Machine Learning (2022) 111:3459–3487
https://doi.org/10.1007/s10994-022-06218-4

Explainable online ensemble of deep neural network pruning for time series forecasting

Amal Saadallah[1] · **Matthias Jakobs**[1] · **Katharina Morik**[1]

Received: 10 December 2021 / Revised: 13 May 2022 / Accepted: 2 July 2022 /
Published online: 2 August 2022

Abstract
Both the complex and evolving nature of time series data make forecasting among one of the most challenging tasks in machine learning. Typical methods for forecasting are designed to model time-evolving dependencies between data observations. However, it is generally accepted that none of them are universally valid for every application. Therefore, methods for learning heterogeneous ensembles by combining a diverse set of forecasters together appears as a promising solution to tackle this task. While several approaches in the context of time series forecasting have focused on how to combine individual models in an ensemble, ranging from simple and enhanced averaging tactics to applying meta-learning methods, few works have tackled the task of ensemble pruning, i.e. individual model selection to take part in the ensemble. In addition, in classical ML literature, ensemble pruning techniques are mostly restricted to operate in a static manner. To deal with changes in the relative performance of models as well as changes in the data distribution, we employ gradient-based saliency maps for online ensemble pruning of deep neural networks. This method consists of generating individual models' performance saliency maps that are subsequently used to prune the ensemble by taking into account both aspects of accuracy and diversity. In addition, the saliency maps can be exploited to provide suitable explanations for the reason behind selecting specific models to construct an ensemble that plays the role of a forecaster at a certain time interval or instant. An extensive empirical study on many 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. Our code is available on Github (https://github.com/MatthiasJakobs/os-pgsm/tree/ecml_journal_2022).

Keywords Ensemble pruning · Deep learning · Online learning · Time series forecasting · Concept drift · Explainability · Saliency maps

Editor: Krzysztof Dembczynski, Emilie Devijver.

Amal Saadallah and Matthias Jakobs have contributed equally to this work.

✉ Amal Saadallah
amal.saadallah@cs.tu-dortmund.de

✉ Matthias Jakobs
matthias.jakobs@tu-dortmund.de

[1] Artificial Intelligence Group, TU Dortmund, Dortmund, Germany

1 Introduction

Time series forecasting has attracted the attention of both academic and industrial communities as it has always been considered one of the principal steps for real-time decision-making and planning in a wide range of applications such as traffic prediction, weather forecasts, and stock market prices prediction (Krawczyk et al., 2017), to name but a few. However, forecasting is also considered one of the most challenging tasks in time series analysis due to the dynamic behavior of time series data, which may involve non-stationary and complex processes (Krawczyk et al., 2017; Saadallah et al., 2018; Cerqueira et al., 2017). Several Machine Learning (ML) models have been successfully applied to solve time series forecasting either by considering as input the whole or some window of an ordered sequence of time dependent observations in an offline or a streaming fashion, or by using time series embedding into a L-dimensional euclidean space to reformulate forecasting as a regression task (Saadallah et al., 2019; Cerqueira et al., 2017). Nevertheless, it is generally accepted that no ML model is universally valid for every forecasting application. This seems to be a particular case of the No Free Lunch theorem by Wolpert (1996), which means no learning algorithm is best suited for all tasks. More recently, Deep Artificial Neural Networks (DNNs) have shown some improvements over previous shallow ANN architectures (Romeu et al., 2013) and have been successfully applied to solve the forecasting task (Romeu et al., 2013; Taieb et al., 2012). This success is mainly explained by their ability to automatically learn new, complex and enriched feature representations from input data (Utgoff & Stracuzzi, 2002). Recurrent-based NNs such as Long Short-Term Memory Networks (LSTMs) (Gers et al., 2002), as well as Convolutional Neural Networks (CNNs) (Livieris et al., 2020), have been widely used as state-of-the-art NN methods in the context of forecasting (Romeu et al., 2013; Gamboa, 2017). However, deep learning models typically have high variances and low biases (Gamboa, 2017). In a complicated and dynamic application environment, it is difficult for an individual deep learning model to maintain high forecasting accuracy and robustness (Zhang et al., 2021).

In this context, one reasonable solution is to combine forecasts of different models in order to obtain one single desired forecast value. This is formally broached in the ML literature by ensemble learning. Ensemble methods (Saadallah et al., 2018; Krawczyk et al., 2017; Cerqueira et al., 2017; Brown et al., 2005; Breiman, 1996; Li et al., 2012; Tsoumakas et al., 2009) have been a very popular research topic during the last decades. The success of ensemble methods stems in part from the fact that they offer attractive solutions to a wide variety of learning problems from the past and the present, such as improving predictive performance (Li et al., 2012; Saadallah et al., 2018), learning from multiple physically distributed data sources (Stolpe et al., 2016), scaling inductive algorithms to large databases (Street & Kim, 2001) and learning from concept-drifting data streams (Saadallah et al., 2018, 2019). Ensemble construction can be divided into three main stages: (i) single model generation, where N possible hypotheses are formulated to model a given learning task. This results in N different individual models called base models; (ii) ensemble pruning, where only a subset of $k < N$ hypotheses is kept. This stage is devised to reduce the ensemble size and corresponding resources consumption while promoting its performance and (iii) model aggregation, where these hypotheses are aggregated together into one single model using some combination rule, e.g. majority voting or averaging.

Most existing methods for ensemble learning for time series data are focused around optimizing the last stage, i.e. model aggregation (Saadallah et al., 2018; Cerqueira et al., 2017, 2017, 2018; Saadallah et al., 2021; Saadallah & Morik, 2021). Even though several

empirical and theoretical studies have proven the importance of ensemble pruning for enhancing both computational efficiency and predictive performance for classification task (Li et al., 2012; Tsoumakas et al., 2009; Zhang et al., 2006; Martinez-Munoz et al., 2008; Caruana et al., 2004; Yu et al., 2011), only few works investigated this task for forecasting (Saadallah et al., 2019; Krikunov & Kovalchuk, 2015; Ma et al., 2015). This can be explained by two main facts. The first is due to the difficulty of the problem itself, especially when combined with time series. Hence, given a set of trained forecasters, selecting the sub-ensemble with the best generalization performance is difficult since it is not easy to estimate the generalization error of the sub-ensemble. In addition, finding the optimal subset of models is a combinatorial search problem with exponential computational complexity. Thus, it is infeasible to compute the exact solution by exhaustive search and approximate search is required (Li et al., 2012; Tsoumakas et al., 2009). This is even more computationally expensive for time series since the selected sub-ensemble have to cope with the time-evolving nature of data and therefore update and adaption of the pruning are required in real-time. The second fact consists of the difficulty of transferring pruning techniques proposed for classification to forecasting since adaptions have to be made in order to comply to the characteristics of time series data (Ma et al., 2015).

In this work, we propose a two-staged online pruning procedure for an ensemble of DNNs for time series forecasting. Since several studies showed that the performance of the ensemble depends largely on the performance of its base models and on the *diversity* amongst them (Li et al., 2012; Tsoumakas et al., 2009; Yu et al., 2011; Saadallah et al., 2019), our pruning procedure is devised to take into account these two aspects. This is achieved by comparing the computed so-called Regions of Competence (RoCs) of the DNNs to each other and to the most recent time series sequence pattern. The RoCs are computed using saliency maps, derived similarly to the way presented in (Saadallah et al., 2021) by establishing a mapping between the input time series and the performance of the DNN. On the one hand, *diversity* is encouraged using clustering of the corresponding DNNs RoCs, thus promoting the selection of diverse patterns by considering only cluster representatives. On the other hand, the second selection stage takes into account the accuracy of the cluster representatives by considering their closeness to the most recently acquired time series pattern. The distance between the current pattern and the most distant RoC from the selected models in the second stage sets its boundary under a form of a diameter. By setting up the number of clusters from the first stage, a bound for this diameter is derived, which automatically determines the final number of models that should be selected so that the expected ensemble error over the most recent time series window is reduced. At test time, we produce forecasts step by step. At the first time step, we perform the pruning like explained above. Afterwards, at each time step, we compute the minimum distance between the RoCs of the base models and the current time series pattern, i.e. the most recently observed window of time series observations. We keep monitoring the difference between this minimum distance and the initial computed minimum. If the difference diverges significantly over time, a drift in the dependency structure between the base models and the target time series is assumed to take place and the ensemble pruning is updated by recomputing the models' selection using the most recently acquired time series pattern following the same methodology. The difference deviation is tested incrementally using the well-known Hoeffding Bound (Hoeffding, 1994). Another drift-detection mechanism is employed to test the presence of drifts in the time series values. The occurrence of such drift triggers the update of both RoCs and the pruning strategy.

Since the base model selection is based on the pre-computed saliency maps-based RoCs, suitable explanations of both the selection and the performance of a specific

ensemble construction are provided. We further conduct comprehensive empirical analysis to validate our framework using 100 real-world time series datasets from various domains. The obtained results demonstrate that our method achieves excellent results in comparison to the the state-of-the-art approaches for ensemble pruning and learning as well as several baselines for time series forecasting. We note that all the experiments are fully reproducible, and both code and datasets are publicly available.[1] The main contributions of this paper are thus summarized as follows.

- We present a novel method for online ensemble pruning of CNNs for time series forecasting using pre-computed saliency maps-based RoCs.
- We derive a theoretical bound for the number of final models to be selected and show how they should be selected in both pruning stages in order to enhance both ensemble accuracy and diversity.
- We update the pruning in an informed manner following concept drift detection in the candidate CNNs' performance and the time series.
- We exploit the saliency maps to provide suitable explanations for the reason behind constructing a specific ensemble to play the role of forecaster at a certain time interval or instant.
- We provide a comparative empirical study with state-of-the-art methods, and discuss their implications in terms of predictive performance and scalability.

In the remainder of this paper, we describe the proposed approach in Sect. 3, after discussing related work in Sect. 2. In Sect. 4, we present an exhaustive experimental evaluation of its efficiency and scalability in practice. Finally, the last section concludes the paper.

2 Literature review

Ensemble pruning has been widely studied in the literature for classification problems (Li et al., 2012; Tsoumakas et al., 2009; Zhang et al., 2006; Martinez-Munoz et al., 2008; Caruana et al., 2004; Yu et al., 2011). Few works tackled this issue for time series forecasting, especially in an online dynamic fashion (Saadallah et al., 2019; Krikunov & Kovalchuk, 2015). Therefore, this section discusses the works on ensemble pruning in general and shows which methods have been transferred from classification or devised for time series forecasting.

Ensemble pruning is a desirable and widely popular method to overcome the deficiency of high computational costs of traditional ensemble learning techniques and improve their performance. An ensemble with a very large number of models may add a lot of computational overhead due to the large memory requirements of some of its base models, e.g. decision trees (Margineantu & Dietterich, 1997). The optimization of run-time overhead is imperative for certain applications where real-time requirements have to be met, such as in online forecasting. In addition, when models are distributed over a network, the reduction of models leads to a reduction of the resulting communication costs (Tsoumakas et al., 2009). Furthermore, both theoretical and empirical studies have shown that ensemble performance depends on the performance of its individual models and how diverse they are

[1] https://github.com/MatthiasJakobs/os-pgsm/tree/ecml_journal_2022.

(Li et al., 2012; Brown et al., 2005; Martinez-Munoz et al., 2008; Caruana et al., 2004). All the previous works agree that encouraging diversity is beneficial to the performance of ensemble, but it is hard to tell the theoretical properties of diversity in ensemble (Li et al., 2012). Therefore, understanding and promoting ensemble diversity remains an important research question in ensemble learning. The generalization performance of an ensemble depends on its empirical error and diversity (Li et al., 2012). It is thus reasonable to design the model selection criterion accordingly. However, it is very challenging to decide which models exactly need to be selected. Hence, given a set of base learners, it is not easy to estimate the generalization performance of a sub-ensemble. In addition, finding the optimal subset is a combinatorial search problem with exponential computational complexity. Thus, it is infeasible to compute the exact solution by exhaustive search and approximate search is needed. Several methods have been proposed in the literature to solve this issue by searching near-optimal solution using either directly using global search (Zhou et al., 2002; Zhang, 2002; Chen et al., 2009) or iteratively using greedy search (Martinez-Munoz et al., 2008; Partalas et al., 2012). Greedy search methods can be further divided into greedy *forward* pruning which starts with an empty set and iteratively adds the learners optimizing a certain criterion, and greedy *backward* pruning that starts with the complete ensemble and iteratively eliminates learners. It has been shown that greedy pruning methods are able to achieve comparative performance and robustness with global search methods but at much smaller computational costs (Li et al., 2012). Both global and greedy search methods can be further divided into three main families based on the paradigm used for the search (Tsoumakas et al., 2009).

The first family encloses ranking-based methods where the ensemble base models are sorted according to a selection criterion. Kappa pruning (Margineantu & Dietterich, 1997) is amongst the most popular methods in this family for classification and uses a diversity measure for the selection. It ranks all pairs of base classifiers based on the κ statistic of agreement calculated on the training set. Kappa pruning could be applied to regression or forecasting by reformulating a suitable pairwise diversity measure. In this context, Ma et al. (2015) transferred several evaluation measures such as Complementarity (Martınez-Munoz & Suárez, 2004), Concurrency (Banfield et al., 2005), and Reduce Error (Margineantu & Dietterich, 1997), for rank-based ensemble pruning using a *forward* greedy search procedure, to time series forecasting. It was shown that Complementarity and Reduce Error have the same flaw since they can not guarantee that the base model supplementing the ensemble the most at a given iteration is the one that will be selected. This is mainly explained by the fact that the error in time series forecasting is directional. Therefore, it is not very reasonable to only focus on decreasing the value of the forecasting error while ignoring its direction. Reduce Error-trend that takes into account the trend of time series and the direction of forecasting error, is then suggested by the authors to mitigate the above issue.

The second family relies on clustering (Giacinto et al., 2000; Lazarevic & Obradovic, 2001). Methods of this family consist of two stages. First, a clustering algorithm is employed in order to discover groups of models that are similar, i.e. make similar predictions. In a second stage, a selection of each cluster's representative is performed to increase the overall diversity of the ensemble. The main issue in this method is the choice of the similarity measure according to which the models will be evaluated, as well as the optimal number of clusters, i.e. the resulting ensemble size after pruning (Tsoumakas et al., 2009).

The third family of methods include optimization-based methods. Different optimization techniques can be employed, including genetic algorithms (Zhou & Tang, 2003), semi-definite programming (Zhang et al., 2006), hill climbing (Caruana et al., 2004) and meta-learning (Partalas et al., 2006). In the context of forecasting, a meta-learning

approach for the estimation of ensemble forecasts errors using regression is used in Krikunov and Kovalchuk (2015) to implement the selection procedure for each forecast. A more sophisticated approach inspired from classification is proposed in Zhang et al. (2021), where Extreme Learning Machines (ELMs) and Hierarchical Extreme Learning Machines (H-ELMs) are integrated as the base models, and four distinct meta-attributes collections, i.e., hard prediction, local accuracy, global accuracy, and prediction confidence, are presented. Each set of meta-attributes is joined to a specific assessment criterion, constructing thus a meta-data set. A meta-learner is trained on this data and used subsequently for deciding whether a base learner should be included in the ensemble or not.

This list of pruning method families is not exhaustive and several paradigms are used to serve this task. A comprehensive review can be found in Tsoumakas et al. (2009). However, most importantly for the time series domain, pruning methods need to be dynamic in order to cope with the time-evolving nature of time series that can be subject to significant changes, more precisely to the so-called concept drift phenomenon (Gama et al., 2014; Krawczyk et al., 2017). Dynamic methods can adapt the pruning strategy either in a blind manner over time, i.e. at each time instant at test time or periodically (Krikunov & Kovalchuk, 2015; Zhang et al., 2021), or in an informed fashion following concept-drift detection (Saadallah et al., 2019). A Drift-aware ensemble pruning for time series forecasting through adaptive model selection using correlation-based measures for similarity and dynamic clustering of the top-similar model with the target time series is proposed in Saadallah et al. (2019).

The last thing to note is that the ensemble combination/aggregation stage can also be used to serve the pruning stage implicitly by learning ensemble weighting schemes. This is achieved by assigning different weights to the ensemble base models and setting some to zero in order to exclude some of the base models. Several methods for automatically learning dynamic ensemble weighting schema for time series forecasting are suggested in the literature (Cerqueira et al., 2018; Saadallah et al., 2021; Gaillard & Goude, 2016; Cerqueira et al., 2017; Saadallah & Morik, 2021). However, not all of them guarantee that some of the weights would be set to zero (Saadallah et al., 2021; Gaillard & Goude, 2016; Cerqueira et al., 2017; Saadallah & Morik, 2021). In Cerqueira et al. (2018), the authors frame their aggregation as a ranking task, in which experts are ranked sequentially by their decreasing weight (i.e. the one predicted to perform better is ranked first). Correlation among the output of the base learners is used to quantify their redundancy. A given learner is penalised for its correlation to each learner already ranked. If it is fully correlated with other learners already ranked, its weight becomes zero. Opposingly, if it is completely uncorrelated with its ranked peers, it gets ranked with its original weight.

Deep learning algorithms construct complex models that are opaque for humans. For some areas such as health care or autonomous systems, e.g. self-driving cars, where a deep understanding of the AI application and the corresponding model behaviour is crucial, there is a great need for Explainable Artificial Intelligence (XAI) (Lamy et al., 2019; Tjoa & Guan, 2020; Zablocki et al., 2021). It is possible to use model-agnostic methods for explainability, such as local interpretable models that are used to explain individual predictions of black box machine learning models (Ribeiro et al., 2016). However, there are two main reasons for why it is necessary to consider explainability methods that are specifically developed for neural networks. First, neural networks learn features and concepts in their hidden layers and specific tools are required to reveal them. Second, the gradient can be exploited to implement explanation methods that are much more efficient than model-agnostic methods since they look into the inside dynamics of the model. A wide variety of explanation methods have been presented

in the literature (Camburu, 2020; Samek et al., 2021; Molnar, 2020). These methods can be grouped into three main families.

The first family of methods consists of visualization-based approaches where either new learned features by the DNN (Olah et al., 2017) or most important input data parts for the DNN's output (i.e. decision) (Simonyan et al., 2013; Selvaraju et al., 2017) are visualized. In the former, feature visualization is used for making the learned features represented in an explicit way. Feature visualization for a "unit" of a neural network (i.e. a "unit" refers either to individual neurons, feature maps (channels), entire layers or the corresponding pre-softmax neuron in the case for classification) is done by finding the input of the unit that maximizes the corresponding activation. In the latter, the so-called heat or saliency maps are used to establish a relationship between the output and the input of a DNN given fixed weights. These maps are widely used in the context of computer vision with CNNs to create class-specific heat-maps based on a particular input image and a chosen class of interest (Selvaraju et al., 2017). These maps are used for visualizing regions in the input image that are the most important for a particular prediction/ decision of the model. They are computed using the gradient of the networks prediction with respect to the input, holding the weights fixed. This determines which input elements (e.g. which pixels in case of an input image) need to be changed the least to affect the prediction the most.

The second family includes the so-called "conceptual" explanations. A concept can be defined as any abstraction, e.g. a colour in the case of an image, an object, a data property or even an idea. Given any user-defined concept, although a neural network might not be explicitly trained with the given concept, the concept-based approaches are devised to determine whether this concept is embedded within the latent space learned by the DNN or not (Kim et al., 2018; Küsters et al., 2020).

The third family of methods are model-based approaches that use model distillation to explain a neural network with a simpler model (Frosst & Hinton, 2017; Cheng et al., 2020). For example, in Frosst and Hinton (2017), the authors express the knowledge acquired by the neural net in a decision tree that generalizes better than the one learned directly from the training data. Since it relies on hierarchical decisions, explaining a particular decision is made much easier.

In Saadallah et al. (2021), the authors use the most salient parts of time series, called the models Region of Competence, to forecast new, unseen data points by comparing the new data to its historical record. To extract these Regions of Competence, they use a variation of Grad-CAM (Selvaraju et al., 2017), which explains the impact of parts of the time series on the model loss, rather than the model prediction itself.

In this work, we combine clustering with a rank-based approach to build an online drift-aware two-staged pruning procedure for time series forecasting. The drift detection will not cover only the time series but also models dependencies/performance over time. In addition, the final number of selected models is set depending on the number of clusters in order to enhance both ensemble's accuracy and diversity. Moreover, we extend the methodology proposed in Saadallah et al. (2021) to ensemble pruning, which results in a more traceable and comprehensible pruning process.

3 Methodology

An online two-staged pruning procedure based on clustering and global ranking is developed to prune an ensemble of deep learning models for time series forecasting by taking into account both **accuracy** and **diversity** at each stage. Both stages are dependent on each other and a bound for the optimal number of final models to be kept is derived. The pruning is not directly performed using the base deep models' outputs but is based on their precomputed Regions-of-Competences (RoCs) which have been proved to be very successful method for single online model selection for time series forecasting (Saadallah et al., 2021). The pruning is also made in an adaptive informed manner following concept drift detection in time series and models' RoCs dependencies. More details about the method are provided in this Section. The RoCs are calculated using performance-based heat-mapping method similar to the way presented in Saadallah et al. (2021). A quick reminder of the method is also be presented.

3.1 Notations and definitions

A time series X is a temporal sequence of values, where $X_t = \{x_1, x_2, \dots, x_t\}$ is a sequence of X until time t and x_i is the value of X at time i. Denote with $\mathbb{P} = \{f_1, f_2, \dots, f_N\}$ the pool of N CNN-based models trained to approximate a true unknown function g that generated X. Let $\hat{x}_{t+f} = (\hat{x}^{f_1}_{t+f}, \hat{x}^{f_2}_{t+f}, \dots, \hat{x}^{f_N}_{t+f})$ be the vector of forecast values of X at a future time instant $t+f, f \geq 1$ (i.e. x_{t+f}) by each of the models in $\mathbb{P}$. A ensemble model $\bar{f}_{\mathbb{P}}$ of $\mathbb{P}$ at time instant $t+f$ can be formally expressed as a convex combination of the forecasts of the models in $\mathbb{P}$.

$$\bar{f}_{\mathbb{P}}(\hat{x}_{t+f}) = \sum_{j=1}^{N} w_j \hat{x}^{f_j}_{t+f} \tag{1}$$

where $w_j, j \in [1, N]$ are the ensemble weights. The weights are constrained to be positive and sum to one. This constraint is necessary for some of the following results. For simplicity, we set the weights to be equal, i.e., $w_j = \frac{1}{N} \forall j \in [1, N]$.

$$\bar{f}_{\mathbb{P}}(\hat{x}_{t+f}) = \frac{1}{N} \sum_{j=1}^{N} \hat{x}^{f_j}_{t+f} \tag{2}$$

The goal of dynamic online ensemble pruning is to identify the subset of models $\mathbb{S} \subset \mathbb{P}$ that should compose the ensemble at each time step $t+f$ such that the expected prediction error of the pruned ensemble is reduced compared the the full ensemble $\bar{f}_{\mathbb{P}}$ for each forecast.

$$\underset{\mathbb{S} \subset \mathbb{P}}{\operatorname{argmax}} \quad \mathbb{E}\left[\left(x_{t+f} - \bar{f}_{\mathbb{P}}(\hat{x}_{t+f})\right)^2 | X_{t+f-1}\right] - \mathbb{E}\left[\left(x_{t+f} - \bar{f}_{\mathbb{S}}(\hat{x}_{t+f})\right)^2 | X_{t+f-1}\right] \tag{3}$$

The pruning is performed using the so-called Regions-of-Competence (RoCs) or expertise of the models computed by the Performance Gradient-based Saliency Maps (PGSMs) (Saadallah et al., 2021). To do so, we divide the time series X_t into $X^{train}_{\omega} = \{x_1, x_2, \dots, x_{t-\omega}\}$ and $X^{val}_{\omega} = \{x_{t-\omega+1}, x_{t-\omega+2}, \dots, x_t\}$, with a provided time window size ω. X^{train}_{ω} is used for training the models in $\mathbb{P}$ and X^{val}_{ω} is used to compute the RoCs using the PGSMs, since to

measure models performance both true and predicted values of the time series are required. The RoCs for each model $f_j, j \in \{1, \dots, N\}$ are obtained by performing time-sliding window operations of size $n_\omega, n_\omega < \omega$ over X_ω^{val} either by one step or by z steps. The resulting time-windows from X_ω^{val} are denoted by $X_{n_\omega}^{val,i}$ with $i \in [1, Z]$ and Z is the total number of resulting windows.

3.2 Base learners

The ensemble base models are CNN-based models that share more or less the same basic types of layers. The common basic structure consists of a sequence of 1D-convolutional layers with different number of filters, followed by a batch normalization layer, in some cases a LSTM layer and an output layer of one neuron. The different architectures are obtained by varying the number of the convolutional layers and their corresponding parameters (i.e. the number of filters) and in some cases adding or removing another neural network type to the last convolutional layer, like a LSTM layer. To obtain further architectures variations, the number of units in the LSTM are also varied. The candidate CNNs are devised such that they use the same L-lagged values of the time series as input to forecast the following time value.

3.3 RoCs computation

The PGSMs are first introduced in Saadallah et al. (2021) and inspired by the class activation saliency maps, more specifically, Grad-CAM (Selvaraju et al., 2017). However, instead of using these maps to derive the importance of certain features for a given class, a mapping between the performance of a given forecasting model and a specific time interval is established. The performance of each model $f_j, j \in \{0, \dots, N\}$ is evaluated using an error-related measure, namely the Mean Squared Error, ϵ_j^i on $X_{n_\omega}^{val,i}$: the ith time interval window of X_ω^{val} of size n_ω. The goal is to derive an estimate of the relevance of each time point in $X_{n_\omega}^{val,i}$ to a certain performance of the model measured by ϵ_j^i of f_j. The intuition behind this method is similar to Grad-CAM exploiting the spatial information that is preserved through convolutional layers, in order to understand which parts of an input image are important for a classification decision. The focus here is on the temporal information explaining certain behaviours of f_j. Therefore, the layer which has produced the last feature maps f_{maps} is considered. For each activation unit u at each generic feature map A, an importance weight w^ϵ associated with ϵ_j^i, is obtained. This is done by computing the gradient of ϵ_j^i with respect to A. Subsequently, a global average over all the units in A is computed:

$$w^\epsilon = \frac{1}{U} \sum_u \frac{\partial \epsilon_j^i}{\partial A_u} \tag{4}$$

where U is the total number of units in A. We use w^ϵ to compute a weighted combination between all the feature maps for a given measured value of the error ϵ_j^i. Since we are mainly interested in highlighting temporal features contributing most to ϵ_j^i, ReLU is used to remove all the negative contributions by:

$$L_j^i = ReLU(\sum_{f_{maps}} w^\epsilon A) \tag{5}$$

$L_j^i \in \mathbb{R}^U$ is used to find the regions in $X_{n_\omega}^{val,i}$ that have mainly contributed to ϵ_j^i of the network f_j. Note that the candidates are designed such that $U < n_\omega$. Note also that different time-windows $X_{n_\omega}^{val,i}$ of size n_ω are created out of X_ω^{val}, so that several performance evaluations of the same model on different windows can take place and the number of RoCs for each model is therefore increased. However, the work in Saadallah et al. (2021) is focused around single best model selection. That is why a ranking of the models on each $X_{n_\omega}^{val,i}$ is performed. Only the RoC of the best model is computed. At test time, the selection of the forecaster is corresponding to the model having the closest RoC to the most recent time series pattern. Opposingly, in this work, the performance (i.e. error) of all the candidate models is measured and the RoC for each model is computed. This can be viewed as computing the most important time series interval responsible for a certain observed performance of each base model in the ensemble. In this way, a buffer RoC_j that contain all the pre-computed RoCs R_j^i on the different validation windows $X_{n_\omega}^{val,i}, i \in [1, Z]$ for each model $f_j, j \in [1, N]$ is created, i.e., $RoC_j = \{R_j^1, \ldots, R_j^Z\}$. It is important to note that in order to obtain one continuous region RoC R_j^i within the time series sequence $X_{n_\omega}^{val,i}$, a smoothing operation is applied to L_j^i. This is done by normalizing L_j^i values between 0 and 1 and applying a threshold $\tau = 0.1$ to filter out smaller values (i.e. these values are set to 0). In addition, a moving-average is applied where each point is compared to the previous and the subsequent value. After the smoothing operation some regions may become empty. This is mainly due to some low L_j^i (i.e. low relevance of the time point) or high discontinuity.

The base models use the same L-lagged values of the time series as input, $X_{t+f-1}^L = \{x_{t+f-L}, \ldots, x_{t+f-1}\}$, $(t+f \geq k)$.In addition to the smoothing employed in Saadallah et al. (2021), in order to constrain the RoCs lengths to L, we reject the smoothed RoCs with length different from L. At test time, in order to forecast the value of X at $t+f, f \geq 1$, the similarity between the input pattern X_{t+f-1}^L and the RoCs for each model in $\mathbb{P}$ is computed. Euclidean distance (Euc) is used to measure the similarity between X_{t+f-1}^L and each $R_j^i, \forall i \in [1, Z], \forall j \in \{1, \ldots, N\}$, within each $RoC_j, \forall j \in \{1, \ldots, N\}$ buffer. For each model $f_j, \forall j \in \{1, \ldots, N\}$, the RoC $\mathcal{R}_j$ satisfying:

$$\mathcal{R}_j = \underset{R_j^i \in RoC_j}{\operatorname{argmin}} \, Euc(R_j^i, X_{t+f-1}^L) \tag{6}$$

is selected to represent f_j for the ensemble pruning for $t+f$.

3.4 Online ensemble pruning

The pruning decision is performed in a step-wise manner online at each time forecast $t+f, f \geq 1$. For simplicity of notation, we assume $f = 1$. The expected error of the ensemble of the models in $\mathbb{P}$ $e_{\bar{f}}$ at a future data point x_{t+f} can be expressed as follows:

$$\begin{aligned} e_{\bar{f}}(x_{t+1}) &= (x_{t+1} - \bar{f}(\hat{x}_{t+1}))^2 \\ &= \frac{1}{N}\sum_{j=1}^{N}(x_{t+1} - \hat{x}_{t+1}^{f_j})^2 - \frac{1}{N}\sum_{j=1}^{N}(\hat{x}_{t+1}^{f_j} - \bar{f}(\hat{x}_{t+1}))^2 \\ &= \bar{e}(x_{t+1}) - \bar{a}(x_{t+1}) \end{aligned} \tag{7}$$

The left term in Eq. 7 refers to the weighted average error of the base models $\bar{e}$ and the right term to the ensemble ambiguity $\bar{a}$ which is simply the variance of the ensemble around the

weighted mean and it measures the disagreement between networks on x_{t+1}. The above relations can be averaged over several H time steps and the ensemble generalization error can be written as:

$$\begin{aligned} E_{\bar{f}} &= \frac{1}{H}\sum_{f=1}^{H} e_{\bar{f}}(x_{t+f}) \\ &= \frac{1}{H}\sum_{f=1}^{H}(\bar{e}(x_{t+f}) - \bar{a}(x_{t+f})) \\ &= \bar{E} - \bar{A} \end{aligned} \tag{8}$$

The RoCs $\mathcal{R}_j, \forall j \in [1, N]$, of each base model indicate the degree of expertise of the corresponding model in forecasting given the most recent input time series sequence pattern that is acquired to forecast the value of X at $t+1$, i.e., X_t^L (See Eq. 6). Since this selection is made based on the closeness of $\mathcal{R}_j$ to X_t^L, $\mathcal{R}_j$ can also be viewed as an estimate of X_t^L by f_j. In other words, the prediction by f_j of the data points in X_t^L are represented approximately by the data points $r_j^l \in \mathcal{R}_j, \forall l \in [1, L]$:

$$x_{t+l-L} \approx r_j^l, \forall l \in [1, L], t \geq L-1 \tag{9}$$

The ensemble $\bar{f}$ of $\mathbb{P}$ can the be expressed as:

$$\bar{f}_{\mathbb{P}}(x_{t+l-L}) = \frac{1}{N}\sum_{j=1}^{N} r_j^l = \bar{r}_l \tag{10}$$

The ensemble error on the pattern X_t^L using RoCs approximation $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_N\}$ by the models in $\mathbb{P}$ can be then written as:

$$\begin{aligned} E_{\bar{f}}^{\mathcal{R}} &= \frac{1}{L}\sum_{l=1}^{L}(x_{t+l-L} - \bar{r}_l)^2 \\ &= \frac{1}{L}\sum_{l=1}^{L}(\frac{1}{N}\sum_{j=1}^{N}(x_{t+l-L} - r_j^l)^2) - \frac{1}{L}\sum_{l=1}^{L}(\frac{1}{N}\sum_{j=1}^{N}(\bar{r}_l - r_j^l)^2) \\ &= \bar{E}^{\mathcal{R}} - \bar{A}^{\mathcal{R}} \end{aligned} \tag{11}$$

$\bar{E}^{\mathcal{R}} = \frac{1}{L}\sum_{l=1}^{L}(\frac{1}{N}\sum_{j=1}^{N}(x_{t+l-L} - r_j^l)^2)$ and $\bar{A}^{\mathcal{R}} = \frac{1}{L}\sum_{l=1}^{L}(\frac{1}{N}\sum_{j=1}^{N}(\bar{r}_l - r_j^l)^2)$. It is very intuitive to see from Eq. 11 that the selection should be made in favor of models whose RoCs are closer to the current pattern (i.e. considering the Euclidean distance) in order to minimize $\bar{E}^{\mathcal{R}}$ and diverse from each other in order to maximize $\bar{A}^{\mathcal{R}}$. To do so, we perform a two-staged pruning.

In a first stage, we start by clustering the candidate models using Euclidean distance into k clusters and select only cluster representatives to take part in the second pruning stage. Models belonging to different clusters are expected to have higher distance between them compared to models belonging to the same cluster. As a result, clusters' representatives have higher average distance to the average pattern $\bar{\mathcal{R}}_L = \{\bar{r}_1, \ldots, \bar{r}_L\}$, which contributes to increasing $\bar{A}^{\mathcal{R}}$. In addition, models belonging to the same cluster have more or less the same distance to the current pattern X_t^L. Therefore, the average error induced by all the models is expected to be similar to the averaged error induced by the clusters' representatives since one representative shows the same error level as all the models belonging to that

same cluster. As a result, $\bar{E}^{\mathcal{R}}$ is approximately preserved. In this way, we promote diversity without increasing the averaged error which reduces the expected ensemble error $E^{\mathcal{R}}_{\bar{f}}$ on X^L_t. The selection of a model $f_r, r \in [1, N_C]$ to be representative of a given cluster of models $\mathcal{C}_m, \forall m \in [1, k]$ where N_C is its size, is done based on its closeness to the current pattern X^L_t.

$$f_r = \underset{\mathcal{R}_j \in \mathcal{C}_m}{\operatorname{argmin}} Euc(\mathcal{R}_j, X^L_{t+f-1}), \forall m \in [1, k] \tag{12}$$

This helps to reduce the averaged error of the base models. The clustering of the RoCs is done using K-means (Burkardt, 2009) with Euclidan distance as similarity measure.

In a second stage, a selection of the best performing clusters representatives, denoted here as ***top-M*** models, is computed using ranking. As the second stage is dedicated to reducing the averaged error $\bar{E}^{\mathcal{R}}$ while promoting the diversity even further, the ranking is computed using the distance of the candidate clusters' representatives to the current pattern X^L_t and a bound for the radius δ of the L-sphere of center X^L_t enclosing the ***top-M*** models that should be preserved in this stage, is derived:

$$\frac{1}{2}\sqrt{\frac{\sum_{l=1}^{L}\sum_{j=1}^{k}(\bar{r}_l - r^l_j)^2)}{k}} \leq \delta \leq \sqrt{\frac{\sum_{l=1}^{L}\sum_{j=1}^{k}(x_{t+l-L} - r^l_j)^2}{k}} \tag{13}$$

where k is the number of clusters (i.e. number of selected models from the first stage). Note that the ensemble's error is always positive, meaning that the ambiguity can be considered as a lower bound of the averaged error of the base models. This applies to the ensemble of the k clusters' representatives $E^{\mathcal{R}}_k$. We have then $\bar{E}^{\mathcal{R}}_k \geq \bar{A}^{\mathcal{R}}_k$ which means that we always satisfy that the upper bound of Eq. 13 is bigger than the lower bound of the same equation.

Proof To ensure a reduction of the averaged error, the error $\bar{E}^{\mathcal{R}}_{\textbf{top-M}}$ induced by the subset of the ***top-M*** models should be lower than $\bar{E}^{\mathcal{R}}_k$ induced by the subset of k cluster representatives.

$$\bar{E}^{\mathcal{R}}_{\textbf{top-M}} \leq \bar{E}^{\mathcal{R}}_k$$

$$\frac{1}{L}\sum_{l=1}^{L}\frac{1}{|\ \textbf{top-M}\ |}\sum_{j=1}^{|\textbf{top-M}|}(x_{t+l-L} - r^l_j)^2 \leq \frac{1}{L}\sum_{l=1}^{L}\frac{1}{k}\sum_{j=1}^{k}(x_{t+l-L} - r^l_j)^2$$

where $|\textbf{top-M}|$ is the cardinality of the ***top-M*** models. Since they are contained within the L-sphere of center X^L_t, we have $\sum_{l=1}^{L}(x_{t+l-L} - r^l_j)^2 \leq \delta^2, \forall j \in [1, |\textbf{top-M}|]$. Therefore, each single model f_j contained within the sphere has at most distance of δ to X^L_t. As a result:

$$\frac{\delta^2}{L} \leq \frac{1}{L}\frac{1}{k}\sum_{l=1}^{L}\sum_{j=1}^{k}(x_{t+l-L} - r^l_j)^2$$

$$\delta^2 \leq \frac{1}{k}\sum_{l=1}^{L}\sum_{j=1}^{k}(x_{t+l-L} - r^l_j)^2$$

$$\delta \leq \sqrt{\frac{\sum_{l=1}^{L}\sum_{j=1}^{k}(x_{t+l-L} - r^l_j)^2}{k}}$$

In addition, we want to ensure that the second stage either preserves the promoted diversity from the clustering stage or enhances it even further by preserving or increasing the ambiguity:

$$\bar{A}^{\mathcal{R}}_{\textbf{top-M}} \geq \bar{A}^{\mathcal{R}}_{k}$$

$$\frac{1}{L}\sum_{l=1}^{L}\frac{1}{|\ \textbf{top-M}\ |}\sum_{j=1}^{|\textbf{top-M}|}(\bar{r}_l - r_j^l)^2) \geq \frac{1}{L}\sum_{l=1}^{L}\frac{1}{k}\sum_{j=1}^{k}(\bar{r}_l - r_j^l)^2)$$

The average pattern $\bar{\mathcal{R}}_L$ of the ***top-M*** models is contained within the L-sphere of center X_t^L and the distance between $\mathcal{R}_j$ and $\bar{\mathcal{R}}_L$ can be then at most 2δ for all the models f_j with $j \in [1, |\textbf{top-M}|]$. Therefore:

$$\frac{4\delta^2}{L} \geq \frac{1}{L}\sum_{l=1}^{L}\frac{1}{k}\sum_{j=1}^{k}(\bar{r}_l - r_j^l)^2)$$

$$\delta \geq \frac{1}{2}\sqrt{\frac{1}{k}\sum_{l=1}^{L}\sum_{j=1}^{k}(\bar{r}_l - r_j^l)^2)}$$

□

In practice, in order to identify the ***top-M*** models for a fixed number of clusters k, we start by clustering the N models in $\mathcal{P}$ and we compute the clusters' representatives (See Eq. 12). Then, we calculate the distance of their RoCs to the current pattern X_t^L. We set δ simply to the upper bound of Eq. 13 (i.e. in this case the lower bound is naturally verified) and models whose distance is lower or equal to δ are selected as the ***top-M*** models to compose the ensemble that forecasts the value of X at $t+1$. The upper bound of Eq. 13 can then be interpreted as promoting the selection inside the L-sphere close to the true recent pattern X_t^L which reduces the averaged error while the lower bound can be viewed as a regularization parameter that disallows radius values that would result in a clustering of the base models very closely around X_t^L which decreases the ambiguity.

The models selected to compose the ensemble at $t+1$ (i.e. ***top-M***) are assumed to remain valid for the following time instants $t+f$ with $f > 1$ unless a concept-drift either in the models' dependencies/performance or in the time series data is detected. If a drift is detected, an alarm to update the pruning decision (i.e.***top-M*** models) is triggered. More details are provided in the following Subsection.

3.5 Drift-aware pruning update

In order to update the pruning decision at test time in an informed manner, two drift detection mechanisms are deployed.

3.5.1 Concept drift in time series

Due to the dynamic behaviour of time series, streaming upcoming values can be subject to significant changes, more specifically to concept drifts (Gama et al., 2014; Krawczyk et al., 2017). As a result, the base models' RoCs have to be updated in order to take into consideration the possible appearance of new patterns in the data and also to gain experience of

which models are better to compose the ensemble to deal with these patterns if they ever reoccur again. The old models' RoCs buffers are maintained and they are enriched with new ones. To do so, if a concept drift in the time series is assumed to take place at a given time instant $t+f, f > 1$, an alarm is triggered to update of the models' RoCs by updating X_{ω}^{val} on which the RoCs are originally computed. This is done by sliding X_{ω}^{val} to include the new recent observations. The detection of concept drifts is performed by monitoring the deviation Δm_{t_f} in the mean of the time series till $t+f$ (Saadallah et al., 2019): $\Delta m_{t_f} = \mathbb{E}(X_{t_f}) - \mu_t$, with $\mu_t = \mathbb{E}(X_t), t \leq t_f$, the initial computed mean of X up to time t. A drift is assumed to take place at t_f if the true mean of Δm_{t_f} diverges in a significant way from 0. We propose to detect the validity of this using the well-known Hoeffding-Bound (Hoeffding, 1994), which states that after W independent observations of a real-valued random variable with range r, its true mean has not diverged if the sample mean is contained within $\pm\xi_m$:

$$\xi_m = \sqrt{\frac{r^2 \ln(1/\sigma)}{2W}} \tag{14}$$

with a probability of $1-\sigma$ (a user-defined hyperparameter). Once $\left|\Delta m_{t_f}\right|$ exceeds ξ_m, an alarm is triggered and the reference mean μ is reset by setting $t = t_f$. This checking procedure is continuously applied online at forecasting time.

The update of the RoCs triggers in turn the update of pruning. Since new RoCs can be added to the different RoCs buffers, changes in the distances of these RoCs to the most recent pattern X_{t+f-1}^{L} are expected. As a result, new RoCs representatives of the models can appear (See Eq. 6) and re-clustering of the models and re-selection of the ***top-M*** then become necessary.

3.5.2 Concept drift in models' performance

Base models' performance is reflected using the distance of their representative RoCs to the current pattern. If we consider forecasting the time series value at $t+1$, the distances measured using the most recent pattern X_t^L can be viewed as a measure of dependencies between the set of base models and the target time series. These dependencies can be continuously computed and monitored over time. The distance d_j^t of the model f_j to X_t^L is calculated using its representative RoC $\mathcal{R}_j$ for the forecasting at time $t+1$. (Eq. 6).

$$d_j^t = Euc(X_t^L, \mathcal{R}_j), \forall j \in [1, N] \tag{15}$$

Naturally, with time-evolving data, dependencies change over time and follow non-stationary concepts. Stationarity in this context can be formulated as follows:

Definition 1 (Weak stationary Dependencies) Let $D_t = \{d_1^t, \ldots, d_N^t\} \in \mathbb{R}^N$ be a resulting similarity vector between the base models and the target time series X_t^L over a window of size L (i.e. derived from the above similarity metric Eq. 15). We sort the value of D_t in an ascending order so that $D_t = \{d_{1,t}, \ldots, d_{N,t}\}$ with $d_{1,t} \leq \cdots \leq d_{N,t}$. Let μ_d denote the minimum value in D_t at the initial instant of its generation $t_i = t$. The dependence structure is said to be *weakly stationary* if the true mean of Δdif_t is 0:

$$\Delta dif_t = \left|d_{1,t} - \mu_d\right| \tag{16}$$

Following this definition, we can assume that the distance between the most similar models to the current pattern within the same pool of models $\mathcal{P}$ sets its boundary under a form of a logical *diameter*. If this boundary diverges in a significant way over time, a drift is assumed to take place. We propose to detect the validity of such assumption using the well-known Hoeffding Bound ξ_d similarly to the way suggested for detecting the drift in the deviation of the mean of the time series (Eq. 14). Once the condition of the *weak stationary dependencies* presented in Definition 1 is violated (i.e. $\Delta dif_t \geq \xi_d$), an alarm is triggered, the ensemble pruning is updated by re-clustering of the models and re-selecting of new ***top-M*** models. Afterwards, the dependencies monitoring process is continued by sliding the time window to update X_t^L by one value to produce the next forecast. Once a drift is detected at time instant t_d, the reference *diameter* μ_d is reset by setting $t_i = t_d$.

Our method is denoted in the following as **OEP-ROC**: Online Ensemble Pruning using performance saliency maps-based Regions-Of-Competence. The concept drift detected in the time series is denoted as `Drift Type I` while the drift in base models' performance is denoted as `Drift Type II`. All the steps of **OEP-ROC** are summarized in Algorithm 1.

Algorithm 1 OEP-ROC

Parameters: Number of Lags:L; size of the validation set: ω; size of time windows within the validation set: n_ω; CNNs Pool of size N: $\mathbb{P}$; Number of cluster: k.

1: **Models Training and RoCs Computation**:
2: Train each $C_j \in P_{CNN}, j \in \{0, \cdots, N-1\}$ on X_ω^{train}.
3: Initialize RoC buffers RoC^j for each $C_j, j \in \{0, \cdots, N-1\}$
4: **for** each $X_{n_\omega}^{val,i} \in X_\omega^{val}$ **do**
5: Compute the corresponding R_i^j using PGSMs (i.e. L_j^i Eq. 5)
6: Add R_i^j to the corresponding buffer RoC^j
7: **end for**
8:
9: **Online Forecasting**: Forecasting next N_f values :
10: To forecast $t+1$:
11: **for** each $j \in [1, N]$ **do**
12: Select the representative RoC $\mathcal{R}_j$ for f_j (Eq.6).
13: **end for**
14: Cluster the models into k clusters using their representative RoCs
15: Select the k clusters' representatives following Eq.12
16: Sort the k representatives according to their distance to X_t^L.
17: Select the ***top-M*** models whose distances are lower than the upper bound in Eq.13. Predict x_{t+1} using the selected ***top-M*** models using eq.2.
18: To forecast $t+f$:
19: **for** $f \in \{2, \cdots, N_f\}$ **do**
20: **if** `Drift Type I` detected **then**
21: Update $X_\omega^{val} = \{x_{t-\omega+j}, x_{t-\omega+2}, \cdots, x_{t+j-1}\}$
22: Recompute and add new RoCs (steps:4-7)
23: **end if**
24: **if** `Drift Type I` $\vee$ `Drift Type II` detected **then**
25: Use the most recent pattern X_{t+f-1}^L
26: Re-cluster and reselect the ***top-M*** models (steps:11-17)
27: **end if**
28: **end for**

4 Experiments

We present the experiments carried out to validate **OEP-ROC** and to answer these research questions:

- *Q1* How does **OEP-ROC** perform compared to the the State of the Art (SoA) and existing online ensemble pruning methods for time series forecasting?
- *Q2* What is the importance of each pruning stage in **OEP-ROC**?
- *Q3* What is the benefit of each drift type detection for the performance of **OEP-ROC**?
- *Q4* What is the impact of different values of the number of clusters k on the performance of **OEP-ROC**?
- *Q5* What is the impact of choosing the size of ***top-M*** automatically using the bound in Eq. 13?
- *Q6* How can different aggregation techniques benefit from our pruning method?
- *Q7* How scalable is **OEP-ROC** in terms of computational resources compared to the most competitive online model selection methods? What is the computational advantage of the **drift-aware** adaption of pruning?
- *Q8* How can **OEP-ROC** be exploited to provide suitable explanations for the reason behind selecting specific models to compose the ensemble at a certain time interval or instant?
- *Q9* How can **OEP-ROC** be used also to provide reasonable explanations for the performance of the selected ensemble at a certain time interval or instant?

4.1 Experimental setup

The methods used in the experiments were evaluated using the root mean squared error (RMSE). The used time series was split into three parts, where the first 50% is used for training (X_{ω}^{train}), the next 25% for validation (X_{ω}^{val}) and the last 25% for testing. The results are compared using the non-parametric Wilcoxon Signed Rank test. We use 100 real-world time series shown in Table 1 for our experiments. For computational reasons, we chose to sample random time series for each dataset to gather a total of 100 diverse time series.

4.2 OEP-ROC setup and baselines

We construct a pool $\mathcal{P}$ of CNN-based candidate models using different parameter settings (e.g. number of filters varies in $\{32, 64, 128\}$, kernel size varies in $\{1, 3\}$), like explained in Sect.n 3.2. For construction of the base learners, we define four architectural building blocks. Our notation of *layer1-layer2* implies a sequential connection between the two layers. Let *Conv1* be a sequential subnet made up of one convolutional layer with ReLU activation, followed by a batch normalization layer. *Conv2* is similar, except that we use max pooling instead of batch normalization. *Conv3* is the same as *Conv1*, but the number of filters for the convolutional layer is reduced by half. Lastly, we define a *ResidualBlock* as *Conv-BatchNorm-ReLU-Conv-BatchNorm-ResidualConnection-ReLU*. From these building blocks, we create our base learners as in Table 2, where each Dropout layer has a probability parameter of 0.9 and *FCN* refers to a fully-connected layer. There, we also show the different configurations we created by varying the number of filters in the convolutional layers as well as the number of hidden units in the LSTM layer.

OEP-ROC has also a number hyper-parameters that are summarized in Table 3.

Table 1 List of datasets used for the experiments

Name	Nr. of time series	Source	Characteristics
Amount registered	1	Bike sharing (Cerqueira et al., 2017)	Hourly, January 1–March 01, 2011
AbnormalHeartbeat	1		3053 measurements (4 kHz)
CatsDogs	1		14,773 audio samples (16 kHz)
Cricket	1		1197 accelerometer readings (184 Hz)
EOGHorizontalSignal	1	UEA	1250 measurements (1 kHz)
EthanolConcentration	1	& UCR (Bagnall et al., 2017)	1 s spectrum measurement
Phoneme	1		1024 samples of audio
Rock	1		2844 samples of spectrum analysis
SNP500	1		
DJI	1	UCI (Dua & Graff, 2017; Hosein-zade & Haratizadeh, 2019)	Daily closing, 2010–2017
NYSE	1		
RUSSELL	1		
Electricity (Hourly)	11		Energy consumption measurements
KDD Cup 2018	13	Monash	Forecast air quality indices (AQIs)
Pedestrian Counts	12	Forecasting	Hourly pedestrian counts from Melbourne
Solar (10 min)	12	Benchmark	Solar power production
M4 (Daily)	12	(Godahewa et al., 2021)	Daily time series from M4 dataset
M4 (Weekly)	13		Weekly time series from M4 dataset
Weather	15		Daily weather forecasts

Table 2 Configurations and architectures for all base learners

Base learner	Architecture	Configurations
Shallow FCN	Conv1-Dropout-FC	$n_{filters} \in \{32, 64, 128\}$
Small CNN	Conv2-LSTM	$n_{filters} \in \{32, 64, 128\}$ $n_{hidden} \in \{10, 30\}$
Medium CNN	Conv1-Conv1-LSTM	$n_{filters} \in \{32, 64, 128\}$ $n_{hidden} \in \{10, 30\}$
Large CNN	Conv1-Conv1-Conv1-LSTM	$n_{filters} \in \{32, 64, 128\}$ $n_{hidden} \in \{10, 30\}$
Fewer Filter CNN	Conv3-Conv3-LSTM	$n_{filters} \in \{32, 64, 128\}$ $n_{hidden} \in \{10, 30\}$
One Residual	ResBlock-Dropout-FCN-Dropout-FCN	$n_{filters} \in \{32, 64, 128\}$
Two Residual	ResBlock-Resblock-Dropout-FCN-Dropout-FCN	$n_{filters} \in \{32, 64, 128\}$

Different configurations are generated by taking all combinations of filters and hidden units as described in the last column. In total, this results in 33 base learners

Table 3 Hyperparameters of our method and their values for the experiments

Parameter	Description	Value
N	Size of the Pool of base models $\mathbb{P}$	33
n_ω	Size of time windows within the validation set	60
z	Number of steps for sliding the n_ω time windows	25
L	Number of lags for training the base models	5
k	Number of base models clusters	$\{5, 10, 15, 20\}$
σ	Hoeffding-Bound parameter	0.05

We compare **OEP-ROC** against the following approaches which include SoA methods for forecasting and ensemble pruning methods devised in the context of forecasting. Some of them operate in an online fashion.

- *SoA Forecasting Models*

  ARIMA (Box et al., 2015): Auto-Regressive Moving Average model.
  LSTM (Gers et al., 2002): Long Short Term Memory Network.
  ETS (Jain & Mallick, 2017): Exponential Smoothing model.
  CNN, **CNN-LSTM** (Romeu et al., 2013): The two best performing base models.

- *Online SoA Pruning Methods*:

  Ran-Pr-*m* : Random selection of base models to construct the ensemble with m indicating the ensemble size.
  Ens: Ensemble of all the base modes in $\mathbb{P}$.
  NCL (Mozaffari & Azad, 2014): Negative correlation Learning for pruning ensembles of deep learning methods. We use the same pool of base models as $\mathbb{P}$.
  OCL (Saadallah et al., 2019): Online drift-aware clustering of the base modes in $\mathbb{P}$ using covariance-based clustering.
  OTOP (Saadallah et al., 2019): Online drift-aware Top best performing models ranking using temporal correlation analysis.
  DEMSC-C (Saadallah et al., 2019): Dynamic Ensemble Members Selection using Clustering: Online drift-aware Top best performing models ranking using temporal correlation analysis combined with covariance-based clustering.
  DEMSC-K (Saadallah et al., 2019): Same as **DEMSC-C** but uses K-means clustering using Dynamic Time Wrapping.
  OS-PGSM (Saadallah et al., 2021): Online single model Selection using Performance Gradient-based Saliency Maps from the pool $\mathbb{P}$ that uses the principle of selection using gradient based RoCs and Hoeffding-based drift detection mechanism in the time series to update the RoCs.
  OSPGM-Int (Saadallah et al., 2019): Same as **OS-PGSM** but the time windows of size n_ω are slided with step size $z = n_\omega$.

- *OEP-ROC Variants*:

  OEP-ROC-C: Variant of **OEP-ROC** that uses only clustering without ***top-M*** selection.

OEP-ROC-TOP: Variant of **OEP-ROC** that performs ***top-M*** selection without clustering.
OEP-ROC-ST: Static variant of **OEP-ROC**. Pruning decided at the initial forecasting instant and kept fixed along testing.
OEP-ROC-Per: Pruning is updated periodically in a blind manner (i.e. without taking into account the occurrence of the drift).
OEP-ROC-I: Variant of **OEP-ROC** that takes into account only the occurrence of `Drift Type I`.
OEP-ROC-II: Variant of **OEP-ROC** that takes into account only the occurrence of `Drift Type II`.
OEP-ROC-k: Variant indicating the number k of clusters used in the clustering stage of **OEP-ROC**.
OEP-ROC-k-topm-M: In **OEP-ROC** and all its above variants the size of ***top-M*** (i.e. $|\mathbf{top\text{-}M}|$) is decided automatically using the bound in Eq. 13. In this variant, we set the size of models to select to a fixed value of M.

We also evaluate how different ensemble aggregation methods (i.e. ensemble weighting methods) can benefit from our pruning strategy. Instead of feeding all the base models in $\mathbb{P}$ into the aggregation schema, we only input the models selected by pruning. To do so, we report the evaluation results over various aggregation methods including:

OEP-ROC-SW: Variant of **OEP-ROC** that uses sliding-window ensemble (Saadallah et al., 2018) for aggregation instead of equal weighting.
SW: Sliding-window ensemble (Saadallah et al., 2018) over all the models in $\mathbb{P}$.
OEP-ROC-OGD: Variant of **OEP-ROC** that uses Online Gradient Descent (Zinkevich, 2003) for aggregation.
OGD: Online Gradient Descent (Zinkevich, 2003) aggregation over all the models in $\mathbb{P}$.
OEP-ROC-FS: Variant of **OEP-ROC** that uses Fixed Share method (Gaillard & Goude, 2016) for aggregation.
FS:Fixed Share method (Gaillard & Goude, 2016) for aggregation over all the models in $\mathbb{P}$.
OEP-ROC-EW: Variant of **OEP-ROC** that uses Exponential Weighting (Gaillard & Goude, 2016) for aggregation.
EW: Exponential Weighting (Gaillard & Goude, 2016) aggregation over all the models in $\mathbb{P}$.
OEP-ROC-MLPOL: Variant of **OEP-ROC** that uses Polynomial Potential aggregation rule with different learning rates for each base model (Wintenberger, 2017) for aggregation.
MLPOL: Polynomial Potential aggregation rule (Wintenberger, 2017) for aggregation over all the models in $\mathbb{P}$.

4.3 Results

Table 4 presents the average ranks and their deviation for **OEP-ROC** and its variants and SoA methods for time series forecasting and online ensemble pruning. For the paired comparison, we compare our method **OEP-ROC** against each of the other methods. We counted wins and losses for each dataset using the RMSE scores. We use the non-parametric Wilcoxon Signed Rank test to compute significant wins and losses, which are presented in parenthesis (significance level 0.05).

Table 4 Comparison of **OEP-ROC** with $k = 15$ to different SoA for 100 time series. The rank column presents the average rank and its standard deviation across different time series. A average rank of 1 means the model was the best performing on all time series

Method	Our Method		Avg. Rank
	Looses	Wins	
OS-PGSM	11(3)	89(71)	24.76 ± 3.11
Ran-Pr-5	20(5)	80(69)	18.35 ± 4.93
Ran-Pr-10	25(6)	75(68)	15.91 ± 5.03
Ran-Pr- 15	31(3)	69(61)	12.82 ± 6.17
CNN	14(7)	86(66)	20.32 ± 6.90
ETS	11(4)	89(71)	24.98 ± 9.36
Ran-Pr-20	32(16)	68(58)	10.42 ± 6.17
ARIMA	16(7)	84(73)	14.91 ± 9.69
OTOP	25(5)	75(60)	15.32 ± 9.89
OEP-ROC-PER	27(9)	73(61)	10.07 ± 5.84
OS-PGSM-Int	17(8a)	83(66)	23.30 ± 4.84
DEMSC-K	19(7)	81(60)	15.52 ± 7.45
CNN-LSTM	26(12)	74(62)	12.63 ± 7.55
DEMSC-C	32(19)	68(55)	13.68 ± 7.66
OEP-ROC-I	33(7)	67(35)	8.98 ± 5.69
OEP-ROC-II	30(7)	70(42)	10.08 ± 5.83
LSTM	43(30)	57(49)	8.88 ± 8.41
OEP-ROC-15-topm-8	33(15)	67(36)	5.59 ± 4.53
NCL	49(25)	51(26)	9.07 ± 9.28
OEP-ROC-15-topm-6	33(11)	67(33)	6.25 ± 5.19
Ens	47(27)	53(33)	7.73 ± 3.34
OEP-ROC-15-topm-10	39(10)	61(35)	4.90 ± 4.55
OEP-ROC-TOP	38(13)	62(33)	4.97 ± 5.84
OCL	36(17)	74(61)	6.17 ± 6.66
OEP-ROC-C	39(13)	61(33)	3.79 ± 3.42
OEP-ROC-ST	37(10)	63(34)	6.31 ± 4.83
OEP-ROC	–	–	3.29 ± 3.08

Figure 1 represents the average rank, and respective standard deviation, of **OEP-ROC** and its variants, state of the art approaches for ensemble pruning, and other typical forecasting baselines.

Table 5 presents the average ranks and their deviation for **OEP-ROC** and its variants.

Table 6 shows the average rank, and respective standard deviation, of different aggregations methods taking as input at one time the pruned models by **OEP-ROC** and all the base models in $\mathbb{P}$ at a second time.

We compare also the run-time of **OEP-ROC** and its variants against the most competitive SoA method, DESMC-C, in Table 7.

Figure 3 shows a comparison between the current input time series pattern X_t^L (left part in black) with the average RoC (See Eq. 10) of the pruned ensemble to perform the forecast. Finally, a more general overview over the RoCs of the models selected by **OEP-ROC** compared to models selected by random pruning is shown in Fig. 2.

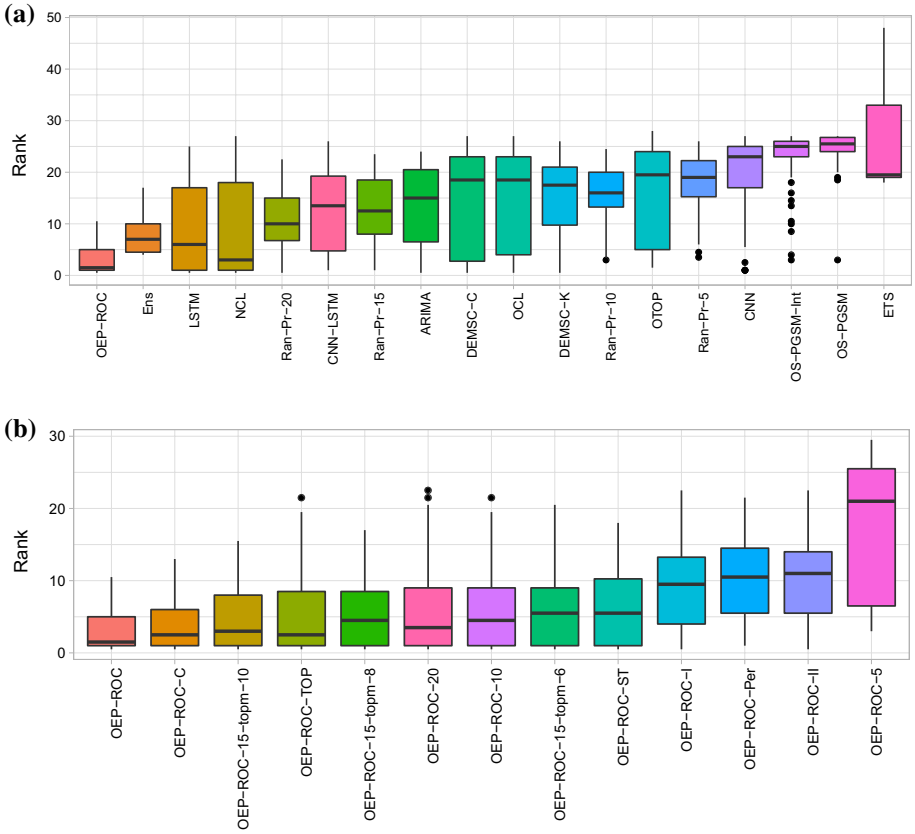

Fig. 1 Distribution of the ranks of **OEP-ROC** with $k = 15$ in comparison to (a) different SoA methods across the different time series and (b) with different variants of **OEP-ROC**

Table 5 Comparison of **OEP-ROC** with $k = 15$ to its variants for 100 time series. The rank column presents the average rank and its standard deviation across different time series

Method	Avg. rank
OEP-ROC-II	9.98 ± 5.91
OEP-ROC-I	8.88 ± 5.76
OEP-ROC-TOP	4.86 ± 5.55
OEP-ROC-PER	9.96 ± 5.91
OEP-ROC-15-topm-8	5.84 ± 4.61
OEP-ROC-5	6.50 ± 6.28
OEP-ROC-15-topm-6	6.14 ± 5.27
OEP-ROC-15-topm-10	4.79 ± 4.62
OEP-ROC-10	5.73 ± 5.87
OEP-ROC-20	5.74 ± 5.27
OEP-ROC-C	3.68 ± 3.49
OEP-ROC-ST	6.19 ± 4.91
OEP-ROC	3.18 ± 3.15

A average rank of 1 means the model was the best performing on all time series

Table 6 Comparison of **OEP-ROC** with $k = 15$ combined with different aggregation methods for 100 time series

Method	Avg. rank
SW	7.10 ± 2.54
OEP-ROC-SW	6.21 ± 3.95
EWA	6.83 ± 2.78
OEP-ROC-EWA	5.25 ± 2.24
OGD	4.81 ± 1.52
OEP-ROC-OGD	4.75 ± 1.75
FS	5.11 ± 2.75
OEP-ROC-FS	4.61 ± 2.16
MLPOL	3.85 ± 1.71
OEP-ROC-MLPOL	3.11 ± 2.05

Table 7 Average runtime plus variance (both in seconds) for three variants of **OEP-ROC** over 5 datasets

Name	Mean runtime (in seconds)	Variance of runtime (in seconds)
OEP-ROC	14.41	13.03
OEP-ROC-ST	0.49	0.24
OEP-ROC-Per	27.61	0.66

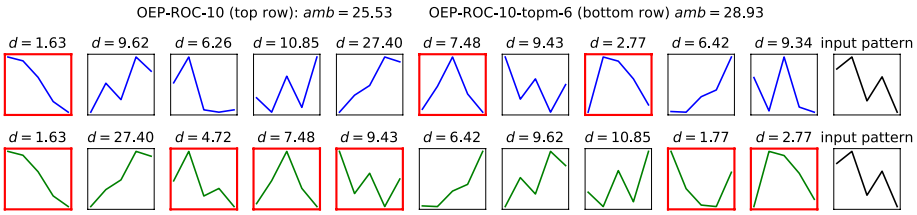

Fig. 2 Comparison of the clusters' representatives of **OEP-ROC-10** (top row) with $k = 10$ and **OEP-ROC-10-topm-6** (low row) (also $k = 10$) on the Abnormal Heartbeat dataset. We report the ambiguity *amb* of the clustered ensemble as well as the euclidean distance of each RoC to the input pattern, which is shown in the right most column. In red, we visualize the models that where chosen by each method for prediction

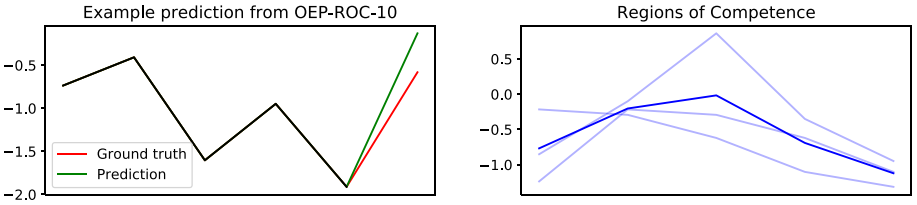

Fig. 3 **Left**: For the time series to predict (black), we show the ground truth value (red) as well as the prediction of **OEP-ROC-10** (green). **Right**: Visualization of each nearest RoC from the selected models (light blue), as well as the mean RoC (dark blue). Both plots were generated on the Abnormal Heartbeat dataset

4.3.1 Comparing OEP-ROC to the state of the art approaches

In the results in Table 4, **OEP-ROC** outperforms almost all the baseline methods in terms of ranks and wins/loses in pairwise comparison. In addition, it is clear from Fig. 1 that the first six best performing methods are variants of **OEP-ROC**. Ens, OCL and NCL seem to have quite good performance. However, their average ranks are approximately more than double the average rank of **OEP-ROC**.

The online ensemble pruning methods, e.g., OCL, OTOP, DEMSC-K and DEMSC-C, show inferior performance compared to **OEP-ROC**. ARIMA, ETS, and CNN, SoA methods for forecasting, are considerably worse in average rank compared to **OEP-ROC**. LSTM and CNN-LSTM show better performance, but are still worse than **OEP-ROC**. The ensemble Ens that uses all 33 models has also worse average rank comparing to **OEP-ROC** which uses on average only 6 base models which is almost a fifth of the pool $\mathcal{P}$ size. These results address research question **Q1**.

4.3.2 Comparing OEP-ROC to its variants

It can be seen from Table 5 that none of the pruning stages on its own (i.e. **OEP-ROC-C** and **OEP-ROC-TOP**) is able to achieve good results similar to **OEP-ROC**, which shows the importance of each stage. It is also clear that **OEP-ROC-C** has better performance than **OEP-ROC-TOP** since proceeding by only ranking the base models according to their closeness to the current pattern kills the diversity (i.e. selection is made in favor of RoCs that are most similar to the current pattern and as a result more or less similar to each other). In this way, the ensemble ambiguity is decreased even though the averaged error of its base models is decreased. This shows that our two-staged procedure helps to establish a trade-off between ensemble's diversity and accuracy. This answers research question **Q2**.

It can also be seen from Table 5 that none of the drift detection mechanisms (i.e. **OEP-ROC-I** and **OEP-ROC-II**) is able on its own to achieve good performance. The combination of both is necessary to update the pruning in the right moment when it is needed. It is also clear that performing the updates periodically in a random blind manner with **OEP-ROC-Per** does not necessarily improves the performance even though it is designed to trigger more updates than all the drift-aware methods. This demonstrates that informed adaption is always beneficial. This answers research question **Q3**.

4.3.3 Optimal choice of *k* and *top-M* size

The right set-up of the number of clusters k to be computed seems also to be a important factor for the performance. While it can be seen from Table 5 that low values of k like 5 do not help to achieve good performance, increasing the value of k seems to improve largely the rank of **OEP-ROC** (i.e. decreasing the rank which means better performance across all the data sets). This can be mainly explained by the fact that small number of clusters would lead to bigger clusters' sizes which means that the selection of the clusters' representatives will no longer be representative of the same error level by all the base models belonging to the same cluster, so the control over the average error of the base models is lost. Increasing the value of k too much is also not desired since it would lead to small cluster size and

more similar cluster representatives which may alter the diversity by decreasing the ambiguity. This answers research question **Q4**.

Finally, Table 5 shows also the usefulness and the benefits of our theoretical insights in setting up the size of ***top-M*** automatically. This size is set up by the derived bound (See Eq. 13) such that the base models' average error is reduced and the ambiguity is increased. Fixed values for M could not achieve the same performance as **OEP-ROC**. The best fixed model selection configuration is **OEP-ROC-15-topm-10** which has a bigger average rank than **OEP-ROC**. This addresses research question **Q5**.

4.3.4 Combination OEP-ROC with different aggregation methods

It can be seen from Table 6 that all aggregation methods achieve better results when combined with **OEP-ROC** than using all the based models in $\mathcal{P}$. The advantage in performance is clearly seen especially for SW, EWA and MLPOL. This can be explained by the fact that the dimension of the input for the ensemble's weights learning is reduced from $N = 33$ to an average of $|\boldsymbol{top} - \boldsymbol{M}| = 6$. This make the meta-learning task in EWA and MLPOL easier. In addition, the weighting schema is focused more around the most suitable base models in terms of accuracy and diversity. The difference in performance between the methods can be explained by the difference in the weights learning paradigm behind each method. This addresses research question **Q6**.

4.3.5 Scalability analysis

To compare scalability between our configurations, we considered **OEP-ROC**, **OEP-ROC-Per** and **OEP-ROC-ST**, because these configurations nicely illustrate which steps of our algorithm are most costly. We show the results in Table 7. As can be seen, **OEP-ROC-ST** is by far the fastest method on average, since it does not adapt its RoCs during the algorithms runtime. We noticed that recreating the RoCs takes by far the longest time in comparison to other steps of the algorithm and we plan to address this in future work. **OEP-ROC-Per** illustrates this problem the best, since it blindly and frequently recreates the RoCs and reclusters the models. Thus, its runtime is always high, no matter if an adaption to new time series properties is necessary or not. **OEP-ROC** strikes a balance between these two extremes and detects whether or not an adaption to a drift is necessary. As can be seen from the table, this results in a high variance of the runtime, since some datasets contain more concept drifts than others. We see this behaviour as a benefit, since **OEP-ROC** outperforms the other two methods we measured its runtime against, indicating that sometimes a higher runtime can be justified by overall better performance.

4.3.6 Explainability aspects

We provide some insights into how **OEP-ROC** can be used to provide suitable explanations for the reason behind specific base models selection to construct the ensemble at a specific time instant of interval. First, we compare the clusters' representatives of **OEP-ROC-10** (top row) and **OEP-ROC-10-topm-6** in Fig. 2. The current input pattern is shown in black on the right side. It can be clearly seen that **OEP-ROC** selects the RoCs that are quite different from each other without being too far from the current pattern which shows the trade-off established by **OEP-ROC** between accuracy and diversity.

OEP-ROC-10-topm-6 is promoting the selection of diverse pattern. However, the fixed size results in too many uninformative models (i.e models outside the L-sphere; See Sect. 3.4 for further details) being picked, leading to higher averaged error. This addresses research question **Q8**.

Figure 3 shows a comparison between the current input time series pattern X_t^L (left part in black) with the RoCs of the pruned ensemble to perform the forecast on the left. A clear similarity between the trend in both patterns can be observed which justifies the choice of this ensemble construction since it has been proven to show some degree of competence in forecasting using patterns with similar trend as input. This is further validated when also comparing between the true time series value (ground truth, red) and the predicted value (green). While these two values differ slightly, an evaluation of all the candidates in this point showed that the ensemble by **OEP-ROC** has the smallest error. This addresses research question **Q9**.

4.4 Discussion and future work

The empirical results indicate that **OEP-ROC** has performance advantages compared to popular forecasting methods and most SoA approaches for online ensemble pruning. We show that our method using RoCs-based pruning in two well-studied stages is able to gain excellent and reliable empirical performance in our setting. The informed adaption and update of the pruning decision following concept drift detection in both time series and base models' performance makes our method in addition to better predictive performance, computationally cheaper than the most competitive SoA. **OEP-ROC** can also be used successfully for providing useful explanations behind the selection of a specific subset of base models to compose the ensemble at a given time instant or interval. As future work, we plan to investigate the impact of varying some parameters in our setting, more specifically the size of the input pattern L. More candidate models from different families of machine learning models can also be considered by devising specific local attribution method to each family of models instead of being restricted to CNNs with gradient-based saliency maps as local attribution method to compute the RoCs. Explainability can further be promoted by making the base models more explainable or supported by their specific explanation tools. To improve the runtime of our methods further, we plan to investigate the impact of sampling RoCs from the validation dataset instead of computing them for the entire validation set every time.

5 Conclusion

This paper introduces **OEP-ROC**: a novel, practically useful online ensemble of DNNs two-staged pruning method using performance saliency maps-based RoCs for time series forecasting. The pruning is updated online in an informed-manner using concept drift detection in both time series and ensemble base models' performance. The two pruning stages in addition to the size of the resulting ensemble are supported by theory. An exhaustive empirical evaluation, including 100 real-world datasets and multiple comparison algorithms showed the advantages of **OEP-ROC** in terms of performance and scalability.

Acknowledgements 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–S18038A).

Author contributions All authors stated consent to this publication and contributed according to the order presented at the beginning of the paper, with the exception that AS and MJ contributed equally.

Funding Open Access funding enabled and organized by Projekt DEAL. Amal Saadallah and Katharina Morik receive funding by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 (DFG project number 124020371). Matthias Jakobs and Katharina Morik receive funding by the Federal Ministry of Education and Research of Germany (BMBF) as part of the Competence Center for Machine Learning (ML2R) (1–S18038A).

Availability of data and material All data presented in this work is either our own or cited correctly. The graphics used can be found in https://github.com/MatthiasJakobs/os-pgsm/tree/ecml_journal_2022.

Code availability All code (and a guide on how to use it) is available at https://github.com/MatthiasJakobs/os-pgsm/tree/ecml_journal_2022.

Declarations

Conflict of interest Not applicable.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery, 31,* 606–660.

Banfield, R. E., Hall, L. O., Bowyer, K. W., & Kegelmeyer, W. P. (2005). Ensemble diversity measures and their application to thinning. *Information Fusion, 6*(1), 49–62.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. Wiley.

Breiman, L. (1996). Bagging predictors. *Machine learning, 24*(2), 123–140.

Brown, G., Wyatt, J. L., & Tiňo, P. (2005). Managing diversity in regression ensembles. *Journal of Machine Learning Research, 6*(2), 1621–1650.

Burkardt, J. (2009). K-means clustering. In Advanced research computing, interdisciplinary center for applied mathematics. Virginia Tech.

Camburu, O.-M. (2020). Explaining deep neural networks. arXiv preprint arXiv:2010.01496.

Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on machine learning*, p. 18

Cerqueira, V., Torgo, L., Pinto, F., & Soares, C. (2017). Arbitrated ensemble for time series forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 478–494. Springer

Cerqueira, V., Torgo, L., Pinto, F., & Soares, C. (2018). Arbitrage of forecasting experts. *Machine Learning, 108,* 913.

Cheng, X., Rao, Z., Chen, Y., & Zhang, Q. (2020). Explaining knowledge distillation by quantifying the knowledge. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12925–12935.

Chen, H., Tiňo, P., & Yao, X. (2009). Predictive ensemble pruning by expectation propagation. *IEEE Transactions on Knowledge and Data Engineering, 21*(7), 999–1013.

Dua, D., & Graff, C. (2017). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

Frosst, N., & Hinton, G. (2017). Distilling a neural network into a soft decision tree. arXiv preprint arXiv:1711.09784

Gaillard, P., & Goude, Y. (2016). Opera: Online prediction by expert aggregation. R package version 1.0. https://CRAN.R-project.org/package=opera

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR), 46*(4), 1–37.

Gamboa, J.C.B. (2017). Deep learning for time-series analysis. arXiv preprint arXiv:1701.01887

Gers, F.A., Eck, D., & Schmidhuber, J. (2002). Applying lstm to time series predictable through time-window approaches. In: Neural Nets WIRN Vietri-01, pp. 193–200. Springer.

Giacinto, G., Roli, F., & Fumera, G. (2000). Design of effective multiple classifier systems by clustering of classifiers. In *Proceedings 15th international conference on pattern recognition*. ICPR-2000, vol. 2, pp. 160–163. IEEE.

Godahewa, R., Bergmeir, C., Webb, G.I., Hyndman, R.J., & Montero-Manso, P. (2021). Monash time series forecasting archive. In *Neural information processing systems track on datasets and benchmarks*. forthcoming

Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. In: The Collected Works of Wassily Hoeffding, pp. 409–426. Springer.

Hoseinzade, E., & Haratizadeh, S. (2019). Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications, 129,* 273–285. https://doi.org/10.1016/j.eswa.2019.03.029.

Jain, G., & Mallick, B. (2017). A study of time series models arima and ets. Available at SSRN 2898968.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., & Viegas, F., et al. (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pp. 2668–2677. PMLR

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion, 37,* 132–156.

Krikunov, A. V., & Kovalchuk, S. V. (2015). Dynamic selection of ensemble members in multi-model hydrometeorological ensemble forecasting. *Procedia Computer Science, 66,* 220–227.

Küsters, F., Schichtel, P., Ahmed, S., & Dengel, A. (2020). Conceptual explanations of neural network prediction for time series. In *2020 International joint conference on neural networks (IJCNN)*, pp. 1–6. IEEE.

Lamy, J.-B., Sekar, B., Guezennec, G., Bouaud, J., & Séroussi, B. (2019). Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach. *Artificial Intelligence in Medicine, 94,* 42–53.

Lazarevic, A., & Obradovic, Z. (2001). Effective pruning of neural network classifier ensembles. In: IJCNN'01. international joint conference on neural networks. Proceedings (Cat. No. 01CH37222), vol. 2, pp. 796–801. IEEE.

Livieris, I. E., Pintelas, E., & Pintelas, P. (2020). A cnn-lstm model for gold price time-series forecasting. *Neural Computing and Applications, 32,* 17351.

Li, N., Yu, Y., & Zhou, Z.-H. (2012). Diversity regularized ensemble pruning. In P. A. Flach, T. De Bie, & N. Cristianini (Eds.), *Machine learning and knowledge discovery in databases* (pp. 330–345). Springer.

Ma, Z., Dai, Q., & Liu, N. (2015). Several novel evaluation measures for rank-based ensemble pruning with applications to time series prediction. *Expert Systems with Applications, 42*(1), 280–292.

Margineantu, D.D., & Dietterich, T.G. (1997). Pruning adaptive boosting. In: ICML, vol. 97, pp. 211–218. Citeseer

Martınez-Munoz, G., & Suárez, A. (2004). Aggregation ordering in bagging. In: Proc. of the IASTED International Conference on Artificial Intelligence and Applications, pp. 258–263. Citeseer

Martinez-Munoz, G., Hernández-Lobato, D., & Suárez, A. (2008). An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(2), 245–259.

Molnar, C. (2020). Interpretable Machine Learning. Lulu.com.

Mozaffari, A., & Azad, N. L. (2014). Optimally pruned extreme learning machine with ensemble of regularization techniques and negative correlation penalty applied to automotive engine coldstart hydrocarbon emission identification. *Neurocomputing, 131,* 143–156.
Olah, C., Mordvintsev, A., & Schubert, L. (2017). Feature visualization. *Distill, 2*(11), 7.
Partalas, I., Tsoumakas, G., Katakis, I., & Vlahavas, I. (2006). Ensemble pruning using reinforcement learning. In hellenic conference on artificial intelligence, pp. 301–310. Springer
Partalas, I., Tsoumakas, G., & Vlahavas, I. (2012). *A study on greedy algorithms for ensemble pruning*. Thessaloniki, Greece: Aristotle University of Thessaloniki.
Ribeiro, M.T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM sigkdd international conference on knowledge discovery and data mining*. pp. 1135–1144.
Romeu, P., Zamora-Martínez, F., Botella-Rocamora, P., & Pardo, J. (2013). Time-series forecasting of indoor temperature using pre-trained deep neural networks. In *International Conference on Artificial Neural Networks*, pp. 451–458. Springer
Saadallah, A., & Morik, K. (2021). Online ensemble aggregation using deep reinforcement learning for time series forecasting. In 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA). IEEE
Saadallah, A., Jakobs, M., & Morik, K. (2021). Explainable online deep neural network selection using adaptive saliency maps for time series forecasting. In: Oliver, N., Pérez-Cruz, F., Kramer, S., Read, J., Lozano, J.A. (eds.) Machine Learning and Knowledge Discovery in Databases. Research Track, pp. 404–420. Springer, Cham.
Saadallah, A., Priebe, F., & Morik, K. (2019). 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
Saadallah, A., Tavakol, M., & Morik, K. (2021). An actor-critic ensemble aggregation model for time-series forecasting. In *IEEE International Conference on Data Engineering (ICDE)*
Saadallah, A., Moreira-Matias, L., Sousa, R., Khiari, J., Jenelius, E., & Gama, J. (2018). Bright-drift-aware demand predictions for taxi networks. *IEEE Transactions on Knowledge and Data Engineering, 32,* 234.
Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., & Müller, K.-R. (2021). Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE, 109*(3), 247–278.
Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.
Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.
Stolpe, M., Bhaduri, K., & Das, K. (2016). Distributed support vector machines: An overview. Solving large scale learning tasks. Challenges and Algorithms, 109–138.
Street, W.N., & Kim, Y. (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 377–382
Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert Systems with Applications, 39*(8), 7067–7083.
Tjoa, E., & Guan, C. (2020). A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE Transactions on Neural Networks and Learning Systems, 32*(11), 4793–4813.
Tsoumakas, G., Partalas, I., & Vlahavas, I. (2009). An ensemble pruning primer. In: Applications of supervised and unsupervised ensemble methods, pp. 1–13. Springer.
Utgoff, P. E., & Stracuzzi, D. J. (2002). Many-layered learning. *Neural Computation, 14*(10), 2497–2529.
Wintenberger, O. (2017). Optimal learning with bernstein online aggregation. *Machine Learning, 106*(1), 119–141.
Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation, 8*(7), 1341–1390.
Yu, Y., Li, Y.-F., & Zhou, Z.-H. (2011). Diversity regularized machine. In *Twenty-second international joint conference on artificial intelligence*.
Zablocki, É., Ben-Younes, H., Pérez, P., & Cord, M. (2021). Explainability of vision-based autonomous driving systems: Review and challenges. arXiv preprint arXiv:2101.05307
Zhang, T. (2002). Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research, 2,* 527–550.

Zhang, Y., Burer, S., Nick Street, W., Bennett, K. P., & Parrado-Hernández, E. (2006). Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research, 7*(7), 1315.

Zhang, S., Chen, Y., Zhang, W., & Feng, R. (2021). A novel ensemble deep learning model with dynamic error correction and multi-objective ensemble pruning for time series forecasting. *Information Sciences, 544,* 427–445.

Zhang, J., Dai, Q., & Yao, C. (2021). Dep-tsp meta: A multiple criteria dynamic ensemble pruning technique ad-hoc for time series prediction. *International Journal of Machine Learning and Cybernetics, 12,* 2213.

Zhou, Z.-H., & Tang, W. (2003). Selective ensemble of decision trees. In: International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-soft Computing, pp. 476–483. Springer

Zhou, Z.-H., Wu, J., & Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artificial intelligence, 137*(1–2), 239–263.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (icml-03)*, pp. 928–936.