward $r(s, a)$. The goal of an MDP is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the total obtained rewards $\max_\pi J(\pi) = \mathbb{E}_\tau \left[\sum_{h=0}^{H-1} \gamma^h r(s_h, a_h) \mid \pi\right]$, where the expectation is over the trajectory $\tau = \{(s_0, a_0, r(s_0, a_0)). \ldots, (s_{H-1}, a_{H-1}, r(s_{H-1}, a_{H-1}))\}$ where $a_h = \pi(s_h)$.

3 DYNAMIC ENSEMBLE FRAMEWORK

In this section, we mainly focus on how to select a sequence of ensemble weights $(w_1, w_2, \ldots, w_H)$ with $w_h \in \mathbb{R}^M$ over M base learners by learning a ensemble policy π. Especially in the presence of auto-regressive base learners, the intermediate (fixed) ensembled sample at the step h can help better forecasting of auto-regressive base learners at next step $h+1$, which subsequently can determine the choice of ensembled sample at the next $h+1$. (see Section 3.1.1 for more details). With this intuition, we will take a reinforcement learning approach to learn an optimal policy function π that provides the optimal ensemble weights sequentially.

In Section 3.1, we give a high-level overview of the MDP formulation for the multi-horizon probabilistic time series forecasting problems. In particular, the classes of ensembled sampling strategies and predictions which determine the state transformation and state transition are discussed in Section 3.1.1 and the careful design of reward computation is explained in Section 3.1.2. Based on the formulated MDP, we then design our simulated environment, TS-GYM in Section 3.2 which provides the interaction among the time series datasets, base learners and the dynamic ensemble agent. Finally, we describe how to employ deep reinforcement learning with our "random extreme point" exploration strategy to learn the optimal ensemble policy in Section 3.3.

3.1 MDP formulation

We describe the high-level formulation of the MDP for our dynamic time-series ensemble framework. Once each episode starts with $h = 1$, the environment provide a set of time series data: 1) T historical data $z_{1:T}$, 2) corresponding future (backtest) output z_{T+1}, 3) corresponding quantile predictions $\{\hat{q}^{\tau,m}_{T+1}\}$ from all M base models for the next step $T+1$. (We defer the details on how to provide such data the environment to Section 3.2). The agent will then decide the ensemble weights $\{w^m_{T+1}\}_{m=1}^M$ to compute the next ensembled predictions. With this new ensembled sample fed into base learners at step $h = 2$, the environment provides next time series output z_{T+2} and associated predictions $\{\hat{q}^{\tau,m}_{T+2}\}$ and go on. See Figure 2a for a high level schema. More formally, we define MDP as follows:

- the fixed-size state $s_h = \left\{z_{1:T}, \{\hat{q}^{\tau_k,m}_{T+h}\}_{k=1,m=1}^{K,M}, h\right\}$,
- the action $a_h = \{w^m_h\}_{m=1}^M = \pi(s_h)$, M-ensemble weights w^m_h from a policy function π ,
- the state transition $\mathcal{P}(s_{h+1} \mid s_h, a_h)$ governed by ensemble dynamics in Section 3.1.1,
- the reward $R(s_h, a_h; z_{T+h})$ [1] which evaluates ensemble prediction against ground-truth z_{T+h} in Section 3.1.2 .

3.1.1 Ensemble dynamics $\mathcal{P}$ and ensembled quantiles. Defining state transition $\mathcal{P}$, which we call ensemble dynamics, narrows down how to construct quantile predictions over M base learners $\{\hat{q}^{\tau_k,m}_{T+h}\}_{k=1,m=1}^{K,M} \in s_h$ as $z_{1:T} \in s_h$ is already given. The idea is based on auto-regressive models where you recursively feed a new ensembled sample to each base learner for the next prediction. In this dynamic, we generate an (intermediate) ensembled sample p_{T+h}, which is fed into each autoregressive base leaner in a recursive manner. This ends up forming a sample path $(\hat{z}^m_{T+1}, \ldots, \hat{z}^m_{T+H})$, through multiple of which we can compute the final ensembled (empirical) quantile prediction $\hat{q}^{\tau,\text{es}}_{T+h}$. The ensemble dynamics appear as the red arrow line in Figure 2a and are described as follows.

Generate ensembled sample path. To begin with, we generate a sample path $(\hat{z}^m_{T+1}, \ldots, \hat{z}^m_{T+H})$ for each base learner as follows: First, for each step h, we sample p_{T+h} from mixture of base learners' distributions $F(Z^m_{T+h})$ proportional to ensemble weights w^m_h, i.e.,

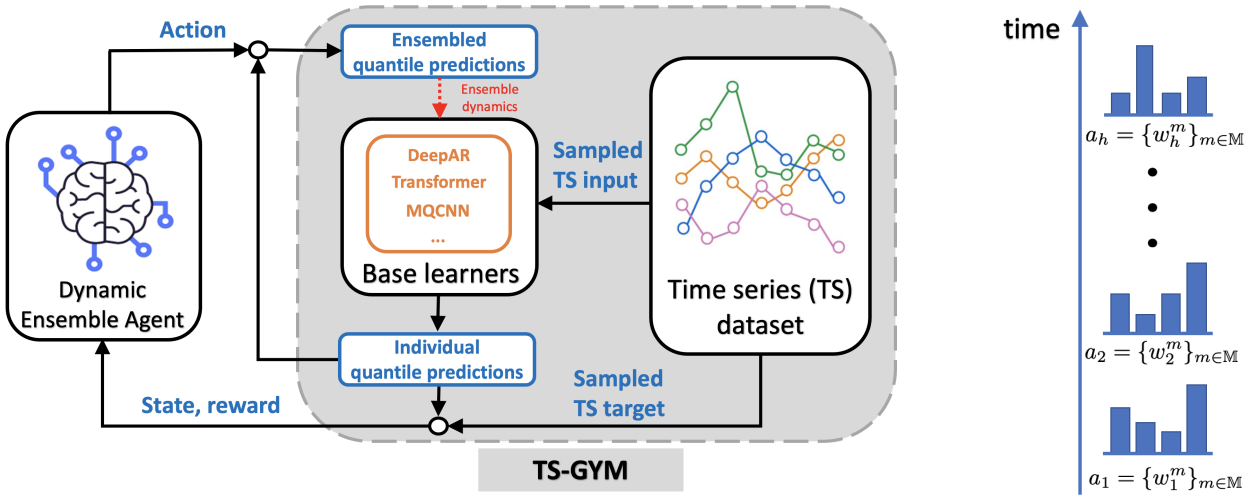
$$p_{T+h} \sim \sum_{m=1}^{M} w^m_h F_{Z^m_{T+h}}. \tag{3}$$

Second, we feed p_{T+h} to each auto-regressive base learners, i.e.,

$$Z^m_{T+h+1} = f^m_h(z_{1:T}, p_{T+1,\ldots,T+h}), . \tag{4}$$

Lastly, we get a sample for each base learner $\hat{z}^m_{T+h} \sim F_{Z^m_{T+h}}$, which can be operated in a recursive manner to generate a sample path $(\hat{z}^m_{T+1}, \ldots, \hat{z}^m_{T+H})$ for all base learners. Note that the auto-regressive feeding $(p_{T+1}, \ldots, p_{T+H})$ is the same across M base learners, but the corresponding sample $\hat{z}^m_{T+h} \sim Z^m_{T+h}$ are distinct on each base learner m.

[1] $R(s_h, a_h; z_{T+h})$ can be regarded as a random reward sampled from $r(s_h, a_h) := \mathbb{E}_{z \sim D_{T+h}}[R(s_h, a_h; z)]$ where D_{T+h} represents the conditional distribution of z_{T+h} given $z_{1:T}$ in the given time series dataset.



(a) Illustration of TS-GYM for training and inference with dynamic ensemble. **(b) Ensemble weights.**

Figure 2: Dynamic ensemble framework.

Construct ensembled quantile. Let's say we collect a set of sample paths $\{(\hat{z}^m_{T+1}, \ldots, \hat{z}^m_{T+H})_l\}_{l=1,m=1}^{L,M}$ where $(\hat{z}^m_{T+1}, \ldots, \hat{z}^m_{T+H})_l$ is l-th sample path above for the base learner m. For samples $\{(\hat{z}^m_{T+h})_l\}_{l=1}^{L}$ on each model m and prediction at h, we construct empirical marginal CDF $\hat{F}$ and estimate the quantile $\hat{q}^{\tau,m}_{T+h}(w)$ for each model. Finally, we get final ensemble from equation 2

$$\hat{q}^{\tau,es}_{T+h} = \sum_{m=1}^{M} w_h^m \hat{q}^{\tau,m}_{T+h}.$$

Remarks.

(1) Like p_{T+h} was sampled, the final ensemble model is ultimately a (single) auto-regressive one that supports sample path and quantiles.
(2) Under auto-regressive dynamic strategy, the ensembled sample p_{T+h} based on ensemble weight from policy affects the performance of individual base learner consecutively and thus final quantile ensemble. In other words, action in the previous step affects state in the current step, i.e., $\mathcal{P}(s_{h+1} \mid s_h, a_h) \neq \mathcal{P}(s_{h+1} \mid s_h)$. We denote our policy as `RL-auto`.
(3) If we don't feed ensembled sample p_{T+h} to auto-regressive learners, but instead do feed its own sample without ensemble, an ensemble or action does not affect the state in the dynamics, i.e., transition dynamics $\mathcal{P}(s_{h+1} \mid s_h, a_h) = \mathcal{P}(s_{h+1} \mid s_h)$. In this naive case, we call it *direct ensemble dynamics* and will compare in the experiment as an ablation study. We denote this naive policy as `RL-naive`.
(4) When some base learners are a type of Seq2Seq, ensemble dynmacis can still form a sample path in equation 3 which only requires sampling each h and use to feed auto-regressive type base learners, but p_{T+h} is ignored and does not affect Seq2Seq baselearners in equation 4. In the extreme case that all base learners are Seq2Seq, our ensemble dynamic is equivalent to *direct ensemble dynamics* (or `RL-naive`).

3.1.2 Reward function. To minimize the total quantile losses and encourage the agent to learn a uniform distribution over the nearly-optimal base learners, we design the reward function as $R(s,a;z) = R_1(s,a;z) + \lambda(s)R_2(s,a)$ for some $\lambda(s) \geq 0$. Here, the first term R_1 measures the performance of the current quantile ensemble predictions $\hat{q}^{\tau_k,es}_{T+h}$ compared with the best quantile predictions among the base learners. and takes the form

$$\begin{aligned} &R_1(s_h, a_h; z_{T+h}) \\ &= \min_m \left\{ \sum_{k=1}^{K} \left(\mathcal{L}(\hat{q}^{\tau_k,m}_{T+h}, z_{T+h}; \tau_k) - \mathcal{L}(\hat{q}^{\tau_k,\text{es}}_{T+h}, z_{T+h}; \tau_k) \right) \right\} \end{aligned} \tag{5}$$

where $\mathcal{L}(\cdot,\cdot;\tau)$ can be any measurement of the forecasting accuracy at the quantile level τ. By designing the R_1 term as a regret w.r.t. the best base learner, we normalize the reward around zero: if the R_1 term is less than 0, then it means that the ensemble prediction is worse than the single best base learner and the corresponding should be punished, and vise versa. Furthermore, R_2 takes the form

$$R_2(s_h, a_h) = D_{\text{KL}}(a_h \mid \text{Unif}(M^*(s_h))) \tag{6}$$

Here, D_{KL} denotes the Kullback–Leibler divergence, $M^*(s_h) = \{m \in M : \mathcal{L}^m(s_h) - \mathcal{L}^{m^\star}(s_h) \leq \epsilon\}$ where an optimal one $m^* = \arg\min_{m\in M} \mathcal{L}^m(s_h), L^m(s_h) = \sum_{k=1}^{K} \mathcal{L}(\hat{q}^{\tau_k,m}_{T+h}, z_{T+h}; \tau_k)$, and a threshold $\epsilon > 0$. denotes the set of nearly-optimal base learners at the state s, and $\text{Unif}(M^*(s))$ denotes a distribution with probability mass $\frac{1}{|M^*(s)|}$ on the indices corresponding to the base learners in $M^*(s)$ and 0 otherwise. We introduce the term R_2 to encourage the ensemble policy to be uniformly distributed among the nearly optimal base learners which could potentially further reduce the estimation error and the variance. Finally, $\lambda(s)$ is a state-dependent hyper-parameter controlling the weights between R_1 and R_2. When there is only a single nearly-optimal base learner, i.e., $|M^*(s)| = 1$, we set $\lambda(s) = 0$ which means that we only incorporate R_2 when there are at least two nearly-optimal base learners.

3.2 Simulated environment: TS-GYM

Before attempting to train the policy π, we first design a novel simulated environment for the time series ensemble, namely TS-GYM, that can simulate state transition (in Section 3.1.1) properly, by extending the OpenAI's gym interface. As illustrated in Figure 2a, it is composed of pre-trained base learners in the ensemble, time series (off-line) dataset, time series samplers, ensemble dynamics and dynamic ensemble agent. During the initialization stage of the environment $h = 1$ on each episode, it first decides the forecast start time T which is uniformly sampled among time horizon in off-line datasets. For example, suppose we have a single time series with length of NH where $N > 0$ is positive integer. Then initiazation choose the forecast time T uniformly among from 1 to $(N-1)H$ so that both $z_{1:T}$ and $z_{T+1:T+H}$ are accessible till the end of this episode. With these, on each step of an episode, TS-GYM provides following information: (1) sample a time series of (historical) observation $z_{1:T}$, (2) the quantile predictions $\{\hat{q}^{\tau_k,m}_{T+h}\}^{K,M}_{k=1,m=1}$ for the next timestamp $T+h$, (3) the step number h, and (4) ground-truth (future) observation z_{T+h}. The first three information is used to construct the state and the last information is used to construct the reward defined in Section 3.1.

Note here that generating all quantile predictions $\{\hat{z}^{\tau_k,m}_{T+h}\}^{K,M}_{k=1,m=1}$ at each timestamp $T+h$ is governed by the choice of ensemble dynamics in Section 3.1.1 where the ensembled quantile predictions themselves may be used for the base learners' prediction in the next timestamp. This will affect the optimal choice of ensemble actions in the end. This process is repeated until we reach the end of the prediction horizon $T+H$, completing one episode. In practice, this whole of procedure can be done with batch sampling or batch episodic learning over multiple items in parallel.

3.3 Learning dynamic ensemble policy with exploration

To learn an optimal ensemble policy π, we employ the deep actor-critic approach DDPG [9] in a continuous action space to maximize culmulative reward. To accelerate the exploration of the base learners' performance, we deploy the "random extreme point" exploration.

Random extreme point exploration. For the exploration of actions, for each step h, we assign the action $a_h = e_m \in \mathbb{R}^M$ where e_m is an one-hot vector[2] with randomly chosen m from $\mathcal{M}$ base learners. This exploration policy encourages the agent to take different individual base learners, efficiently collecting the observations on not only the sampled base learner performance but also various dynamic ensemble patterns. In addition this requires no prior knowledge on the base learners.

4 EXPERIMENTS

The extensive experiments are conducted to demonstrate the effectiveness of the proposed dynamic ensemble approach in adapting the ensemble strategy to the time series item and prediction timestamp in Section 4.1. Then, we spend to investigate properties of our ensemble methods from dynamic weights to the phenomena of boosting the performance of the auto-regressive base learner by feeding the better ensemble sample in Section 4.2.

[2]only m-th element equals one and zeros otherwise.

4.1 Benchmark experiments on dynamic ensemble

4.1.1 *Experiment setup.*

Datasets and base learners. We perform experiments on four real benchmark datasets that are widely used in forecasting literature: *exchange rate*, *elec*, *traf* and *solar* from [22]. For more dataset details, see appendix A.1. We consider the global deep learning based probabilistic forecasters from GluonTS [1]: `DeepAR` [23], `MQ-CNN` [20, 25], `NBEATS`[15], `TFT` [10] and `Transformer` [24], although many other forecasters can be easily included in our dynamic ensemble framework. Since the performance of `DeepAR` can be heavily dependent on the distribution outputs, we trained `DeepAR` with three different distribution outputs: Gaussian, Student's t and Poisson distribution referred as `DeepAR-G`, `DeepAR-T` and `DeepAR-P`, respectively. All base learners are trained using the default configurations in GluonTS [1] .

RL training. To evaluate the performance of our `RL-auto` policy, which leverage ensemble dynamic in an autoregressive ways. RL algorithm (DDPG) is implemented in PyTorch [21] and trained on AWS Sagemaker [8] with `ml.p3.2xlarge` instances. We differentiate, train and test environment with TS-GYM by providing corresponding train and test splits.

Ensemble baselines. We compare our RL-based dynamic ensemble approach with the following static ensemble baselines:

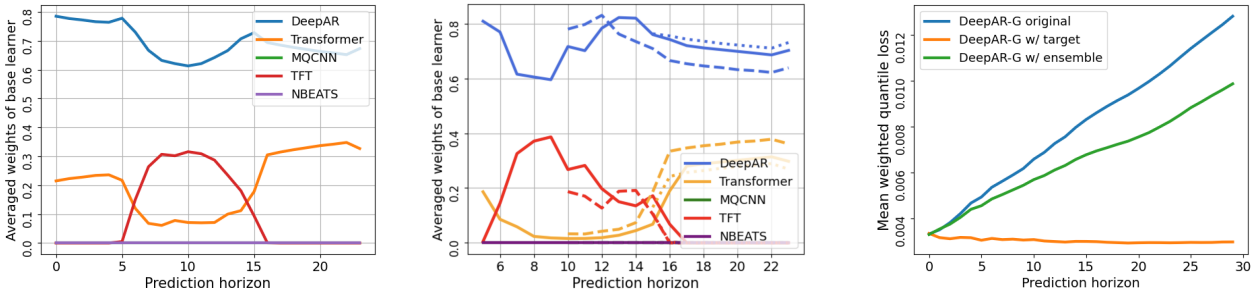
- Mean/Median: for each item and timestamp, take a simple mean/median of all base learners.
- Global optimal ensemble: of all of the possible weights of base learners which are shared across items and timestamps, choose the weight for which the associated convex combinations of base learners lead to the best performance in the backtest validation set.
- `Winner-takes-all` (WTA): choose the single base learner which leads to the best performance in the backtest validation set.

4.1.2 *Benchmark results.*

Message 1: Our auto-regressive dynamic ensemble, `RL-auto`*, is the best or at least on par against other 4 baselines.* We evaluate the time series forecasting results by the mean weighted quantile loss defined in Equation 7 in the appendix. The results of all dynamic ensemble approaches including ours are summarized in Table 1. From the results in Table 1, we can further report three metrics, winning rate, average ranking, and averaged stability score (amount of % degradation compared with winning method). For winning rate, our `RL-auto` ensemble wins **3 out of 4** datasets against other baselines and Median ensemble won 1 of them. In the average ranking (lower is better), `RL-auto` and Median method is **1.5 ranking** and 2.25 ranking respectively whereas mean and WTA method is 3.25 and 3.25 respectively. In terms of stability score, our `RL-auto` and Median ensemble is only **-2.5%** and -17% degradation against optimal strategies respectively whereas Mean and WTA method is at least -100% and -70%.

Base learner/ Ensemble strategy	Exchange Rate	Electricity	Traffic	Solar
DeepAR-T	0.0075	0.0548	0.0814 (0, 0861)	0.3252
DeepAR-G	0.0067	0.0618	0.1087(0.1028)	0.3117
DeepAR-P	0.2261	0.0910	0.9828(0.9757)	0.3137
Transformer	0.0298	0.0266	0.0822(0.0905)	0.3584
MQ-CNN	0.0133	0.0544	0.1073(0.1480)	0.7735
TFT	0.0060	0.0844	0.0896(0.1000)	0.3253
NBEATS	0.0106	0.0480	0.2153(0.2139)	0.9983
Mean	0.0359	0.0490	0.4171	0.3790
Median	0.0090	**0.0489**	0.0907	0.3256
Global optimal	0.0124	0.0790	0.1820	0.3913
Winner-takes-all (WTA)	0.0133	0.0548	0.0814	0.7735
RL-auto policy (Ours)	**0.0060**	0.0544	**0.0813**	**0.3058**

Table 1: Performance comparison on real-world benchmark datasets. The winning method among ensemble methods are made bold. The retangular is the one selected in Winner-takes-all ensemble method. The values in the parenthesis are the accuracy evaluated in the backtesting window.



(a) Learning time-varying weights. **(b) Adaptive to forecasting start time.** **(c) Reducing error accumulation.**

Figure 3: Three properties of dynamic ensemble: time-varying, adaptive, and reducing accumulating errors. The ensemble weights and the quantile loss are averaged over all items in Solar dataset.

Message 2: Our RL-auto *can generalize to unseen items or new forecasting time among learnable stategies.* For the case of unseen item during ensembling phase, WTA and global optimal ensemble that relies on backtest window is not feasible.

4.2 Investigating properties of dynamic ensemble

Property 1: Capturing time-varying ensemble weights. We first demonstrate the capability of our dynamic ensemble framework to learn the time-varying ensemble weights when the optimal base learners vary along the prediction horizon. We examine policy trained on the motivating example on the dataset *Solar* in Section 1 more closely. Our dynamic ensemble approach is able to learn ensemble weights which are consistent with the time-varying pattern of the optimal base learners. In particular, we can see from Figure 3a that (1) only Transformer, TFT and DeepAR are given positive ensemble weights during the prediction, (2) the ensemble weights of transformer remain relatively high in prediction timestamps $[0, 6] \cup [16, 29]$ while dropping below 0.1 during prediction timestamps $[7, 15]$, (3) the ensemble weights of TFT remain 0 in prediction timestamps $[0, 5] \cup [16, 29]$ but dominate the ensemble weights of transformer in prediction timestamps $[7, 15]$, (4) the ensemble weights of DeepAR remain high during the entire prediction horizon because its relatively good performance during the entire prediction horizon.

Property 2: Adaptive to forecasting start time. The traditional ensemble methods highly rely on training the ensemble weights with the data in backtesting window and then using the learned ensemble weights for the targeted prediction window. Such methods only work for a fixed forecast start time and rely on the consistency between the data in backtesting and targeted prediction window. If there is a pattern shift in terms of the forecasters' performance between the backtesting and the targeted prediction window, the learned ensemble weights will be sub-optimal for the targeted prediction window or even worse than the simple average method. On the contrary, since we train the optimal ensemble policy from the entire historical data instead of relying only on the backtesting window, our dynamic ensemble framework can adaptively predict the optimal ensemble weights based on the given context information and the current predictions from the base learners. Figure 3b shows the policy which generates the ensemble weights in Figure 3a, but for the different forecasting start time. The solid, dashed, and dotted lines correspond to the forecasting start time with the

Base learner /Ensemble strategy	DeepAR-T	DeepAR-G	DeepAR-P	Mean	Global Optimal	RL-naive	RL-auto
solar	0.3252	0.3117	0.3137	0.3088	0.3302	0.3148	**0.2840**

Table 2: Ablation study for the effect of auto-regressive feeding of ensembled sample. RL-auto predicts bette than RL-naive.

timestamp 5, 10, 15 in the prediction horizon of Figure 3a. We can see that the learned policy can adapt the ensemble weights for any forecasting start time.

Property 3: Reducing error accumulation of auto-regressive forecasters. Improving the base learners' performance is important for the improving the accuracy of the final ensembled predictions, and for allowing a broader set of admissible ensemble polices (in the extreme case, if all base learners perform equally well, then any ensemble strategy is optimal). We demonstrate the capability of auto-regressive ensemble (as shown in Figure 3c) on boosting the performance of AR forecasters. In particular, we focus on the DeepAR models with different distribution outputs: Gaussian , Student's t and Poisson distribution and train the ensemble policy using our dynamic ensemble approach with auto-regressive ensemble dynamics on *exchange rate* dataset. Figure 3c shows the mean weighted quantile losses of the DeepAR-G over the prediction horizon for 3 different strategies:

- using DeepAR with Gaussian distribution (denoted as DeepAR-G original);
- using DeepAR with Gaussian distribution, but feed the true target value as the auto-regressive input in Equation 4 (denoted as DeepAR-G w/ target);
- using the DeepAR with Gaussian distribution, but feed the samples from the mixture of distributions in Equation 3 as the auto-regressive input in Equation 4 (denoted as DeepAR-G w/ ensemble);

We can observe that by feeding a more accurate input to the auto-regressive forecaster, DeepAR-G w/ ensemble improves DeepAR-G original consistently over the entire prediction horizon. The mean weighted quantile loss for DeepAR-G original and DeepAR-G w/ ensemble are 0.01466 and 0.00988, respectively, which demonstrates a 32.6% performance boost.

4.3 Ablation study

We conduct the ablation on AR dynamics that is explicitly considered in our algorithm in comparison to the methods where the AR feedback is not explicit. We term these ablations as RL-auto and RL-naive. We consider the *solar* dataset with base learners DeepAR-T, DeepAR-G and DeepAR-P.

Table 2 highlights the significance of AR dynamics that is explicit in our MDP formulation. With same set of base learners the AR dynamics is able to achieve **11**% better result than the naive dynamics. Further, the RL-auto is better (**8**%) than all models/ensemble strategy considered, thus showing the significance of base learner boosting via AR feedback.

5 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we propose a novel dynamic ensemble framework from the lens of deep RL which are able to adapt the ensemble strategy to the time series item and prediction timestamp. In addition, we explore the potential of the ensemble prediction on boosting the performance of the auto-regressive base learner and improve their performance for longer horizons. The effectiveness of the proposed approach is validated by experiments on real-world data. The current approach considers an RL policy for each dataset which is still restrictive with respect to the real potential of RL. In future work, extension to multiple dataset is a potential direction that can combine the generalization ability of the RL policy across different datasets. Further, metadata about the dataset and any context can be embedded in the MDP state to account for inter-data variations.

REFERENCES

[1] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Sundar Rangapuram, David Salinas, Jasper Schulz, et al. GluonTS: Probabilistic and neural time series modeling in Python. *Journal of Machine Learning Research*, 21 (116):1–6, 2020.

[2] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys (CSUR)*, 2022.

[3] David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec. Network inference via the time-varying graphical lasso. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 205–213, 2017.

[4] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proceedings 2001 IEEE international conference on data mining*, pp. 289–296. IEEE, 2001.

[5] Jongho Kim, Youngsuk Park, John D Fox, Stephen P Boyd, and William Dally. Optimal operation of a plug-in hybrid vehicle with battery thermal and degradation model. In *2020 American Control Conference (ACC)*, pp. 3083–3090. IEEE, 2020.

[6] Paul D Larson. Designing and managing the supply chain: concepts, strategies, and case studies. *Journal of Business Logistics*, 22(1):259, 2001.

[7] Julie Letchner, Christopher Ré, Magdalena Balazinska, and Matthai Philipose. Access methods for markovian streams. In *2009 IEEE 25th International Conference on Data Engineering*, pp. 246–257. IEEE, 2009.

[8] Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, et al. Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 731–737, 2020.

[9] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[10] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[11] Michael Mathioudakis, Nick Koudas, and Peter Marbach. Early online identification of attention gathering items in social media. In *Proceedings of the third ACM international conference on Web search and data mining*, pp. 301–310, 2010.

[12] Yasuko Matsubara, Yasushi Sakurai, B Aditya Prakash, Lei Li, and Christos Faloutsos. Rise and fall patterns of information diffusion: model and implications. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 6–14, 2012.

[13] Yasuko Matsubara, Yasushi Sakurai, Naonori Ueda, and Masatoshi Yoshikawa. Fast and exact monitoring of co-evolving data streams. In *2014 IEEE International Conference on Data Mining*, pp. 390–399. IEEE, 2014.

[14] Yasuko Matsubara, Yasushi Sakurai, Willem G van Panhuis, and Christos Faloutsos. Funnel: automatic mining of spatially coevolving epidemics. In *KDD*, pp. 105–114. ACM, 2014.
[15] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv:1905.10437*, 2019.
[16] Spiros Papadimitriou and Philip Yu. Optimal multi-scale patterns in time series streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pp. 647–658, 2006.
[17] Youngsuk Park, Kanak Mahadik, Ryan A Rossi, Gang Wu, and Handong Zhao. Linear quadratic regulator for resource-efficient cloud services. In *Proceedings of the ACM Symposium on Cloud Computing*, pp. 488–489, 2019.
[18] Youngsuk Park, Ryan Rossi, Zheng Wen, Gang Wu, and Handong Zhao. Structured policy iteration for linear quadratic regulator. In *International Conference on Machine Learning*, pp. 7521–7531. PMLR, 2020.
[19] Youngsuk Park, Danielle Maddix, François-Xavier Aubet, Kelvin Kan, Jan Gasthaus, and Yuyang Wang. Learning quantile functions without quantile crossing for distribution-free time series forecasting. *arXiv:2111.06581*, 2021.
[20] Youngsuk Park, Danielle Maddix, François-Xavier Aubet, Kelvin Kan, Jan Gasthaus, and Yuyang Wang. Learning quantile functions without quantile crossing for distribution-free time series forecasting. In *International Conference on Artificial Intelligence and Statistics*, pp. 8127–8150. PMLR, 2022.
[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
[22] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in Neural Information Processing Systems*, 32:6827–6837, 2019.
[23] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.
[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
[25] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.
[26] Yunyue Zhu and Dennis Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pp. 358–369. Elsevier, 2002.

A EXPERIMENT SETUP

A.1 Real-world dataset

Table 3 summarizes the four benchmark real-world datasets that we use to evaluate our dynamic ensemble approach.

Dataset	Freq	Domain	# Time series	Prediction length
exchange rate	daily	$\mathbb{R}^+$	40	30
elec	hourly	$\mathbb{R}^+$	2950	24
traf	hourly	$[0, 1]$	6741	24
solar	hourly	$\mathbb{R}^+$	959	24

Table 3: Benchmark dataset descriptions

A.2 Implementation of DDPG

We use the DDPG implementation from OpenAI spinning up baselines. The last layer of policy network is a softmax layer with output dimensions as the number of base learners considered. For hyper-parameter tuning we consider the hyper-parameters in [9] and some specific to dynamic AR ensemble. The final hyper-parameters used for different datasets for the experiment in Section 4.1 is given in Tables 4 and 5. The default weights among AR model parameter is used to set the weights among the AR model if all the AR models in the hybrid dynamics gets zero weight at certain step in the RL; λ controls the trade-off as explained in the reward function section. The reward scale is the scaling applied to mean-wQL to be comparable with the secondary reward r_2. Round threshold is the number of decimal digits for rounding the mean-wQL to get ranking for base learners.

Hyperparamters	*exchange rate*	*elec*	*traf*	*solar*
episodes per epoch	5	5	5	5
start episodes	40	50	50	50
update after episodes	5	5	5	5
update steps per prediction length	4	4	4	4
update every episodes	0.5	0.25	0.25	0.5
discount factor	0.99	0.99	0.99	0.99
epochs	40	60	60	70
polyak	0.99	0.99	0.99	0.99
learning rate for policy	0.0005	0.0005	0.0005	0.0005
learning rate for Q value	0.0005	0.0005	0.0005	0.0005
noise level for action	0.05	0.05	0.05	0.1

Table 4: Hyperparameters of DDPG algorithm in various real-world datasets.

Hyperparameters	*exchange rate*	*elec*	*traf*	*solar*
train batch size	40	200	100	200
reward scale	100	0.0001	10	0.01
round threshold	2	2	2	2
λ	0.5	0.5	0.5	0.5
default weights among auto-regressive models	$[1, 0, 0]$	$[1, 0, 0]$	$[1, 0, 0]$	$[1, 0, 0]$

Table 5: Hyperparameters of TS-GYM in various real-world datasets.

A.3 Error metric in TS-GYM

We evaluate with the mean weighted quantile loss.

$$\frac{1}{q} \frac{\sum_{i=1,j=T+1,k=1}^{N,T+h,q} \max\left\{\tau_k(z_{i,j} - \widetilde{z}_{i,j,k}), (1-\tau_k)(\widetilde{z}_{i,j,k} - z_{i,j})\right\}}{\sum_{i=1,j=T+1}^{N,T+h} |z_{i,j}|} \quad (7)$$

where $\{z_{i,j}\}_{i=1,j=T+1}^{N,T+h}$ are the true values of future time series and $\{\widetilde{z}_{i,j,k}\}_{i=1,j=T+1,k=1}^{N,T+h,q}$ are the estimated quantile predictions.