An Actor-Critic Ensemble Aggregation Model for Time-Series Forecasting*

Amal Saadallah
Artificial intelligence Group
TU Dortmund
Germany
amal.saadallah@cs.tu-dortmund.de

Maryam Tavakol
Artificial intelligence Group
TU Dortmund
Germany
maryam.tavakol@cs.tu-dortmund.de

Katharina Morik
Artificial intelligence Group
TU Dortmund
Germany
katharina.morik@tu-dortmund.de

Abstract—**Ensemble models are widely used as an effective technique in time-series forecasting, and recently, are inclined toward leveraging meta-learning methods due to their proven predictive advantages in combining individual models in an ensemble. However, finding the optimal strategy for ensemble aggregation is an open research question, particularly, when the ensemble needs to be adapted in real-time. In this paper, we propose a novel meta-learning approach for aggregation of linearly weighted ensembles for the task of time-series forecasting. We outline a deep reinforcement learning framework with a coherent design of the components of the environment and the objective function as an aggregation method in our task. In this framework, the combination policy in ensembles is modeled as a sequential decision making process which is able to capture the temporal behavior in time-series, and an actor-critic model aims at learning the optimal weights in a continuous action space. An extensive empirical study on various real-world datasets demonstrates that our method achieves excellent or on par results in comparison to the state-of-the-art approaches as well as several baselines.**

Index Terms—**Ensemble learning , weighting , meta- learning , actor-critic networks , time-series forecasting.**

I. Introduction

An ensemble model is defined as a collection of several models that addresses the same learning task as the individual models to improve the overall performance [1]. The learning procedure of an ensemble consists of three main steps. First, a set of h possible hypotheses generate h base models, such that each model is able to individually provide an accurate approximation of an unknown function f. The second step carries out model pruning to keep only a subset $m < h$ of the hypotheses. The last step, which is the main focus of this paper, consists of combining these hypotheses together into one single model. Note that we use the term combination and aggregation interchangeably throughout the paper.

Over the past decades, several approaches have been developed for ensemble learning in general [2], [3], and for ensembles on time-series forecasting in particular [1], [4]–[6]. Many of these methods focus on optimizing one specific step in ensemble learning, amongst them, model combination is of the highest interest [1], [4], [5]. Combination strategies in an ensemble can be categorized into three main families: voting schemes which employ either the majority or (weighted) average of the votes (e.g., bagging [3]), cascading paradigm where the outputs of the base models are iteratively added to the training set one at a time, and stacking approaches [7] in which the combination of actual outputs is often learned from previous experiences by a meta-learning strategy. The latter learns a combination rule by incorporating a set of meta-features and/or performance-based landmarking features [1]. Hence, learning the optimal combination strategy in a linearly weighted ensemble, remains an open research question [8].

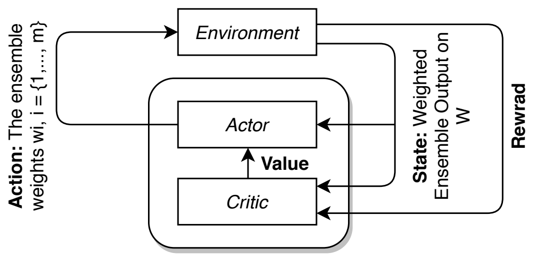

Environment
Actor
Value
Critic
Action: The ensemble weights wi, i = {1,..., m}
State: Weighted Ensemble Output on W
Rewrad

Fig. 1: Components of EA-DRL approach

In this paper, we propose a novel meta-learning technique for dynamic ensemble aggregation in the problem of time-series forecasting. In addition to the fact that time-series data-points are inherently time ordered, they may encompass occasional concept drifts, and the combination strategy of an ensemble requires to dynamically adapt to their changes. Therefore, we take a sequential approach based on Reinforcement Learning (RL) that is able to capture the temporal changes that occur in the data and provides the optimal combination strategy of the ensemble in real-time scenarios. Although reinforcement learning has successfully performed in many complex tasks [9], it has been insufficiently explored in the domain of time-series analysis as well as ensemble learning. To the best of our knowledge, we are the first to exploit reinforcement learning to learn the optimal combination strategy of ensembles in time-series forecasting tasks.

In our approach, we first create a set of heterogeneous base models that are trained in parallel and separately from each other to maximize diversity. Second, we learn a combination

[*] This work is supported by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 and the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01—S18038AB).

policy (optimal weights of the base models) using a meta-learning approach in a continuous space of weights that would lead to the most accurate ensemble construction given a finite window of previous time-series values. Our framework is addressed by EA-DRL: Ensemble Aggregation using Deep Reinforcement Learning, throughout the paper, and the overall procedure of EA-DRL is illustrated in Figure 1. In this work, we leverage an actor-critic architecture in deep learning settings to learn the combination policy of linearly weighted ensembles. We employ the approach presented in [10] to learn a policy in the continuous action space, in which the actions are determined to be the set of weights of the ensemble. We further conduct comprehensive empirical analysis to validate our framework using 20 real-world time-series datasets. The obtained results show that our method outperforms standard state-of-the-art methods for ensemble learning such as sliding-window averaged ensemble [4], stacking [7], and bagging [3], and performs better or on par with adaptive approaches for dynamic ensemble selection [6].

II. Ensemble Aggregation Model

This section presents EA-DRL, an approach based on Reinforcement Learning (RL) for ensemble aggregation in time-series forecasting. In this approach, first a pool of base models is constructed to perform the forecasting task, which is further used to learn an optimal aggregation policy in an RL framework. Note that the learning of both the base models and the optimal policy are performed offline, and we show how to leverage the obtained policy in online scenarios for forecasting future values of time-series.

A. The Ensemble Problem Setting

A univariate time-series X_t is a temporal sequence of values up to time t, $X_t = \{x_1, x_2, \ldots, x_t\}$, where x_i represents the value of X at time i. Let $\mathbb{M} = \{f_1, f_2, \ldots, f_m\}$ be a pool of m base models trained to approximate a true unknown function f that generated X_t. An ensemble model $\overline{f}$ of $\mathbb{M}$ at a future data-point $t+j$ ($j \geq 1$) can be formally expressed as a convex combination of predictions of the individual base models in $\mathbb{M}$

$$\overline{f}(x_{t+j}) = \sum_{i=1}^{m} w_i^{t+j} f_i(x_{t+j}), \tag{1}$$

where $w_i^{t+j}, i \in \{1, \ldots, m\}$, determine the weights of the ensemble for the time $t+j$. For notational simplicity, assume in the following that our objective is to predict x_{t+1} (i.e., $j = 1$) without loss of generality. We thus aim to find a set of weights for the ensemble that minimizes the expected prediction error for the next forecast

$$\begin{aligned} &\underset{w_i^{t+1}, \forall i}{\operatorname{argmin}} \quad \mathbb{E}\big[\big(f(x_{t+1}) - \overline{f}(x_{t+1})\big)^2 \,|X_t\big], \\ &\text{s.t.} \quad w_i^{t+1} \geq 0, \forall i \in \{1, \cdots, m\}, \sum_{i=1}^{m} w_i^{t+1} = 1 \end{aligned} \tag{2}$$

The above objective function indicates that the weights are time-related, since they require to be adapted at every time-step in order to comply with the temporal properties of the underlying process. More specifically, time-series are inherently time ordered, and the forecasting task depends on a sequential analysis that is able to capture the temporal changes that occur in the data. Therefore, we take a sequential approach based on RL to learn an optimal policy function that automatically provides the optimal set of weights in real-time scenarios given the situations at different time-steps.

B. The MDP Framework

An RL problem is mathematically formulated via a Markov Decision Process (MDP) [22]. An MDP is defined by a five tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, in which $\mathcal{S}$ are the states, $\mathcal{A}$ the actions, $\mathcal{R} : \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function, and $\gamma \in [0, 1)$ is the discount factor. The goal of an MDP is to learn a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the total obtained reward, and decides about what actions to take at every state. In this section, we characterize our meta-learning task for finding the optimal weights of ensembles for time-series forecasting in an MDP framework.

Actions. An action is interpreted as a decision made at time t to be executed at $t + 1$. In our setting, a decision at every time-step is to determine the set of weights for the ensemble that optimizes the objective in Equation 2. Therefore, the action $\boldsymbol{a}_t \in \mathcal{A}$ is defined as an m-dimensional vector, $\boldsymbol{a}_t = (a_{t,1}, a_{t,2}, \ldots, a_{t,m})^\top$, taken at time t that corresponds to the vector of ensemble weights $\boldsymbol{w}_{t+1} = (w_1^{t+1}, w_2^{t+1}, \ldots, w_m^{t+1})^\top$. These weights are attributed to each of the m single models in the ensemble to predict the future value of x_{t+1}. This definition of actions leads to a continuous m-dimensional action space.

States. Since we are dealing with time-evolving data where the performance of the base models and the ensemble rely on that, the state $s_t \in \mathcal{S}$ is set to reflect both the dynamics of the time-series and the effect of performed actions until time t. Let ω be a provided window size of a validation set X^ω that corresponds to the previous ω values of the time-series until time t, $X^\omega = \{x_{t-\omega+1}, x_{t-\omega+2}, \ldots, x_t\}$. We consider state s_t to be the current window of time-series that is used for forecasting the next value, i.e., x_{t+1}. However, the next state $s_{t+1} \in \mathcal{S}$ should be devised in a way that it reflects the result of a taken action $\boldsymbol{a}_t$ at state s_t. Therefore, we take the output of the ensemble $\overline{f}$ on X^ω as a state, instead of X^ω itself, since it reflects the result of the internal combination policy (i.e., $\boldsymbol{a}_t$), and also, captures the characteristic traits of X^ω. That means, $s_t = \{\hat{x}_{t-\omega+1}, \hat{x}_{t-\omega+2}, \ldots, \hat{x}_t\}$, where $\hat{x}_i$ is the output of ensemble at time t given by Equation (1) and using the weight vector $\boldsymbol{a}_t$, $\hat{x}_i = \bar{f}(x_i)$.

Transition function. The transition function is deterministic in our problem as selecting an action only leads to one possible next state: $\mathcal{P}(s_{t+1}|s_t, \boldsymbol{a}_t) = 1$ when $s_{t+1} = \{\hat{x}_{t-\omega+2}, \hat{x}_{t-\omega+3}, \ldots, \hat{x}_{t+1}\}$ and is zero otherwise.

Reward function. We further determine the reward of taking an action $\boldsymbol{a}_t$ in state s_t as a function $\mathcal{R}(s_t, \boldsymbol{a}_t)$, which for brevity in notation we denote by r_t. In our framework, we benefit from an ensemble accuracy-related measure since we are searching for a combination strategy that would result in the most accurate ensemble model. One alternative is to use a direct estimation of the ensemble performance using, for instance, an inverse measure to the ensemble forecasting error. Nevertheless, the magnitude of this error does not only depend on its performance or on the performance of the single models, but also relates to the time-varying structure of the time-series itself. This can result in slower convergence rate of the RL algorithm. Therefore, to stabilize the reward, we opt for a rank-based definition instead.

Assume that the set of m models in $\mathbb{M}$ together with the ensemble model form a total of $m+1$ models. Subsequently, we compile a ranked list of all the models (including the ensemble) using their corresponding forecasting error, in which, ρ^{f_i} indicates the rank of model f_i. The lower the rank is in the obtained ranking, the more accurate the model is (i.e., rank 1 means the model performs the best). The reward r_t is thus defined by

$$r_t = m + 1 - \rho^{\overline{f}}, \tag{3}$$

where $\rho^{\overline{f}}$ is the rank of the ensemble among all the models, and the lower it is, the more accurate the ensemble is and the higher the reward value will be. An empirical evaluation of the importance of the choice of the reward is illustrated in the experiment section.

C. Learning the Combination Policy

Once the meta-learning task is phrased in an MDP framework, the policy π is learned in favor of maximizing the reward which correlates to an inverse measure of the ensemble performance. Consequently, the objective function presented in Equation 2 is turned into learning the optimal policy of the MDP via an RL algorithm. We employ the deep actor-critic approach presented in [10] to learn an optimal combination policy in a continuous action space. This approach is selected since it is well-suited for both continuous and high-dimensional action and state spaces (for large m and ω). In this architecture, the actor is accountable for selecting an action given the current state, and the critic estimates a value function which provides adequate evaluation for the actor. Both parts are represented by (deep) neural networks that can be optimized by gradient descent-based methods. As a result, the actor and the critic networks are called the *policy network* and the *value network*, respectively. The value network predicts the value of an action $\boldsymbol{a}_t$ in state s_t via $Q(s_t, \boldsymbol{a}_t|\phi)$, where ϕ is the parameter vector of the value network; see [10]: Eq. (3)-(5). On the other hand, the policy network learns a policy $\pi(s_t|\theta)$ which yields a deterministic policy in state s_t given the network parameters θ; see [10]: Eq. (6). During the learning, the actor takes the gradients derived from the policy gradient theorem and adjusts the policy parameters θ, and the critic network estimates the approximate value function for the current policy π via Bellman equation.

Algorithm 1 Forecasting next N_f values

```
Require: validation set $X^\omega$; policy $\pi(s|\theta)$ ; window size: $\omega$
1: set $s$ to $\{\hat{x}^E_{t-\omega+1}, \hat{x}^E_{t-\omega+2}, \cdots, \hat{x}^E_t\}$
2: predict $\boldsymbol{w}$ using $\pi(s|\theta)$
3: predict $x_{t+1}$ using Equation 1
4: for $j \in \{2, \cdots, N_f\}$ do
5:     update $s$ by removing oldest value and adding $x_{t+j-1}$
6:     predict $\boldsymbol{w}$ using $\pi(s|\theta)$
7:     predict $x_{t+j}$ using Equation 1
```

D. Improving Convergence

A replay buffer R is initialized to store transitions $T^k = (s^k_t, \boldsymbol{a}^k_t, r^k_t, s^{k+1}_t), k \in \{1, \cdots, N_{max}\}$ with N_{max} the maximum number of stored transitions . At each iteration i within each *episode*, N transitions are randomly sampled from R in [10]. Instead of adopting random sampling, we suggest to induce a diversity sampling by showing equal number of transitions with actions resulting in high reward values and transitions with actions resulting in low reward values. In this way, both types of actions resulting in high and low rewards are fed to both critic and actor networks. This is achieved by sampling :

$$\begin{cases} N/2 \text{ transitions with reward } r^k_t \geq \underset{k \in \{1,\cdots,N_{max}\}}{\text{median}} (r^k_t) \\ N/2 \text{ transitions with reward } r^k_t < \underset{k \in \{1,\cdots,N_{max}\}}{\text{median}} (r^k_t) \end{cases} \tag{4}$$

E. Online Forecasting

After the policy network $\pi(s|\theta)$ is learned, we apply the model for predicting the weights of the ensemble (i.e., actions) that will be used for predicting the future values of time-series in an online manner. Let state s be X^ω, the predicted weights via a are used to predict x_{t+1}. Afterwards, the ω-length vector of time-series X^ω (i.e., the state s) is moving forward by one value. That means, the oldest value is removed and the predicted value $\hat{x}^E_{t+1}$ is added to the current window. The new sate s' and $\pi(s|\theta)$ are employed to predict the weights of the ensemble to forecast the next value of the time-series. The procedure is repeated until N_f desired values of the time-series are forecasted. This stage is summarized in Algorithm 1.

III. Empirical Study

In this section, we present the experiments that evaluate the performance of EA-DRL for forecasting to answer the following research questions. **Q1:** How does EA-DRL perform compared to the state-of-the-art and existing dynamic ensemble combination approaches for time-series forecasting?; **Q2:** How critical the reward function is for the convergence of the reinforcement learning approach?; **Q3:** What is the impact of improving the convergence compared to the learning procedure proposed in [10]?; **Q4:** How scalable is EA-DRL in terms of computational resources compared to the most successful ensemble approaches for forecasting?

TABLE I: List of Datasets used for the experiments.

dataset-ID	Time-series	Data source	Data characteristics
1	Water consumption	Oporto city [5]	Daily– Jan. 2012 to Oct. 2016
2	Humidity	Bike sharing [5]	Hourly–Jan. 1, 2011 to Mar. 01, 2011
3	Windspeed		
4	Total bike rentals		
5	Vatnsdalsa	River flow [5]	Daily–Jan. 1, 1972 to Dec. 31, 1974
6	Total cloud cover	Weather data [23]	Hourly–Apr. 25, 2016 to Aug. 25, 2016
7	Precipitation		
8	Global horizontal radiation	Solar radiation monitoring [5]	Hourly–Feb. 16, 2016 to May 5, 2016
9	Taxi Demand 1	Porto Taxi Data [4]	Half-hourly (taxi pick-ups)–Jul. 01, 2013 to Jun. 30, 2014
10	Taxi Demand 2		
11	NH4 concentration	NH4 in wastewater [24]	10-minute steps–Nov. 30, 2010 at 16:10 to Jan. 01, 2011 at 6:40
12	Humidity RH_3	Appliances Energy [24]	10-minute steps– Jan. 11, 2016 at 17:00 to May 27, 2016 at 18:00
13	Humidity RH_4		
14	Humidity RH_5		
15	Temperature T_{out}		
16	Wind speed		
17	Tdewpoint		
18	France CAC	European stock indices [24]	10-minute steps–Jan. 11, 2016 at 17:00 to May 27, 2016 at 18:00
19	Germany DAX (Ibis)		
20	Switzerland SMI		

Experimental Setup We conduct our experiments on 20 real-world time-series data from 9 different domains, which are briefly described in Table I. Each dataset is further split into training and testing sets via a $75\% - 25\%$ ratio. We evaluate various methods in terms of the root mean squared error (RMSE), and the error is used to create a ranked list of models according to their performance that serves as the evaluation metric in our analysis. The results are further assessed using the Bayesian correlated t-test to compare pairs of models in a single dataset, and the Bayes sign test to compare pairs of methods across multiple datasets [25]. Moreover, an embedding dimension of $k = 5$ is used for all the time-series.

Single base models set-up Additionally, we construct a pool $\mathbb{M}$ of single base models for the ensemble learning to incorporate diverse families of models in the ensemble. We mentioned earlier that there is no single method for forecasting that outperforms all the other methods on every time-series. Hence, we incorporate and test different families of models: **ARIMA:**Autoregressive Integrated Moving Average [26], **ETS:** Exponential Smoothing [27], **GBM:**Gradient Boosting Machines [33], **GP:** Gaussian Processes [34], **SVR:**Support Vector Regression [35],**RFR:** Random Forest [3], **PPR:**Projection Pursuit Regression [36], **MARS:** MARS [37], **PCMR:** Principal Component Regression [38], **DT:**Decision Tree Regression [1], **PLS:** Partial Least Squares Regression [38], **MLP:** Multilayer Perceptron [39], **LSTM:**Long short-term memory network [29], **Bi-LSTM:**Bidirectional LSTM [40], **CNN-LSTM:** CNN-based LSTM [30] and **Conv-LSTM:**Convolutional LSTM [32]. Regression models are also included in $\mathbb{M}$ and are applied after using time series embedding to dimension k. Using different parameter settings for each approach, we generate a pool of 43 single base models that will be used for constructing the ensemble model.

EA-DRL set-up In EA-DRL setting, both policy and value networks are based on MLPs, which perform simple regression and multi-regression (m-dimensional weights), respectively. In addition, a standard normalization is applied to the output of the policy network, so that all the weights are positive and sum to one. The hyperparameters of EA-DRL are tuned by model selection which result in discount factor $\gamma = 0.9$, learning rate $\alpha = 0.01$, and $max.ep$ and $max.iter$ of 100.

State-of-the-art Methods We compare the performance of EA-DRL against several standard baselines as well as state-of-the-art approaches that are briefly described in the following: **ARIMA** [26], **LSTM** [29], **StLSTM** [41]: Stacked LSTM model where multiple hidden LSTM layers are stacked one on top of another. This model can be viewed as an ensemble of LSTMs combined using a cascading approach, **RF** [3]: random forest, **GBM** [33]: gradient boosting machine, **SE** [42]: A static ensemble model that averages the performance of all base learners using arithmetic mean, **SWE** [4]: A linear combination of predictions of the base models, in which the weights are based upon recent performance over a time sliding-window, **EWA** [16]: An ensemble combination using exponential weighted averages,**FS** [16]: The fixed share approach which is designed for tracking the best expert across a time-series, **OGD** [16]: An approach based on online gradient descent that provides theoretical loss bound guarantees, **MLPOL** [16]: A polynomially weighted average forecast combination, **Stacking** [7]: An ensemble approach using random forest as a meta-learner, **Clus** [6]: A meta-learner based on dynamic clustering method to group similar models together, and only cluster representatives are selected to form the ensemble using **SWE**, **Top.sel** [6]: A dynamic method to select the best performing base model and combining them using **SWE**, **DEMSC** [6]: A drift-aware combination of **Top.sel** for ensemble pruning and **Clus** for diversity enhancement to construct the **SWE**-based ensemble model.

A. Evaluation Results

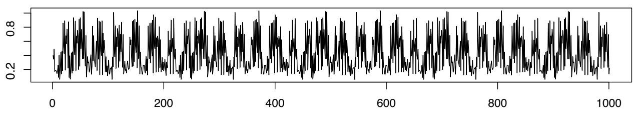
On predictive performance

Table II presents the pairwise comparisons between EA-DRL and the baseline approaches using the Bayesian correlated t-test. It exhibits number of wins and losses of EA-DRL compared to the other methods in the table. The numbers in parenthesis represent significant wins/losses with probability above 95%. In addition,we evaluate the distribution of ranks across different time-series for all the methods (the lower the better). A rank of one means that the model is the best performing on all datasets. The results show that our approach achieves the best performance among the evaluated methods. Furthermore, EA-DRL outperforms the baseline methods in terms of wins/loses in pairwise comparison, however not DEMSC and MLPOL. The approaches that are based on combining individual forecasters, e.g., SE, SWE, etc., and common ensemble methods, such as RF, GBM, Stacked LSTM and Stacking, show inferior performance compared to EA-DRL. ARIMA and LSTM, state-of-the-art methods for forecasting, have a considerable difference in the average rank as well. The two competitive approaches to our method are DEMSC and MLPOL that perform well in the pairwise comparison, nevertheless, both attain a higher average rank. DEMSC is based on real-time update of meta-learning strategy behind, while EA-DRL is devised offline and only predictions are computed in real time. More details are provided when discussing the computational efficiency.

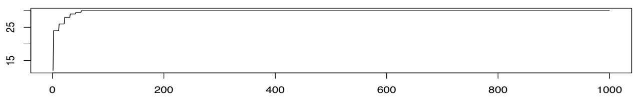
TABLE II: Pairwise comparison between EA-DRL and baseline methods averaged over all 20 datasets ($\omega = 10$).

Method	Pairwise Looses	comparison Wins	Avg. Rank
ARIMA	8(0)	11(6)	7.74 ± 4.0
RF	2(2)	17(15)	10.21 ± 3.0
GBM	0(0)	19(18)	14.11 ± 2.6
LSTM	4(0)	16(12)	8.42 ± 4.7
StLSTM	1(0)	19(18)	13.16 ± 2.4
SE	5(0)	15(14)	7.80 ± 3.8
SWE	4(2)	16(15)	10.37 ± 4.35
EWA	5(1)	15(10)	6.24 ± 2.3
FS	6(4)	14(9)	6.90 ± 3.0
OGD	5(2)	15(10)	6.28 ± 2.3
MLPOL	9(1)	11(6)	5.37 ± 3.7
Stacking	2(0)	18(16)	12.36 ± 2.5
Clus	4(0)	16(15)	12.05 ± 5.0
Top.sel	4(2)	16(11)	7.68 ± 4.2
DEMSC	11(0)	9(3)	4.53 ± 3.7
EA-DRL	-	-	$\mathbf{2.89 \pm 1.9}$

Reward Setting In the next experiment, we study the convergence of deep actor-critic algorithm in [10] using two different settings for the reward function. Figure 2a shows the results for the reward defined as 1−NRMSE, where NRMSE is the normalized RMSE of the computed ensemble using the corresponding action (i.e., weights) on X^{ω}, while Figure 2b uses the reward as defined in Equation 3. The same setup of EA-DRL is applied for the second approach using the NRMSE as reward. As it is mentioned in Section II-B, Algorithm 1 in [10] does not converge using the first definition of the reward, since the magnitude of forecasting errors do not only depend on the models but is also changing with time. Thus, the choice of the reward is critical for the convergence of the reinforcement learning strategy. This answers the research question **Q2**.



(a) Reward computed using $1 - NRMSE$



(b) Reward computed using *Eq. 3*

Fig. 2: Learning curves of Alg.1 [10] with two different reward definitions. In the x-axis, the number of episodes. In the y-axis, the average reward over each episode.

On improving the convergence we compare the runtime of EA-DRL using transitions sampling procedure as explained in Section II-D to the training procedure based on random sampling of transitions [10]. Our sampling reduces the number of required episodes for convergence to 100 episode while more than 250 episodes were required using random sampling of transitions. In average, the execution time for learning the policy using ou sampling procedure in the offline phase is around $300\,\text{min}$ while in the latter one is around $735\,\text{min}$, highlighting thus the usefulness of improving the convergence in optimizing computational resources even for the offline learning phase. This answers the research question **Q3**.

TABLE III: Empirical runtime comparison between EA-DRLand DEMSC.

Method	Avg. Runtime in sec.
EA-DRL	37.93 ± 10.83
DEMSC	67.97 ± 27.4

Execution time Last but not least, we compare the runtime of EA-DRL against the most competitive state-of-the-art method, i.e., DEMSC, where the results are summarized in Table III. Note that the policy network is trained offline and the reported runtime is calculated for computing the time-series predictions in real-time using Algorithm 1. DEMSC relies on informed update (i.e., following a drift detection mechanism) of the strategy for model combination. The reported runtime for DEMSC concerns also only the online predictions computation and any operation computed offline is not taken into account to ensure a fair comparison. The results demonstrate that EA-DRL has a lower average time. Consequently, in addition to the performance results, EA-DRL performs better in terms of runtime, even though it does not update the policy in real-time, compared to the most competitive method. This answers the research question **Q4**.

B. Discussion and Future Work

The empirical results indicate that EA-DRL has performance advantages compared to other ensemble methods and is competitive with the most recent state-of-the-art approaches for dynamically combining forecasting methods. However, compared to DESMC, which needs to select the top-performing base models, cluster them, compute clusters representatives to take part in a weighted ensemble, and update the whole process online once a drift in the dependencies between single models is detected, our method is computationally cheaper and the policy is learned offline and only deployed online. One potential future research direction would be to investigate the impact of an online update of the policy, for instance in a periodic manner, or in an informed fashion following a drift-detection mechanism in the data and/or the performance of the ensemble. We can additionally incorporate a pruning step into our framework, so that only relevant models take part in the weighting/combination stage. Furthermore, ensemble diversity is known to be one of the most important aspects for successful ensemble construction. Such aspect can be further examined in our framework by adding a diversity-related measure in the formulation of the reward. Moreover, this paper is focused on forecasting problems. Notwithstanding, our intuition is that the central idea behind EA-DRL can be generalized to other i.i.d. domains, e.g., standard regression and classification tasks. We will also explore these extensions in the future work.

IV. Conclusions

In this paper, we introduced EA-DRL: a novel and practically effective ensemble aggregation framework for time-series forecasting that employs a deep reinforcement learning approach as a meta-learning technique. We exploited an actor-critic algorithm, in which the actor is trained to gain experience on how to select the best set of weights given a previous combination of models in the ensemble. Once the optimal policy is learned offline, the ensemble weights are predicted in real-time using the policy network and ensemble predictions are then computed. An extensive empirical evaluation of EA-DRL on twenty real-world datasets demonstrated superb results compared to multiple baseline algorithms, and achieved competitive performance with the state-of-the-art.

References

[1] J. Khiari, L. Moreira-Matias, A. Shaker, B. Ženko, and S. Džeroski, "Metabags: Bagged meta-decision trees for regression," in *Joint European Conference on Machine Learning & Knowledge Discovery in Databases*. Springer, 2018, pp. 637–652.

[2] T. G. Dietterich *et al.*, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.

[3] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[4] A. Saadallah, L. Moreira-Matias, R. Sousa, J. Khiari, E. Jenelius, and J. Gama, "Bright-drift-aware demand predictions for taxi networks," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[5] V. Cerqueira, L. Torgo, F. Pinto, and C. Soares, "Arbitrated ensemble for time series forecasting," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2017, pp. 478–494.

[6] A. Saadallah, F. Priebe, and K. Morik, "A drift-based dynamic ensemble members selection using clustering for time series forecasting," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2019.

[7] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[8] V. Cerqueira, F. Pinto, L. Torgo, C. Soares, and N. Moniz, "Constructive aggregation and its application to forecasting with dynamic ensembles," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 620–636.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[11] J. G. D. Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, vol. 22, no. 3, pp. 443–473, 2006, twenty five years of forecasting.

[12] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[13] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proceedings of the 20th international conference on machine learning (icml-03)*, 2003, pp. 928–936.

[14] P. Gaillard and Y. Goude, "Forecasting electricity consumption by aggregating experts; how to design a good set of experts," in *Modeling and stochastic learning for forecasting in high dimensions*. Springer, 2015, pp. 95–115.

[15] L. Todorovski and S. Džeroski, "Combining classifiers with meta decision trees," *Machine learning*, vol. 50, no. 3, pp. 223–249, 2003.

[16] P. Gaillard and Y. Goude, *opera: Online Prediction by Expert Aggregation*, 2016, r package version 1.0. [Online]. Available: https://CRAN.R-project.org/package=opera

[17] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org, 2017, pp. 1126–1135.

[18] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.

[19] I. Partalas, G. Tsoumakas, I. Katakis, and I. Vlahavas, "Ensemble pruning using reinforcement learning," in *Hellenic Conference on Artificial Intelligence*. Springer, 2006, pp. 301–310.

[20] I. Partalas, G. Tsoumakas, and I. Vlahavas, "Pruning an ensemble of classifiers via reinforcement learning," *Neurocomputing*, vol. 72, no. 7-9, pp. 1900–1909, 2009.

[21] C. Feng and J. Zhang, "Reinforcement learning based dynamic model selection for short-term load forecasting," in *2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2019, pp. 1–5.

[22] R. S. Sutton and A. G. Barto, "Reinforcement learning," 1998.

[23] T. Stoffel and A. Andreas, "Nrel solar radiation research laboratory (srrl): Baseline measurement system (bms); golden, colorado (data)," 7 1981.

[24] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[25] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2653–2688, 2017.

[26] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[27] G. Jain and B. Mallick, "A study of time series models arima and ets," *Available at SSRN 2898968*, 2017.

[28] R. J. Hyndman, Y. Khandakar *et al.*, *Automatic time series for forecasting: the forecast package for R*. Monash University, Department of Econometrics and Business Statistics ..., 2007, no. 6/07.

[29] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying lstm to time series predictable through time-window approaches," in *Neural Nets WIRN Vietri-01*. Springer, 2002, pp. 193–200.

[30] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using cnn-lstm neural networks," *Energy*, vol. 182, pp. 72–81, 2019.

[31] I. E. Livieris, E. Pintelas, and P. Pintelas, "A cnn–lstm model for gold price time-series forecasting," *Neural Computing and Applications*, pp. 1–10, 2020.

[32] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[33] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[34] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006, vol. 2, no. 3.

[35] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, 1997, pp. 155–161.

[36] J. H. Friedman and W. Stuetzle, "Projection pursuit regression," *Journal of the American statistical Association*, vol. 76, no. 376, pp. 817–823, 1981.

[37] J. H. Friedman *et al.*, "Multivariate adaptive regression splines," *The annals of statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[38] B.-H. Mevik, R. Wehrens, and K. H. Liland, *pls: Partial Least Squares and Principal Component Regression*, 2018. [Online]. Available: https://CRAN.R-project.org/package=pls

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[40] Q. Sun, M. V. Jankovic, L. Bally, and S. G. Mougiakakou, "Predicting blood glucose with an lstm and bi-lstm based deep neural network," in *2018 14th Symposium on Neural Networks and Applications (NEUREL)*. IEEE, 2018, pp. 1–5.

[41] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[42] R. T. Clemen and R. L. Winkler, "Combining economic forecasts," *Journal of Business & Economic Statistics*, vol. 4, no. 1, pp. 39–46, 1986.

[43] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.