Explainable Online Deep Neural Network Selection using Adaptive Saliency Maps for Time Series Forecasting[⋆]

Amal Saadallah, Matthias Jakobs, and Katharina Morik

Artificial Intelligence Group
Department of Computer Science, TU Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. Deep neural networks such as Convolutional Neural Networks (CNNs) have been successfully applied to a wide variety of tasks, including time series forecasting. In this paper, we propose a novel approach for online deep CNN selection using saliency maps in the task of time series forecasting. We start with an arbitrarily set of different CNN forecasters with various architectures. Then, we outline a gradient-based technique for generating saliency maps with a coherent design to make it able to specialize the CNN forecasters across different regions in the input time series using a performance-based ranking. In this framework, the selection of the adequate model is performed in an online fashion and the computation of saliency maps responsible for the model selection is achieved adaptively following drift detection in the time series. In addition, the saliency maps can be exploited to provide suitable explanations for the reason behind selecting a specific model at a certain time interval or instant. 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.

Keywords: Deep Neural Networks · Time Series Forecasting · Model Selection · Grad-CAM· Explainability.

1 Introduction

Both the complex and time-evolving nature of time series make forecasting one of the most challenging tasks in time series analysis [26].

Several machine learning methods have been proposed to solve this task either by dealing with the data as ordered sequences of observations in an online or a streaming manner, or by using time series embeddings which map a set of target observations to a k-dimensional feature space corresponding

⋆ 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).

to the k past lagged values of the observation [8, 26]. In particular, Artificial Neural Networks (ANNs) have been widely applied to solve the forecasting task [24, 28]. Nowadays, deep ANNs (DNNs) have shown some improvements over previous shallow ANN architectures [24]. In fact, DNNs have shown the ability to automatically learn new, complex and enriched feature representation from input data [29], thus achieving good performance in solving a wide variety of task. Recurrent-based NNs such as Long Short-Term Memory Networks (LSTMs), as well as Convolutional Neural Networks (CNNs), have been widely used as state-of-the-art NN methods in the context of forecasting [24, 13]. Many improvements over these network architectures have been proposed in literature, ranging from optimizing the architecture structure to combining these networks together in one single forecasting task [13, 20, 19]. However, it is generally accepted that none of the proposed machine learning forecasting methods is universally valid for every application, and even within the same application, models have varying relative performance over time [26, 25, 9, 22]. Hence, different forecasting models have different areas of expertise and a varying relative performance [9, 8, 22]. Therefore, adequate and adaptive model selection in real-time is required to cope with the time evolving nature of time series and the fact that models have certain expected level of expertise in predicting a given sequence in the time series. While some works focused on online single model selection, others have been based on the assumption that no single model is expert the whole time and suggested to combine several single models in an ensemble framework by adaptively combining single models into one [26, 25, 8, 9]. Given a set of candidate models for performing a well-defined forecasting task, different tactics ranging from statistical estimations to applying meta-learning to learning the adequate selection strategy have been suggested. The approaches for single model selection can be divided into three main families. The first family of methods is based on approximating a posterior over the expected error of the different candidates using parametric [6] or non-parametric estimation methods [2]. These methods are not practical in the context of forecasting since continuous composite densities for the error function of the target and estimated time series values have to be approximated. The results depends largely on the quality of approximation. The second family consists of using empirical estimation of the unseen error of a given model using a independent validation / calibration dataset. Models with lowest estimated error are selected subsequently [23]. These methods are quite ineffective in practice since the estimated empirical error is usually lower than the true error. The third family is based on the meta-learning paradigm, where the selection of the adequate method is decided by another machine learning model which learns from previous selection realizations characterized by a set of devised meta features [8, 26].

Meta-learning can also be used for model selection by specializing the set of candidate models over different parts of the input so that each part gets assigned to one expert model based on comparison of predicted candidate model performances [8, 9]. These parts are called **Region of Competence** (RoC) of a model [22]. The RoCs of one model are either stored individually or clustered and cluster centres are stored. At test time, the distance of the current input (i.e. in our case time series input sequence) to the RoCs or the RoCs cluster centres are computed, selecting the model with the lowest distance to perform the prediction. Generally, the computation of the RoCs is done by considering a static division of the time series into equally sized intervals. One way is to consider each time series observation t and its corresponding k lagged values as one interval and sliding the time window by one time step, whereas another way is to split the training or validation set into equally sized intervals [22].

Generally, the selection is performed in a static manner [25], i.e. the decision is made once at a time in favour of one model and this model is used subsequently to forecast all the required values at test time. The selection can also be updated continuously (i.e. blindly at each time instant or periodically) [8, 9]. However, this is usually expensive in terms of time and resources [26], especially when the candidate models include DNNs. Few works in the literature performed the selection of single or ensemble models in an informed manner following drift detection of the relative performance of candidate models [26, 25].

Candidate models in model selection can be the result of considering different parameter settings of the same model or by training different models belonging to different families of models. In the former case, the first family of model selection methods is widely used [6, 2], while in the latter, a wide variety of selection approaches have been proposed [23, 8, 26]. However, the search for optimal network architectures for a given application is still an open research question [15]. This is even more challenging in the case of forecasting, where the decision for the adequate architecture have to be made in real-time. Due to their high training run-time and general resource consumption it is usually impractical to search for the adequate architecture at test time at each time instant or even in a periodic manner. Therefore, we focus in this work on approaching this problem by considering different candidate models of DNNs from different architectures (i.e. based on CNNs combined with other NNs models) and we perform the selection of the adequate architecture in real-time in an adaptive informed manner using concept drift detection in the time series.

We start by computing the RoCs of candidate CNNs using saliency maps. Saliency maps are usually used to establish a relationship between the output and the input of a neural net given fixed weights. They are widely used in the context of computer vision with CNNs to create a class-specific heatmap based on a particular input image and a chosen class of interest [32]. These maps are used for visualizing the regions of input that are "important" for prediction by the model and for understanding a model's prediction [27]. We suggest not only to transfer the class-activation maps from the context of classification to forecasting, but also to establish a mapping between the input time series and the performance so that dynamic RoCs are computed for each single CNN. Opposingly to the aforementioned approaches, the RoCs are considered as dynamic since their size is automatically decided and changed over time by the saliency map depending on the input time series sequence and the CNN performance. The RoCs are computed using a time-sliding window over a validation set. At test time, we produce forecasts step by step. At each time step, the distance between the recent observed window of time series observations (i.e. lagged values used to compute the forecast) and the pre-computed RoCs is determined. The model corresponding to the RoC with the lowest distance is selected to perform the forecasting. Additionally, the pre-computed RoCs are adaptively updated in case a concept drift is detected in the time series by sliding the validation set to take into account the probable presence of new concepts in the data when computing RoCs. The saliency maps can also be exploited to provide explanations for the reason behind selection one particular model for a given sequence of input data.

We further conduct comprehensive empirical analysis to validate our framework using 102 real-world time series datasets from various domains. The obtained results demonstrate that our method achieves excellent results in comparison to the SoA approaches for DNN selection 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 CNNs selection for time series forecasting by computing RoCs for a set of candidate CNN-based models using an adaption of saliency maps.
– We update the RoCs in an informed manner following concept drift detection in the time series data.
– We exploit the saliency maps to provide suitable explanations for the reason behind selecting a specific model at a certain time instant or interval.
– We provide a comparative empirical study with state-of-the-art methods, and discuss their implications in terms of predictive performance and scalability.

2 Literature review

Over the recent years, deep learning methods have been successfully applied in a wide variety of real-world learning tasks, including time series forecasting [20, 10, 19]. Currently, Recurrent Neural Networks (RNNs), and particularly Long-Short Term Memory (LSTM) nets, are considered to be the state-of-the-art in time series forecasting [7, 19, 20]. Thanks to their design based on recurrent connections, these networks have the ability to learn from the entire history of previous time series values. Another alternative for the use of DNNs in the forecasting task is to employ a Convolutional Neural Network (CNN) with multiple layers of dilated convolutions [31]. The layered structure of CNNs enables them to work well on noisy series, by removing the noise within each subsequent layer and extracting only the meaningful patterns, performing thus similarly to neural networks which use wavelet transform on input time series [7]. This also allows for the receptive field of the network to expand exponentially, hereby making the network, similarly to RNNs, access a wide range of historical data. Some works have focused on improving CNN based architectures by combining CNN and

[1] `https://github.com/MatthiasJakobs/os-pgsm/tree/ecml2021`

LSTM in one single model to take advantage of the ability of LSTMs to cope with long temporal correlations [20, 19]. In [21], the authors propose an undecimated convolutional network for time series forecasting using the undecimated wavelet transform. An autoregressive weighting schema for forecasting financial time series is presented in [5] where the weights are learnt through a CNN. However, convolutional architectures in literature are much more commonly applied to time series classification problems compared to forecasting [7, 11].

The aforementioned works have focused on searching the most suitable network architecture for a well-defined application. At test time, the architecture and the learned weights are kept fixed and used to produce forecasts. However, to cope with the time-evolving nature of time series data, the forecasting schema has to be designed in a dynamic adaptive manner [25, 26]. Since the same model can't be guaranteed to hold the same performance over time [25, 26, 8, 9, 22], online adequate model selection is required. This is usually hard to achieve with DNNs in general since the architecture tuning and the re-training of such models are intensively time consuming operations [7, 20]. We suggest to mitigate this problem by training different CNN-based models with various architectures offline and decide for the online selection of the adequate network at each time instant at test time. The selection is achieved using a computation of the so-called RoCs using saliency maps (i.e. known also as attribution heat-maps) [32].

Saliency maps (in the form of *class activation maps (CAMs)*) were originally designed for computer vision classification tasks to visualize which part of an image is of high relevance to the network to make its decision [32, 27]. They are considered tools for better understanding a models behaviour, e.g. by providing insight into model failure modes [32]. Many saliency map generation methods are post-hoc methods, in the sense that they are applied to an already-trained model. CAM uses the feature maps produced by the last convolutional layer of a CNN. This is motivated by the fact that the last convolution layer is expected to contain both high-level semantic and detailed spatial information [27]. More recently, CAMs have been applied in the context of time series classification to explain which features and which joint contribution of all the features during which time interval are responsible for a given time series class [3]. However, to the best of our knowledge, ours is the first work to apply saliency maps for online CNN-based model selection for time series forecasting. The generation of these maps is performed in an informed adaptive fashion. In addition, they are exploited to provide explanation for particular timely model selection.

3 Methodology

This section introduces our method and its main stages. In a first stage, we train different candidate CNN-based models with various architectures offline. The second stage consists of determining the RoCs for these models using sliding windows over a validation set. The RoCs are computed using a modified version of saliency maps, in the sense that instead of using these maps as class activation maps (CAM), we employ them to establish a relation between a relative "good" performance of a given candidate CNN and a particular pattern within the input time-sliding window sequences. We base our method on a gradient-based technique for generating saliency maps called Grad-CAM [27]. We call our modified version in the following "Performance Gradient-based Saliency Map (PGSM)". In the third stage, in order to produce a forecast at a given time instant t_f, the distance of the current input time sequence (i.e. time series observations from t_{f-k} to t_{f-1}, $f > k$) to the computed RoCs of each model is measured. The model corresponding to RoC with the lowest distance is selected to forecast the time series value at t_f. A concept drift detection mechanism in the time series is employed at test time (i.e. during forecasting). Once a drift is detected, an alarm is triggered to update the validation set by taking into account the new observed time series values and to subsequently update the RoCs. The PGSMs can be used to provide suitable explanations for the reason behind selecting a specific model at a certain time interval or instant. Practical examples of explanations are shown with details in Section 4. Our framework is denoted in the rest of the paper, OS-PGSM: Online CNN-based models Selection using Performance Gradient-based Saliency Maps.

3.1 Preliminaries

A time series X is a temporal sequence of values, where $X_t = \{x_1, x_2, \cdots, x_t\}$ is a sequence of X until time t and x_i is the value of X at time i. Denote with $P_{CNN} = \{C_0, C_1, \cdots, C_{N-1}\}$ the pool of trained CNN-based models. Let $\hat{x} = (\hat{x}^{C_0}, \hat{x}^{C_2}, \cdots, \hat{x}^{C_{N-1}})$ be the vector of forecast values of X at time instant $t+f, f \geq 1$ (i.e. x_{t+f}) by each of the models in P_{CNN}. The goal of the dynamic online selection is to identify which $\hat{x}^{C_j}$ should be used to produce this forecast.

We divide the time series X_t into $X_\omega^{train} = \{x_1, x_2, \cdots, x_{t-\omega}\}$ and $X_\omega^{val} = \{x_{t-\omega+1}, x_{t-\omega+2}, \cdots, x_t\}$, with ω a provided window size. X_ω^{train} is used for training the models in P_{CNN} and X_ω^{val} 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 $C_j, j \in \{0, \cdots, N-1\}$ 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.

3.2 Candidate CNN Architectures

The candidate models are CNN-based models that share more less the same basic types of layers. The common basic structure consists of sequence of 1D-convolutional layers with different filter and kernel sizes, 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 size of filters and kernels) 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.

3.3 Online Model Selection

Performance Gradient-based Saliency Maps The PGSMs are inspired from the class activation saliency maps, more specifically, Grad-CAM [27]. This method has been proven to successfully pass commonly used sanity checks, which are devised to check whether the saliency map is truly providing insights into what the model is doing or not [1]. However, instead of using these maps to derive the importance of certain features for a given class, we use them to map the performance of a given forecasting model to a specific time interval. The performance of each model $C_j, j \in \{0, \cdots, N-1\}$ is evaluated using an error-related measure, namely the Mean Squared Error, ϵ_j^i on $X_{n_\omega}^{val,i}$: the i^{th} time interval window of X_ω^{val} of size n_ω. Our goal is to estimate the importance of each value in $X_{n_\omega}^{val,i}$ to the measured error ϵ_j^i of C_j. This can be interpreted similarly 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. However, we are focused here on the temporal information explaining certain behaviour/performance of C_j. To do so, the last 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 the ϵ_j^i with respect to A. Subsequently, a global average over all the units in A is computed:

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

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 a ReLU is used to remove all the negative contributions by:

$$L_j^i = ReLU(\sum_{f_{maps}} w^\epsilon A) \tag{2}$$

$L_j^i \in \mathbb{R}^U$ is used to find the regions in $X_{n_\omega}^{val,i}$ that have mainly contributed to ϵ_j^i of the network C_j. Note that the candidates are designed such that $U < n_\omega$.

RoCs Computation Our goal is to determine the region of competences of each model on X_ω^{val}. However, one single evaluation of the models on X_ω^{val} obviously lead to just one best model. Therefore, we need to split X_ω^{val} into equally sized time intervals of size n_ω, so that different evaluations of the candidate models are performed and different rankings are derived. To increase this number of evaluations, the intervals of size n_ω can be obtained using a time-sliding window approach over X_ω^{val} where the sliding operations are performed each z-steps. The lower z, the higher the number of evaluations is. If z is set to 1, the sliding window approach is performed in a step-wise manner. After evaluating the models on each of the $X_{n_\omega}^{val,i}$, the RoC of the model with the lowest error is computed using L^i_j of the PGSMs, where j the index of the candidate model C_j satisfying: $C_j = \text{argmin}_{z\in\{1,\cdots,N\}}\, \epsilon^i_z$.

To obtain one continuous region RoC R^j within the time series sequence $X^{val,i}_{n_\omega}$, a smoothing operation is applied to L^i_j. This is achieved by normalizing L^i_j values between 0 and 1 and applying a threshold $\tau = 0.5$ to filter out smaller values (i.e. these values are set to 0). Further smoothing using a moving-average of size 3, is applied where each point is compared to the previous and the subsequent value. Whenever R^j of model C_j is computed, it is added to a corresponding RoC^j buffer which includes all collected RoCs for the model C_j (i.e. since C_j can be the best performing model on different $X^{val,i}_{n_\omega}$).

Online Forecasting For forecasting the value of X at $t+f$ (assume $f = 1$ for simplicity), the candidate CNNs are devised such that they use the same k-lagged values of the time series as input, $p^k_t = \{x_{t-k+1}, \cdots, x_t\}$, $(t \geq k)$. To perform the selection, the distance of the input pattern p^k_t to the RoCs for each model in P_{CNN} is measured. The RoCs of a given model $C_j, j \in \{0, \cdots, N-1\}$ are already collected in $RoC^j = \{R^j_1, R^j_2, \cdots, R^j_{M^j}\}$, where M^j is the total number of regions of competence that have been determined by the PGSMs. Since the length of each RoC can be different from k (i.e. length of p^k_t), Dynamic Time Wrapping (DTW) [4] is used to measure the similarity between p^k_t and each $R^j_z, z \in \{1, \cdots, M^j\}$ within each $RoC^j, j \in \{0, \cdots, N-1\}$. The model C_b satisfying :

$$C_b = \underset{\substack{j\in\{0,\cdots,N-1\};\\ z\in\{1,\cdots,M^j\}}}{\text{argmin}} \; DTW(R^j_z, p^k_t) \tag{3}$$

is selected to forecast $t+1$.

RoCs Update As explained above, the ROCs are computed offline using the validation set X^{val}_ω. However, due to the dynamic behaviour of time series, streaming upcoming values can be subject to significant changes, more specifically to concept drifts [12]. As a result, the ROCs have to be updated to take into account the possible presence of new patterns after the occurrence of such drifts and also to gain knowledge of which models are more adequate to handle these patterns if they ever reoccur again (i.e. note that the already computed RoCs are preserved and enriched with the new ones). Once a drift is detected, an alarm is triggered to update of the ROCs by sliding X^{val}_ω 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 [26]: $\Delta m_{t_f} = \mathbb{E}(X_{t_f}) - \mu$, with $\mu = \mathbb{E}(X_t), t \leq t_f$, 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 [16], which states that after W independent observations of a real-value random variable with range r, its true mean has not diverged if the

```
Parameters: size of the validation set: $\omega$; size of time windows within the
 validation set: $n_\omega$; CNNs Pool: $P_{CNN}$.
 1: Models Training and RoCs Computation:
 2: Train each $C_j \in P_{CNN}, j \in \{0, \cdots, N-1\}$ on $X_\omega^{train}$.
 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:     Determine the best performing $C_j$.
 6:     Compute the corresponding $R_i^j$ using PGSMs (i.e. $L_j^i$ Eq. 2 )
 7:     Add $R_i^j$ to the corresponding buffer $RoC^j$
 8: end for
 9: Online Forecasting: Forecasting next $N_f$ values :
10: predict $x_{t+1}$ by the model $C_b$ selected using eq. 3
11: for $j \in \{2, \cdots, N_f\}$ do
12:     if an alarm is triggered (concept drift detected) then
13:         Update $X_\omega^{val} = \{x_{t-\omega+j}, x_{t-\omega+2}, \cdots, x_{t+j-1}\}$
14:         Recompute and add new RoCs (steps:4-7)
15:     end if
16:     predict $x_{t+j}$ by the model $C_b$ selected using eq. 3
17: end for
```

Algorithm 1: OS-PGSM

sample mean is contained within $\pm\xi_m$:

$$\xi_m = \sqrt{\frac{r^2 \ln(1/\delta)}{2W}} \tag{4}$$

with a probability of $1 - \delta$ (a user-defined hyperparameter). Once $|\Delta m_{t_f}|$ 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. All the steps of OS-PGSM are summarized in Algorithm 1.

4 Experiments

We present the experiments carried out to validate OS-PGSM and to answer these research questions: **Q1:** How does OS-PGSM perform compared to the state-of-the-art (SoA) and existing online model selection methods for time series forecasting?; **Q2:** What is the advantage of reducing the step size z for sliding the window of size n_ω over X_ω^{val} on the performance of OS-PGSM? **Q3:** What is the advantage of updating the RoCs in an informed fashion (i.e. following drift detection)?; **Q4:** How scalable is OS-PGSM in terms of computational resources compared to the most competitive online model selection methods? and what is the computational advantage of the **drift-aware** adaption of the models' RoCs?; **Q5:** How can OS-PGSM be exploited to provide suitable explanations for the reason behind selecting a specific model 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 using 50% for training (X_{ω}^{train}), and 25% for validation (X_{ω}^{val}) and 25% for testing. We use 102 real-world time series. A full list of the used datasets, together with a description, is given in the code repository[2].

4.2 OS-PGSM Setup and Baselines

We construct a pool P_{CNN} 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 Section 3.2. By combining these different parameters and adding/removing LSTM layer, 12 different CNNs with various architectures are created. OS-PGSM has also a number hyper-parameters that are summarized in Table 1. In our experiments, k is set equal to n_ω and the RoCs

Parameter	Description	Value
k	number of lagged values (size of the input to CNNs p_t^k)	5
ω	size of validation set	25% of the dataset length
n_ω	size of time windows within the validation set	5
z	number of time steps with which time windows within the validation set are slided	1
δ	Hoeffding-Bound parameter	0.05

Table 1: Hyperparameters of OS-PGSM and their values for the experiments.

(after computation and smoothing) result in even smaller size than k. However, this is not problematic since we are interested in extracting distinctive patterns that are responsible for good performance of a given candidate. The difference in lengths of the RoC and the input sequence (k) are handled by the DTW measure. We compare OS-PGSM against the following approaches which include SoA methods for forecasting and model selection methods devised in the context of forecasting. Some of them operate in an online fashion. First, we compare against **ARIMA** and Exponential Smoothing (**ETS**) [18]. Next, we add the two best performing candidate models, denoted as **CNN** and **CNN-LSTM** [24]. Additionally, a simple **LSTM** is added for comparison [14]. **KNN-RoC** [22] which computes static RoCs using complete intervals of the validation set as input and the rank of the individual candidates on each interval as labels for a *KNN* classifier, using DTW distance and $K = 3$, is also used for comparison. At test time, the *KNN* predicts which candidate should be selected. Next, we also compare ourselves against a simple validation procedure where the CNNs are evaluated offline and the best model is selected to forecast all the upcoming data points [23]. **ADE** [8, 9] was recently developed for online dynamic ensemble of forecasters construction. However, instead of selecting many models, we select

[2] `https://github.com/MatthiasJakobs/os-pgsm/tree/ecml2021`

the best performing model using the same principle. A Random Forest is used for estimating each candidate error and select the best model based on the lowest predicted error at test time. As **Stacking**, we denote a method where a meta-learner (Random Forest) is trained to predict which model to select using a set of meta-features consisting of input time sequence statistical characteristics and performance-based features [30]. Finally, we compare ourselves against **Adaptive Mixture** [17], which consists of some experts (Shallow CNNs) and a gating network. The gating network acts as a selector by performing a single output stochastic switch to select a given expert with the estimated switch probability.

We also compare OS-PGSM with some variants of itself. Note that OS-PGSM uses the Hoeffding-based drift detection mechanism to update the RoCs.

OS-PGSM-Int: Same as our method, but the time windows of size n_ω are slided with step size $z = n_\omega$.

OS-PGSM-Euc: Instead of using DTW as similarity measure, we use Euclidean distance. However, values in the RoC corresponding to 0 are not taken into consideration in the k-lagged values sequence (p_t^k).

OS-PGSM-Int-Euc: It is a combination of **OS-PGSM-Int** and **OS-PGSM-Euc**.

OS-PGSM-St: Same as our method, but the RoCs are not updated using the drift detection mechanism. The RoCs are computed and stored offline, only the selection takes place online.

OS-PGSM-Per: Same as our method, but the RoCs are update periodically in a blind manner (i.e. without taking into account the occurrence of concept drifts) with periodicity each upcoming 10% data points.

4.3 Results

Table 2 presents the average ranks and their deviation for all methods. For the paired comparison, we compare our method OS-PGSM 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).

In the results in Table 2, OS-PGSM outperforms the baseline methods in terms of wins/loses in pairwise comparison. The online model selection approaches, e.g., KNN-RoC, ADE-Single and Adaptive Mixture, show inferior performance compared to OS-PGSM. ARIMA, ETS, LSTM, and CNN, SoA methods for forecasting, are considerably worse in average rank compared to OS-PGSM. CNN-LSTM shows slightly better performance, but is still worse than OS-PGSM. The most competitive SoA approach to OS-PGSM is ADE-Single. Nevertheless, it has a higher average rank and a lower performance than all the variants of our method.These results address the research question **Q1**.

Comparing OS-PGSM to different variants of our method, we see a clear advantage in using all the choices in our method. First, the DTW distance is better in sketching the similarities between the input sequences and the RoCs, especially when both have different lengths as explained above. This explains

	ARIMA	ETS	LSTM	CNN	CNN -LSTM	Valid.	KNN -RoC	ADE -Single
Losses	7(6)	5(4)	17(8)	18(6)	23(6)	20(12)	22(16)	30(19)
Wins	95(93)	97(96)	85(83	84(80)	79(71)	82(72)	80(73)	72(61)
Avg. Rank	12.90	13.04	9.97	9.00	7.30	7.84	7.41	4.28
±	4.35	4.26	3.60	3.60	3.59	4.18	3.95	2.90

	Stacking	Adapt. Mixture	OS-PGSM Int	Euc	Int-Euc	St	Per	OS-PGSM
Losses	11(10)	17(8)	40(7)	35(10)	33(9)	49(10)	45(7)	-
Wins	91(80)	85(84)	62(54)	66(56)	69(66)	53(44)	57(46)	
Avg. Rank	12.11	10.93	3.62	3.86	3.95	2.93	3.09	**2.78**
±	4.12	5.78	2.80	3.10	3.13	2.62	3.05	**2.63**

Table 2: Comparison of OS-PGSM to different SoA for 102 time series. The rank column presents the average rank and its standard deviation across different time series. A rank of 1 means the model was the best performing on all time series.

why the variants using Euclidean distance have worse performance. In addition, by setting $z = 1$, higher number of windows of size n_ω are created and as a result, a higher number of RoCs are computed (See Section 3.3). This contributes to creating richer information about RoCs of different candidates, compared to setting $z = n_\omega$ in OS-PGSM-Int and OS-PGSM-Int-Euc. Finally, OS-PGSM-St is even better then OS-PGSM-Per, which shows that unnecessary updates are not always beneficial. Opposingly, OS-PGSM which relies on informed adaption of the RoCs using concept drift detection is better than OS-PGSM-Per and OS-PGSM-St. This can be explained by the fact that the update of the RoCs is only beneficial for datasets where concept drifts can be detected and more probably taking into account these new appearing concepts is helpful for the selection of models since a knowledge of which models are more adequate to handle these patterns if they ever reoccur again is gained and the old sets of RoCs are enriched. Figure 1 show an illustrative example of the RoCs of C_7 before and after drift detection. New patterns are added as new RoCs to the old RoCs of of C_7 . This answers research questions **Q2-Q3**.

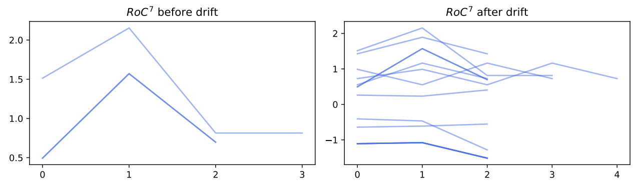

Fig. 1: RoCs for the candidate model C_7 before and after drift detection.

In the next experiment, we compare the runtime of OS-PGSM and its variants against the most competitive SoA method, ADE-Single, in Table 3. The reported runtime for OS-PGSM and OS-PGSM-Per takes into account the computation of the new RoCs. ADE-Single [9] relies on periodic update of the meta-learning

Method	ADE-Single	OS-PGSM	OS-PGSM St	OS-PGSM Int	OS-PGSM Per
Avg. Runtime	167.87	8.42	2.33	0.90	154.11
±	56.40	18.30	4.80	1.65	204.22

Table 3: Empirical runtime comparison between different methods in Seconds.

strategy behind the selection (same periodicity as OS-PGSM-Per). All the reported runtimes concern only the online predictions and any operation computed offline is not taken into account. The results demonstrate that OS-PGSM and its variants have lower average runtime than ADE-Single. OS-PGSM-Int is faster than OS-PGSM-St since fewer evaluation windows of size n_ω are created and as a result, a lower number of RoCs is generated for distance comparisons. OS-PGSM has lower runtime than OS-PGSM-Per. This is due to using drift detection to update the RoCs only when necessary. This results in faster predictions and less computational requirements. The high deviation of the runtime of OS-PGSM is due to the different numbers of drifts per time series. This answers question **Q4**.

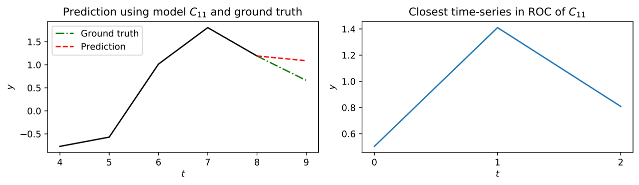

Fig. 2: Comparison of the current input pattern to the closest RoC (C_{11}).

Last but not least, we provide some insights how OS-PGSM can be used to provide suitable explanations for the reason behind model selection. Figure 2 shows a comparison between the current input time series pattern p_t^k (left part in black) with the RoC of the selected model to perform the forecast. A clear similarity between both patterns can be observed which justifies the choice of this model since it has been proven to show some degree of competence in forecasting using similar patterns as input. This is further validated when also comparing between the true time series value (ground truth, green) and the predicted value (red). while these two values differ slightly, an evaluation of all the candidates in this point showed that our selected model C_{11} has the smallest error. A more general overview over the RoCs for AbnormalHeartbeat dataset is shown in Figure 3. Some models have quite similar RoCs (patterns). For example, C_0 appears to be expert in increasing linear trend patterns, or in peaks followed by a slight plateau, while C_7 is the best in dealing with sharp peaks. As it can be seen with the varied amount of transparency of lines of the RoCs, many identical RoCs are collected for each models, confirming thus the assumption that certain models are experts on specific input regions of the time series. Some

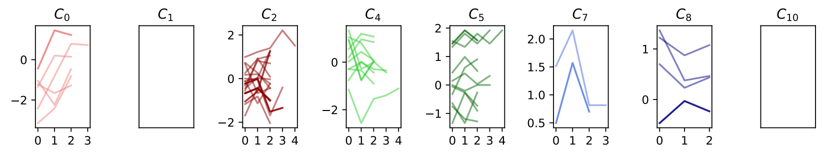

Fig. 3: Visualization of RoCs for AbnormalHeartbeat data using OS-PGSM.

models do not have any RoC. This can be explained by the fact that they never get selected in the validation process or their PGSMs are too small to form a pattern that it is why they get filtered out in the smoothing procedure. This also leads to better explainability in the sense of sparseness, since not every model is forced to contribute to the forecasting. Hence, models that are poorly designed or not well-trained, get ignored during the selection. Practitioners of our method can then use this insight to focus their attention on improving certain, poorly performing models, or remove the unused models entirely to save runtime. Another visualization aspect of the forecasting is shown in Figure 4. We focus

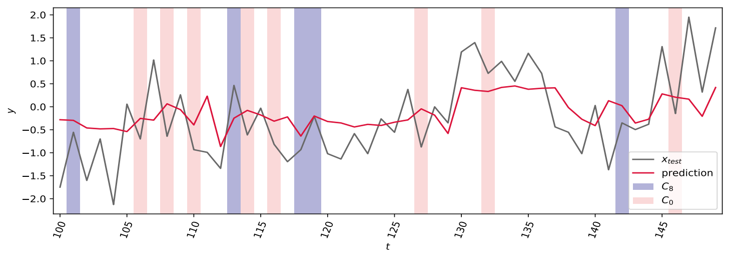

Fig. 4: A visualization of model selection one the AbnormalHeartbeat dataset.

for visualization clarity on the two models (C_8 and C_0). We highlight regions where they are selected by OS-PGSM. Notice that preceding every decision where C_0 is chosen, the time series exhibits a peak, which corresponds to the model's RoCs in Figure 3. The same conclusion can be drawn for C_8, which is picked after valley-shaped parts. While the two models are not picked for all peaks and valleys, our method clearly aligns certain time series regions with specific models. All the shown aspects address clearly research question **Q5**.

4.4 Discussion and future work

The empirical results indicate that OS-PGSM has performance advantages compared to popular forecasting methods and the most recent SoA approaches for online forecasting models selection. We show that our method, using PGSMs for adaptively computing RoCs of different CNN-based forecasters, is able to gain excellent and reliable empirical performance in our setting. The informed update of the RoCs following concept drift detection makes our method in addition to better predictive performance, computationally cheaper than the most competitive SoA, namely ADE-single. OS-PGSM can also successfully be used for providing useful explanations behind model selection which can be used to optimize further our framework. As future work, we plan to investigate the impact of varying some parameters in our setting, more specifically n_ω and k. More candidate models can also be considered. The selection can be made in favour of top-M best performing models, so that an adaptive dynamic ensemble can be created. In addition, the possible resulting big number of RoCs can be optimized further using a clustering inside each RoC^j for each model, and only clusters representatives are considered for distance computation to select the best model.

5 Concluding Remarks

This paper introduces OS-PGSM: a novel, practically useful online CNN-based models selection using saliency maps framework for time series forecasting. OS-PGSM uses gradient-based saliency maps to derive Region of Competences RoCs of a set of candidate models. These RoCs are updated in an informed-manner using concept drift detection in the time series. An exhaustive empirical evaluation, including 102 real-world datasets and multiple comparison algorithms showed the advantages of OS-PGSM in terms of performance and scalability.

References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity checks for saliency maps. arXiv preprint arXiv:1810.03292 (2018)
2. Argiento, R., Guglielmi, A., Pievatolo, A.: Bayesian density estimation and model selection using nonparametric hierarchical mixtures. Computational Statistics & Data Analysis **54**(4), 816–832 (2010)
3. Assaf, R., Schumann, A.: Explainable deep neural networks for multivariate time series predictions. In: IJCAI. pp. 6488–6490 (2019)
4. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD workshop. vol. 10, pp. 359–370 (1994)
5. Binkowski, M., Marti, G., Donnat, P.: Autoregressive convolutional neural networks for asynchronous time series. In: International Conference on Machine Learning. pp. 580–589. PMLR (2018)
6. Birgé, L., Massart, P.: Gaussian model selection. Journal of the European Mathematical Society **3**(3), 203–268 (2001)
7. Borovykh, A., Bohte, S., Oosterlee, C.W.: Conditional time series forecasting with convolutional neural networks. arXiv preprint arXiv:1703.04691 (2017)
8. Cerqueira, V., Torgo, L., Pinto, F., Soares, C.: Arbitrated ensemble for time series forecasting. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 478–494. Springer (2017)
9. Cerqueira, V., Torgo, L., Pinto, F., Soares, C.: Arbitrage of forecasting experts. Machine Learning **108**(6), 913–944 (2019)

10. Demertzis, K., Iliadis, L., Anezakis, V.D.: A deep spiking machine-hearing system for the case of invasive fish species. In: INISTA. pp. 23–28 (2017)
11. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery **33**(4), 917–963 (2019)
12. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM computing surveys (CSUR) **46**(4), 1–37 (2014)
13. Gamboa, J.C.B.: Deep learning for time-series analysis. arXiv preprint arXiv:1701.01887 (2017)
14. Gers, F.A., Eck, D., Schmidhuber, J.: Applying lstm to time series predictable through time-window approaches. In: Neural Nets, pp. 193–200. Springer (2002)
15. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
16. Hoeffding, W.: Probability inequalities for sums of bounded random variables. In: The Collected Works of Wassily Hoeffding, pp. 409–426. Springer (1994)
17. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural computation **3**(1), 79–87 (1991)
18. Jain, G., Mallick, B.: A study of time series models arima and ets. Available at SSRN 2898968 (2017)
19. Kim, T.Y., Cho, S.B.: Predicting residential energy consumption using cnn-lstm neural networks. Energy **182**, 72–81 (2019)
20. Livieris, I.E., Pintelas, E., Pintelas, P.: A cnn–lstm model for gold price time-series forecasting. Neural computing and applications **32**(23), 17351–17360 (2020)
21. Mittelman, R.: Time-series modeling with undecimated fully convolutional neural networks. arXiv preprint arXiv:1508.00317 (2015)
22. Priebe, F.: Dynamic model selection for automated machine learning in time series (2019)
23. Rivals, I., Personnaz, L.: On cross validation for model selection. Neural computation **11**(4), 863–870 (1999)
24. Romeu, P., Zamora-Martínez, F., Botella-Rocamora, P., Pardo, J.: Time-series forecasting of indoor temperature using pre-trained deep neural networks. In: International conference on artificial neural networks. pp. 451–458. Springer (2013)
25. Saadallah, A., Moreira-Matias, L., Sousa, R., Khiari, J., Jenelius, E., Gama, J.: Bright—drift-aware demand predictions for taxi networks. IEEE Transactions on Knowledge and Data Engineering **32**(2), 234–245 (2020)
26. Saadallah, A., Priebe, F., Morik, K.: A drift-based dynamic ensemble members selection using clustering for time series forecasting (2019)
27. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: ICCV. pp. 618–626 (2017)
28. Taieb, S.B., Bontempi, G., Atiya, A.F., Sorjamaa, A.: 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 (2012)
29. Utgoff, P.E., Stracuzzi, D.J.: Many-layered learning. Neural computation **14**(10), 2497–2529 (2002)
30. Wolpert, D.H.: Stacked generalization. Neural networks **5**(2), 241–259 (1992)
31. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
32. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: CVPR. pp. 2921–2929 (2016)