DYNAMIC ENSEMBLE FOR PROBABILISTIC TIME-SERIES FORECASTING VIA DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

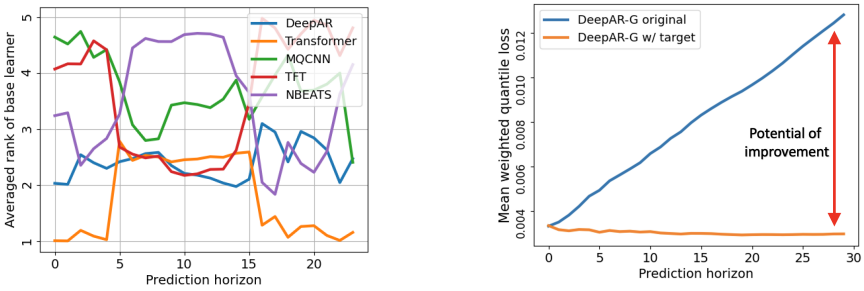
It is well known that ensemble improve the accuracy of forecasting tasks. However, most of ensembling strategies designed for probabilistic time series forecasting are *static* methods, in the sense that they either assume the time-invariant ensemble strategies over the prediction horizon, or are non-adaptive to the forecast start point. In addition, the static methods naively rely on the predictions of the base forecasters but fail to utilize base learners themselves efficiently. In this paper, we propose a novel *dynamic ensemble policy* to overcome three major limitations mentioned above via deep Reinforcement Learning (RL) framework. To learn such a policy, we design a Markov Decision Process (MDP), together with our environment (*TS-GYM*) that supports the interaction between the agent or ensembler, offline datasets and base learners. In doing so, we effectively leverage the power of the ensemble to improve each of the base learners by reducing the error accumulation of each base learner via consecutively feeding a better ensembled sample to each base learner. The proposed ensembling method has several desirable properties such as uncertainty quantification and the ability to generate sample path, on top of significant performance gain. The effectiveness of the proposed framework is demonstrated on multiple synthetic and real-world experiments.

1 INTRODUCTION

Time series data occur naturally in countless domains including supply chain optimization (Larson, 2001; Wen et al., 2017), medical analysis (Keogh et al., 2001; Matsubara et al., 2014b), financial analysis (Zhu & Shasha, 2002; Hallac et al., 2017), sensor network monitoring (Papadimitriou & Yu, 2006; Letchner et al., 2009), cloud computing (Park et al., 2019; 2021), optimal control of vehecle (Kim et al., 2020) and social activity mining (Mathioudakis et al., 2010; Matsubara et al., 2012; 2014a). 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 (Benidis et al., 2022).

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 dataset *Solar*. In this example, 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 has 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 (Salinas et al., 2020). 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 implementation of DeepAR) over the prediction horizon on *exchange rate* dataset. On the other hand, if we can provide base learners such as DeepAR with more accurate estimations of the future as the auto-regressive input, the prediction error can be significantly decreased for the long horizon predictions (see the orange curve in Figure 1b). The huge difference in the prediction error between these two cases show the huge potential to improve the auto-regression based models if we can provide more accurate estimations during the prediction horizon. However, none of the traditional ensemble methods utilize the ensemble predictions as the feedback to boost 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?*



(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" and "DeepAR w/ target" shows the potential improvement we can gain if the accuracy of the auto-regressive input to DeepAR can be improved.

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

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 RELATED WORK

Probabilistic time series forecasting In recent years there has been an increasing interest in "probabilistic forecasting", namely forecasting models that account for the data's uncertainty by modeling the distribution of target values, rather than predicting a single point estimate. Probablistic

forecasting is useful for business purposes such as supply and demand, inventory management, staff scheduling and topology planning (Larson, 2001). Modern open source packages such as Kats (facebookresearch, 2021), Merlion (Bhatnagar et al., 2021) and GluonTS (Alexandrov et al., 2020a) offer probabilistic forecasting, and include some popular probabilistic forecasters such as Prophet (Taylor & Letham, 2018), and deep learning probabilistic forecasters such as DeepAR (Salinas et al., 2020), MQ-CNN (Wen et al., 2017; Park et al., 2022), MQF2 (Kan et al., 2022), NBEATS (Oreshkin et al., 2019), TFT (Lim et al., 2021) and Transformer (Vaswani et al., 2017). There are several advances in improving those models in adversarial robustness (Yoon et al., 2022; Liu et al., 2022) and few-shot learning (Jin et al., 2022).

Time series ensemble The literature on ensembling methods for time series predictions have focused solely on static ensembling strategies, namely ones that have access to the predictions of the base learners but not to the base learners themselves. In that situation, a debate on the theory of ensembling for time series was sparked by an empirical observation that a simple average of the base learners is often superior to more sophisticated ensemble methods (a problem called the "forecast combination puzzle", see Stock & Watson (2004) and Bates & Granger (1969)). See Smith & Wallis (2009), Claeskens et al. (2016), and Elliott (2011)). While theory lags, however, sophisticated static ensembling methods have often been observed to work well. (See Donaldson & Kamstra (1996), Moon et al. (2020), and Massaoudi et al. (2021). Particularly interesting is Gastinger et al. (2021), with a large empirical study.)

Contrary to the situation considered in these papers, literature on ensembling methods that have direct access to the base learners, rather than only to their predictions, is limited. Recently, RL based approaches are proposed in Saadallah & Morik (2021) and Fu et al. (2022). Saadallah & Morik (2021) consider action dependent state (window of ensemble predictions) transition. Their work focus on online policy learning with update timing determined by a concept-drift detection algorithm. In Fu et al. (2022) the state (time series for a given context window and base learners performance at the next window) transition is action independent with action taken for H steps at a time. In addition, their methods are only designed for the point based forecasting problem and do not demonstrate the capability of capturing the non-stationary ensemble weights.

3 PRELIMINARIES

3.1 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 $x_{i,t} \in \mathbb{R}^d$, as t varies over time at fixed discrete intervals. For an i-th time series (often called i-th item), we wish to make predictions for the next H timestamps, 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 $x_{i,1:T+H}$. 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. For notational simplicity we will drop the item index i and covariates $x_{i,t}$ unless explicitly stated. We now formally define a forecasting model as a set of random variable valued functions $\{f_h\}_{h=1}^{H}$ such that, for $h = 1, \ldots, H$

$$Z_{T+h} = f_h(z_{1:T}, \quad_{T+h-1}); \tag{1}$$

where $_{T+h-1}$ is the hidden state variable passed from the previous (or older) step. The evolution of f_h and $_{T+h-1}$ depend on the type of the base model. For the auto-regressive model which uses the recursive prediction strategy, the hidden state $_{T+h-1}$ is generated by passing a sample $\hat{z}_{T+h-1} \sim Z_{T+h-1}$ from previous time step to the forecaster decoder for the next prediction in a recursive manner. Often the decoder is homogeneous, i.e., $f_h = f$ for $h = 1, \ldots, H$. On the other hand, Seq2Seq model which uses the direct prediction strategy, directly forecast the future time series without involving the evoluation of the hidden state, i.e., $_{T+h-1} = _{T}$ for all $h = 1, \ldots, H$. Refer to Alexandrov et al. (2020b) for the detailed modeling. In Section 4, we will explore a different choice for the auto-regressive step, using the entire ensemble.

Then, the associated -quantile predictions can be followed as $\hat{z}_{T+h} = q\ (Z_{T+h})$ where, for a random variable $Z \in \mathbb{R}$ with its culmulative distribution F_Z and a quantile level $\in (0, 1)$, q is denoted as the quantile function, i.e., $q\ (Z) := F_Z^{-1}(\ ) = \inf\{z \in \mathbb{R} : \ \leq F_Z(z)\}$.

3.2 FORECASTING ENSEMBLE

For each m-th base learner, we denote $\hat{z}_{T+h}^{k;m}$ as the $_k$-quantile prediction at time step $T+h$ on a quantile level where $_k \in \{\ _k\}_{k=1}^{K}$. Then, $\{\hat{z}_{T+h}^{k;m}\}_{k=1;m=1}^{K;M}$ 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 $h = 1;\dots;H$,

$$\hat{z}_{T+h}^{;\text{es}} = \sum_{m=1}^{M} w_h^m \hat{z}_{T+h}^{;m}; \tag{2}$$

where $w_h^m \geq 0$ with $\sum_{m=1}^{M} w_h^m = 1$ are the ensemble weights.

3.3 REINFORCEMENT LEARNING

Reinforcement learning (RL) is usually formulated as a Markov Decision Process (MDP), which can be defined as a tuple $(\mathcal{S};\mathcal{A};\mathcal{P};r;\ ;H)$ 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, $r:\mathcal{S}\times\mathcal{A}\to\mathsf{R}$ is the reward function, $\in(0;1)$ 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 $:\mathcal{S}\to\mathcal{A}$ that maximizes the total obtained rewards $\max\ J(\ ) = \mathsf{E}\left[\sum_{h=0}^{H-1}\ ^h r(s_h;a_h)\middle|\ \right]$, where the expectation is over the trajectory $=\{(s_0;a_0;r(s_0;a_0)):\dots:(s_H;a_H;r(s_H;a_H))\}$ where $a_h = \ (s_h)$.

4 DYNAMIC ENSEMBLE FRAMEWORK

In this section, we mainly focus on how to select a sequence of ensemble weights $(w_1;w_2;\dots;w_H)$ with $w_h\in\mathsf{R}^M$ over M base learners by learning a ensemble policy . Especially in the presence of auto-regressive base learners, ensemble weights chosen at the step h may affect the forecasting of auto-regressive base learners and also ensemble weights chosen at the next step $h+1$ (see Section 4.1.1 for more details). With this intuition, we will take a reinforcement learning approach to learn an optimal policy function that provides the optimal ensemble weights sequentially.

In Section 4.1, 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 4.1.1 and the careful design of reward computation is explained in Section 4.1.2. Based on the formulated MDP, we then design our simulated environment, TS-GYM (in Section 4.2) 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 4.3.

4.1 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 fixes an arbitrary forecasting start point T, and then starts to provide a time series pair of both historical input $z_{1:T}$ and corresponding future (backtest) output z_{T+1} as well as corresponding quantile predictions $\{\hat{z}_{T+1}^{;m}\}$ from alll M base models for the next step $T+1$. (We defer the details implementation of the environment to Section 4.2). The agent will then decide the ensemble weights to compute the ensembled predictions, and update the ensemble policy based on the accuracy of the ensembled predictions. Depending on the type of ensemble dynamics, the ensembled predictions may also affect the base learners' future predictions. Then, in the next step $h=2$, the environment provides next time series output z_{T+2} and associated predictions $\{\hat{z}_{T+2}^{;m}\}$ and go on. See Figure 2a for a high level schema.

More formally, for each step $h = 1; \cdots; H$ of an episode, given the information provided by the environment (e.g., historical observation $z_{1:T}$, and future (backtest) observation z_{T+h}, a pool of all quantile predictions $\{\hat{z}_{T+h}^{k;m}\}_{k=1;m=1}^{K;M}$, and step h) , we define MDP as follows:

- the fixed-size state $s_h = \left\{z_{1:T}; \{\hat{z}_{T+h}^{k;m}\}_{k=1;m=1}^{K;M}; h\right\}$,
- the action $a_h = \{w_h^m\}_{m=1}^M =$ (s_h), M-ensemble weights w_h^m from a policy function ,
- the state transition $\mathcal{P}(s_{h+1} \mid s_h; a_h)$ governed by ensemble dynamics in Section 4.1.1,
- the reward $R(s_h; a_h; z_{T+h})$ [1] which evaluates ensemble prediction against ground-truth z_{T+h} in Section 4.1.2 .

4.1.1 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 $\{\hat{z}_{T+h}^{k;m}\}_{k=1;m=1}^{K;M} \in s_h$. Here, we proposed three strategies: direct dynamic, auto-regressive dynamic and their composition. The idea of direct ensemble is similar to Seq2Seq models which employs the direct prediction strategy. The idea of auto-regressive dynamic is based on auto-regressive models where you recursively feed a new ensembled sample to each base learner for the next prediction. The ensemble dynamics appear at the step represented by the red arrow line in Figure 2a.

Direct dynamic. As a direct ensembling over base learner, we first compute quantiles by base learner itself over H horizon, i.e., we compute $\hat{z}_{T+h}^{;m} = q\ (Z_{T+h}^m)$ for all $h = 1; \cdots; H$, based on Equation 1. Then the final quantile ensemble becomes $\hat{z}_{T+h}^{;es} = \sum_{m=1}^M w_h^m \hat{z}_{T+h}^{;m}$ in Equation 2. Note that the base learner's predictions are not affected by the ensembling. In other words, the transition dynamic $\mathcal{P}(s_{h+1} \mid s_h; a_h) = \mathcal{P}(s_{h+1} \mid s_h)$ is actually irrelevant to the ensembling weights.

Auto-regressive dynamic. 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 through which we can compute the final ensembled (empirical) quantile prediction $\hat{z}_{T+h}^{;es}$.

To begin with, we generate a sample path $(\hat{z}_{T+1}^m; \cdots; \hat{z}_{T+H}^m)$ for each base learner as follows: First, for each step h, we sample p_{T+h} from mixture of base learners' distributions $\mathrm{P}(Z_{T+h}^m)$ proportional to ensemble weights w_h^m, i.e.,

$$p_{T+h} \sim \sum_{m=1}^M w_h^m \mathrm{P}(Z_{T+h}^m): \tag{3}$$

Second, we feed p_{T+h} to each autoregressive base learner, i.e.,

$$Z_{T+h+1}^m = f^m(z_{1:T};\ {}_{T+h}^m); \tag{4a}$$

$${}_{T+h}^m = g^m(p_{T+h};\ {}_{T+h-1}^m): \tag{4b}$$

where g^m represents the m-base model's evolution dynamics for the hidden state ${}_{T+h}^m$. Lastly, we get a sample for each base learner $\hat{z}_{T+h}^m \sim Z_{T+h}^m$, which can be operated in a recursive manner to generate a sample path $(\hat{z}_{T+1}^m; \cdots; \hat{z}_{T+H}^m)$ for all base learners.

After collecting a set of sample paths $\{(\hat{z}_{T+1}^m; \cdots; \hat{z}_{T+H}^m)_l\}_{l=1;m=1}^{L;M}$ where $(\hat{z}_{T+1}^m; \cdots; \hat{z}_{T+H}^m)_l$ is l-th sample path above for the m-base learner, we construct the empirical marginal distribution $\hat{\mathrm{P}}(\hat{Z}_{T+h}^m)$ based on the samples $\{(\hat{z}_{T+h}^m)_l\}_{l=1}^L$ for all $h = 1; \cdots; H$. Then, the final (ensemble-dependent) quantile prediction of each base learner is obtained as $\hat{z}_{T+h}^{;m}(w) = q\ \left(\hat{\mathrm{P}}(\hat{Z}_{T+h}^m)\right)$ for all $m = 1; \cdots; M$ with the final ensemble $\hat{z}_{T+h}^{;es} = \sum_{m=1}^M w_h^m \hat{z}_{T+h}^{;m}$. Note that, like p_{T+h} was sampled, the final ensemble model is ultimately a (single) auto-regressive one that supports sample path and quantiles.

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, meaning, unlike the direct dynamic, the transition dynamics $\mathcal{P}(s_{h+1} \mid s_h; a_h) \neq \mathcal{P}(s_{h+1} \mid s_h)$.

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

Hybrid dynamic. Note that the auto-regressive dynamic strategy is not applicable for Seq2seq base learners. Still, under the hybrid dynamic strategy, Seq2seq base learners can contribute to generate ensembled samples together, i.e., ensembled sample $p_{T+H} \sim \sum_{m=1}^{M} w_h^m \mathrm{P}(Z_{T+h}^m)$ sampled from both Seq2seq and autoregressive ones, which would be fed into (only) auto-regressive base learners. The behaviours of Seq2seq base learner is the exactly same in sampling and constructing quantile prediction without any feedback loop like ensembled sample, which means any auto-regressive base learners does not affect Seq2seq one's prediction.The final ensemble under hybrid dynamic is capable of auto-regressive model, supporting desirable sample path through recursive feedings.

4.1.2 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 $R(s;a;z) = R_1(s;a;z) + \quad (s)R_2(s;a)$ for some $\quad (s) \geq 0$. Here, the first term r_1 measures the performance of the current quantile ensemble predictions $\hat{z}_{T+h}^{k;\mathrm{es}}$ compared with the best quantile predictions among the base learners. and takes the form

$$R_1(s_h;a_h;z_{T+h}) = \min_m \left\{ \sum_{k=1}^{K} \left(\mathcal{L}(\hat{z}_{T+h}^{k;m}; z_{T+h};\ _k) - \mathcal{L}(\hat{z}_{T+h}^{k;\mathrm{es}}; z_{T+h};\ _k) \right) \right\} \tag{5}$$

where $\mathcal{L}(\cdot;\cdot;\ )$ can be any measurement of the forecasting accuracy at the quantile level $\ $. 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

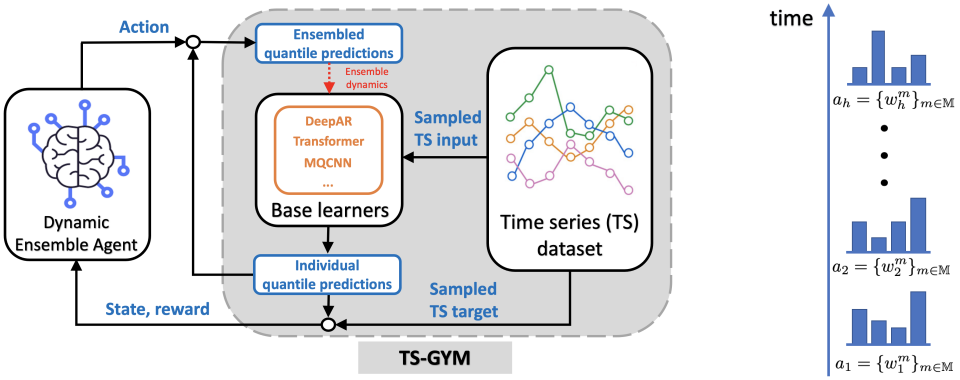
$$R_2(s_h;a_h) = D_{\mathrm{KL}}(a_h \mid \mathrm{Unif}(M^*(s_h))) \tag{6}$$

where D_{KL} denotes the Kullback–Leibler divergence, $M^*(S) = \{m \in M : \mathcal{L}(m) - \mathcal{L}(m^*) \leq \ \}$ for a threshold $\ > 0$. denotes the set of nearly-optimal base learners at the state s, and $\mathrm{Unif}(M^*(s))$ 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, $\quad (s)$ 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 $\quad (s) = 0$ which means that we only incorporate R_2 when there are at least two nearly-optimal base learners.

4.2 SIMULATED ENVIRONMENT: TS-GYM

Before attempting to train the policy $\ $, we first design a novel simulated environment for the time series ensemble, namely TS-GYM, that follows state transition (in Section 4.1.1) 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 $h = 1$, it first decides forecast start time T which is uniformly sampled among time horizon in off-line datasets, and then starts to provide following information: (1) sample a time series of (historical) observation $z_{1:T}$, (2) the quantile predictions $\{\hat{z}_{T+h}^{k;m}\}_{k=1;m=1}^{K;M}$ 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 4.1.

Note here that generating all quantile predictions $\{\hat{z}_{T+h}^{k;m}\}_{k=1;m=1}^{K;M}$ at each timestamp $T+h$ is governed by the choice of ensemble dynamics in Section 4.1.1 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 in parallel.



(a) Illustration of TS-GYM and ensemble dynamic inside. (b) Sequential weights.

Figure 2: Dynamic ensemble framework.

4.3 Learning dynamic ensemble policy with exploration

To learn an optimal ensemble policy , we employ the deep actor-critic approach DDPG (Lillicrap et al., 2015) 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 $a_h = e_m \in \mathsf{R}^M$ 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.

5 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 5.1. 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 5.2.

5.1 Benchmark experiments on dynamic ensemble

5.1.1 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 (Salinas et al., 2019). For more dataset details, see appendix A.1. We consider the global deep learning based probabilistic forecasters from GluonTS (Alexandrov et al., 2020b): `DeepAR` (Salinas et al., 2020), `MQ-CNN` (Wen et al., 2017; Park et al., 2022), `NBEATS`(Oreshkin et al., 2019), `TFT` (Lim et al., 2021) and `Transformer` (Vaswani et al., 2017). 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 (Alexandrov et al., 2020b) .

MDP formulation and RL training To evaluate the performance of our general dynamic ensemble framework, we take the most general ensemble dynamics, which is the *hybrid quantile ensemble dynamics*. In particular, we will apply the auto-regressive ensemble dynamics to the `DeepAR` models with different distribution outputs and apply the direct ensemble dynamics to the rest of the base learners. The samples from the `DeepAR` models from the previous timestamps will then recursively

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

feed as the input to `DeepAR` models at the next timestamps. In defining the reward function, we adopt the *mean weighted quantile loss* (see Equation 7 in Appendix) as the accuracy measurement of our predictions. RL algorithm (DDPG) is implemented in PyTorch (Paszke et al., 2019) and trained on AWS Sagemaker (Liberty et al., 2020) with `ml.p3.2xlarge` instances. Train and test are done with TS-GYM specific to the given dataset.

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.

5.1.2 BENCHMARK RESULTS

Message 1: Our hybrid dynamic ensembles 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 Equation 7 in the appendix. The results of all dynamic ensemble approaches including our hybrid quantile ensemble dynamics 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 winining method). For winning rate, our `RL-hybrid` ensemble is 50% (wins in two out of four datasets) against other 4 baselines whereas Median and `Winner-takes-all` ensemble won 25% respectively. In the average ranking, Median and our `RL-hybrid` method is 1.75 and 2 respectively whereas Mean and WTA method is 3.75 and 3.5 respectively. In terms of stability score, our `RL-hybrid` and Median ensemble is -10% and -15% respectively whereas Mean and WTA method is at least -100% and -70%. Please see more detailed analysis dataset by dataset in Appendix B.

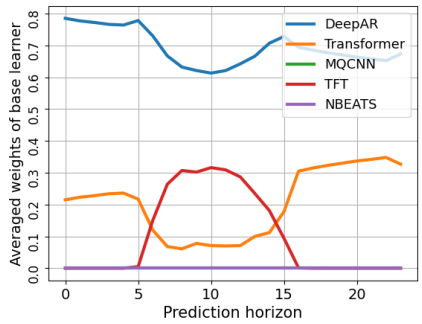
Message 2: Overfitting and distribution shift hinders coherent ensembles over all ensemble methods. We also observe the over-fitting of some base learners from the results of `Winner-takes-all`. In *exchange rate*, *elec* and *solar* datasets, the best base learner in the backtest validation set is not the best base learner in the prediction testing window. It would be challenging to learn a good ensemble strategy in this situation. However, our approach can overcome this over-fitting issue to some extend and still be able to learn good ensemble policy for *exchange rate* and *solar* datasets. This is partially because the ensemble policy is trained using the entire time series dataset instead of just the backtest window. In addition, although `Winner-takes-all` gives the best forecasting accuracy for *traf*, the severe over-fitting of `MQ-CNN` (see accuracies inside parenthesis of Table 1) slightly degrades the performance of our approach since the uniform weights are encouraged for the nearly-optimal base learners in our ensemble framework.

Base learner/ Ensemble strategy	exchange rate	elec	traf	solar
`DeepAR-T`	0:0075	0.0548	0.0879 (0:113)	0:3252
`DeepAR-G`	0:0067	0:0618	0:1140	0:3117
`DeepAR-P`	0:2261	0:0910	0:9828	0:3137
`Transformer`	0:0298	0:0266	0:0908	0:3584
`MQ-CNN`	0.0133	0:0544	1:8793 (0.166)	0:7735
`TFT`	0:0060	0:0844	0:1144	0:3253
`NBEATS`	0:0106	0:0480	0:2270	0:9983
Mean	0:0359	0:0490	0:2029	0:3790
Median	0:0090	**0.0489**	0:0905	0:3256
Global optimal	0:0124	0:0790	0:1991	0:3913
`Winner-takes-all`	0:0133	0:0548	**0.0879**	0:7735
`RL-hybrid` (Ours)	**0.0060**	0:0544	0:1141	**0.3058**

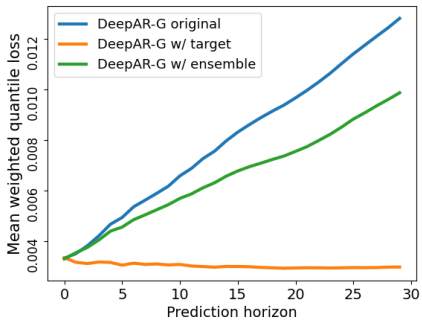
Table 1: Performance comparison on real-world benchmark datasets. The winning method among ensemble methods are made bold. The retangular is the one selected in `Winner-takes-all` ensemble method. The values in the parenthesis are the accuracy evaluated in the backtesting window.

5.2 INVESTIGATING PROPERTIES OF DYNAMIC ENSEMBLES

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.



(a) Learning time-varying ensemble weights.



(b) Boost performance of AR model.

Figure 3: The learned ensemble weights are consistent with the performances of the base learners. over the prediction horizon. QL and rank are averaged over all items in the dataset.

Property 2: Boosting the performance of auto-regressive (AR) 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 3b) 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 3b 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 4b (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 4b (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.

Property 3: Auto-regressive dynamic ensemble is more powerful than direct dynamic through 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`.

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 to compare auto-regressive vs direct dynamic.

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.

REFERENCES

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020a. URL http://jmlr.org/papers/v21/19-820.html.

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, 2020b.

John M Bates and Clive WJ Granger. The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468, 1969.

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.

Aadyot Bhatnagar, Paul Kassianik, Chenghao Liu, Tian Lan, Wenzhuo Yang, Rowan Cassius, Doyen Sahoo, Devansh Arpit, Sri Subramanian, Gerald Woo, Amrita Saha, Arun Kumar Jagota, Gokulakrishnan Gopalakrishnan, Manpreet Singh, K C Krithika, Sukumar Maddineni, Daeki Cho, Bo Zong, Yingbo Zhou, Caiming Xiong, Silvio Savarese, Steven Hoi, and Huan Wang. Merlion: A machine learning library for time series. 2021.

Gerda Claeskens, Jan R Magnus, Andrey L Vasnev, and Wendun Wang. The forecast combination puzzle: A simple theoretical explanation. *International Journal of Forecasting*, 32(3):754–762, 2016.

R Glen Donaldson and Mark Kamstra. Forecast combining with neural networks. *Journal of Forecasting*, 15(1):49–61, 1996.

Graham Elliott. Averaging and the optimal combination of forecasts. *University of California, San Diego*, 2011.

facebookresearch. Kats. https://github.com/facebookresearch/Kats, 2021.

Yuwei Fu, Di Wu, and Benoit Boulet. Reinforcement learning based dynamic model combination for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6639–6647, 2022.

Julia Gastinger, Sébastien Nicolas, Dušica Stepić, Mischa Schmidt, and Anett Schülke. A study on ensemble learning for time series forecasting and the need for meta-learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

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.

Xiaoyong Jin, Youngsuk Park, Danielle C. Maddix, Hao Wang, and Yuyang Wang. Domain adaptation for time series forecasting via attention sharing, 2022.

Kelvin Kan, François-Xavier Aubet, Tim Januschowski, Youngsuk Park, Konstantinos Benidis, Lars Ruthotto, and Jan Gasthaus. Multivariate quantile function forecaster. In *International Conference on Artificial Intelligence and Statistics*, pp. 10603–10621. PMLR, 2022.

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.

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.

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

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.

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.

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.

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.

Linbo Liu, Youngsuk Park, Trong Nghia Hoang, Hilaf Hasson, and Jun Huan. Towards robust multivariate time-series forecasting: Adversarial attacks and defense mechanisms. *arXiv preprint arXiv:2207.09572*, 2022.

Mohamed Massaoudi, Shady S Refaat, Ines Chihi, Mohamed Trabelsi, Fakhreddine S Oueslati, and Haitham Abu-Rub. A novel stacked generalization ensemble-based hybrid lgbm-xgb-mlp model for short-term load forecasting. *Energy*, 214:118874, 2021.

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.

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.

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, 2014a.

Yasuko Matsubara, Yasushi Sakurai, Willem G van Panhuis, and Christos Faloutsos. Funnel: automatic mining of spatially coevolving epidemics. In *KDD*, pp. 105–114. ACM, 2014b.

Jihoon Moon, Seungwon Jung, Jehyeok Rew, Seungmin Rho, and Eenjun Hwang. Combination of short-term load forecasting models based on a stacking ensemble approach. *Energy and Buildings*, 216:109921, 2020.

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.

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.

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.

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.

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.

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.

Amal Saadallah and Katharina Morik. Online ensemble aggregation using deep reinforcement learning for time series forecasting. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–8. IEEE, 2021.

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.

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.

Jeremy Smith and Kenneth F Wallis. A simple explanation of the forecast combination puzzle. *Oxford Bulletin of Economics and Statistics*, 71(3):331–355, 2009.

James H Stock and Mark W Watson. Combination forecasts of output growth in a seven-country data set. *Journal of forecasting*, 23(6):405–430, 2004.

Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

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.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

TaeHo Yoon, Youngsuk Park, Ernest K Ryu, and Yuyang Wang. Robust probabilistic time series forecasting. In *International Conference on Artificial Intelligence and Statistics*, pp. 1336–1358. PMLR, 2022.

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 Lillicrap et al. (2015) and some specific to dynamic AR ensemble. The final hyper-parameters used for different datasets for the experiment in Section 5.1 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; 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.

A.2.1 EXPERIMENTS IN TABLE 1

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.

A.3 IMPLEMENTAIONS OF TS-GYM

Error metric We evaluate the forecasting error in terms of the mean weighted quantile loss. See the precise definition in the appendix.

$$\frac{1}{q}\frac{\sum_{i=1;j=T+1;k=1}^{N;T+h;q}\max\{\ _k(z_{i;j}-\hat{z}_{i;j;k});(1-\ _k)(\hat{z}_{i;j;k}-z_{i;j})\}}{\sum_{i=1;j=T+1}^{N;T+h}|z_{i;j}|} \tag{7}$$

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

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
	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.

B BENCHMARK RESULT DISCUSSION

For the more detailed discussion, we can observe that the proposed `RL-hybrid` method outperforms all base models and baselines on all *exchange rate* and *solar* datasets. For *exchange rate*, which is a regular dataset with clear daily patterns, a single base learner usually performs very well. Our `RL-hybrid` method is able to identify the single best base learner (`TFT`). On the other hand, *exchange rate* is less regular and more challenging. Our `RL-hybrid` method is better (2%) than all base models and baselines considered. This is because our dynamic ensemble method are able to capture the time-varying patterns of the base learners' performance profile and boost the performance of the auto-regressive base learners (see Section 5.2 for more discussions).