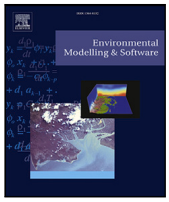
Environmental Modelling and Software 170 (2023) 105851

Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: www.elsevier.com/locate/envsoft



Effective air pollution prediction by combining time series decomposition with stacking and bagging ensembles of evolving spiking neural networks

Piotr S. Maciąg [a,*], Robert Bembenik [a], Aleksandra Piekarzewicz [a], Javier Del Ser [b,c], Jesus L. Lobo [b], Nikola K. Kasabov [d,e,f]

[a] Warsaw University of Technology, Institute of Computer Science, Nowowiejska 15/19, 00-665, Warsaw, Poland
[b] TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain
[c] University of the Basque Country (UPV/EHU), 48013 Bilbao, Bizkaia, Spain
[d] Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland University of Technology, AUT Tower, Level 7, 2-14 Wakefield Street, Auckland 1010, New Zealand
[e] University of Ulster, United Kingdom
[f] IICT Bulgarian Academy of Sciences, Bulgaria

ARTICLE INFO

Keywords:
Air pollution prediction
Evolving spiking neural networks
Bagging ensembles
Stacking ensembles
CEEMDAM
Time series decomposition

ABSTRACT

In this article, we introduce a new approach to air pollution prediction using the CEEMDAN time series decomposition method combined with the two-layered ensemble of predictors created based on the stacking and bagging techniques. The proposed ensemble approach is outperforming other selected state-of-the-art models when the bagging ensemble consisting of evolving Spiking Neural Networks (eSNNs) is used in the second layer of the stacking ensemble. In our experiments, we used the PM_{10} air pollution and weather dataset for Warsaw. As the results of the experiments show, the proposed ensemble can achieve the following error and agreement values over the tested dataset: error RMSE 6.91, MAE 5.14 and MAPE 21%; agreement IA 0.94. In addition, this article provides the computational and space complexity analysis of eSNNs predictors and offers a new encoding method for spiking neural networks that can be effectively applied for values of skewed distributions.

1. Introduction

Prediction of air pollution values is an important research task intensively studied in the research literature. Long-term exposition to excessive air pollution levels can lead to severe medical conditions, such as cardiovascular diseases, stroke or asthma (Badyda and Grellier, 2014). According to the European Environment Agency estimations, excessive air pollution levels contribute to around 50 000 premature deaths in Poland each year (EEA, 2020). Brunekreef and Holgate (2002) comprehensively reviewed scientific literature providing strong evidence for the adverse effects of air pollution on human health and well-being.

Among the most hazardous air pollutants (Brunekreef and Holgate, 2002) enumerates PM_{10} and $PM_{2.5}$ - fine particles with diameters <10 μm and <2.5 μm, respectively. Both PM_{10} and $PM_{2.5}$ pollutants tend to accumulate in the human respiratory system and can contribute to developing different types of respiratory diseases. The other commonly monitored and hazardous pollutants are ozone pollution (O_3) or nitrogen oxides (NO_x). High levels of O_3 and NO_x pollution contribute to the irritation and emergencies of the human respiratory system (Zhang et al., 2019). In a meta-review based on 2091 publications of hazardous effects of ozone pollution from the years 1988–2013, Bell et al. (2014) concluded that the damaging effects of short-term ozone pollution exposure are most dangerous for the older populations, significantly rising the mortality rate when compared to the younger populations. Thus, it is of particular importance to develop effective methods of air pollution prediction in order to minimize its harmful impact on human health and life.

1.1. Key scientific issues in air pollution prediction

Despite continuous development of new and more advanced methods dedicated to air pollution prediction, there are still key scientific problems that remain. One can enumerate several of them:

- Lack of appropriate sensors. Often times a large number of observations of air pollution values in the existing datasets is missing

* Corresponding author.
E-mail addresses: piotr.maciag@pw.edu.pl (P.S. Maciąg), robert.bembenik@pw.edu.pl (R. Bembenik), aleksandra.piekarzewicz.stud@pw.edu.pl (A. Piekarzewicz), javier.delser@tecnalia.com (J. Del Ser), jesus.lopez@tecnalia.com (J.L. Lobo), nkasabov@aut.ac.nz (N.K. Kasabov).

https://doi.org/10.1016/j.envsoft.2023.105851
Received 11 April 2023; Received in revised form 9 October 2023; Accepted 12 October 2023
Available online 16 October 2023

due to the sensors' failures or even lack of appropriate sensors in specific locations. In addition, sensors in even the most developed air pollution monitoring networks (such as the DEFRA DEFRA, 2018 network) usually provide data for only limited, most common types of pollutants, such as $PM_{2.5}$ or O_3 pollution. Data for some other pollutant types, such as e.g. sulfur dioxide (SO_2), are generally much less available.

- Effective integration of data from multiple sources. The datasets applied for the training of an effective air pollution predictor often need to integrate data from multiple sources, such as air pollutant values measured by various sensors and meteorological data (e.g. registered precipitation or wind parameters). Moreover, often sensors measuring air pollution values and weather parameters are placed in slightly different locations. Thus, it becomes the question of how to accurately interpolate such attributes in a specific location for which historical measurements are not available.
- Collecting a wide spectrum of data. In order to improve effectiveness of air pollution, it becomes increasingly important to collect datasets consisting not only of time series attributes (such as observed air pollution or precipitation values), but also of other data e.g. satellite images representing cloud coverage in a region of pollution prognosis (Mao et al., 2017). Such diverse and multimodal data can be difficult to obtain or can be available only in limited locations.
- Effective training of a prediction model given limited amount of data. Often only limited sources can provide data (such as only a few air pollution sensors). If that is the case we cannot improve effectiveness of air pollution prediction simply by training model using more data. Thus, we must then search for a model that provides the best prediction results given the limited amount of data that we were able to acquire.

1.2. Main limitations of existing approaches and possible solutions

Several recent studies focused on analyzing key limitations of the existing approaches to air pollution prediction. Masood and Ahmad (2021) enumerates the following key limitations of the most popular prediction models, such as Support Vector Machine (SVM), Artificial Neural Networks (ANNs) and Deep Neural Networks (DNNs):

- Sensitivity to the selection of input parameters and their impact on the effectiveness of prediction quality.
- Significant computational and memory complexity of some type of prediction models, especially DNNs.
- Susceptibility to underperform in prediction of air pollution due to suboptimal selection of learning parameters affecting certain types of predictors (e.g. ANNs).

One of the possible solutions to the above-mentioned limitations are machine learning methods that combine model's training with algorithms (often inspired by nature) that aim to optimize the prediction quality of a taught model. Examples of such methods proposed in the literature are:

- Adnan et al. (2023a) presented prediction models that combine Convolutional Neural Networks (CNNs) and Long-Short Term Networks (LSTM) with two optimization methods called Reptile Search Algorithm (RSA) and weighted mean of vectors (INFO). The models of Adnan et al. (2023a) were successfully applied for prediction of water temperature. As it was shown there, the LSTM-INFO model was highly effective in making predictions, outperforming other combinations of prediction and optimization models tested there.
- In Adnan et al. (2022a), a heuristic-based Chaotic Red Fox Optimization Algorithm (CRFOA) was introduced. The developed algorithm was applied to search for the optimal parameters of the activation functions and neuronal weights of the Extreme Learning Machine network. The authors of Adnan et al. (2022a) named the model ELM-CRFOA and presented that it is highly effective in performing prediction tasks such as the estimation of the shear strength of concrete beams.
- Adnan et al. (2023b) offered a new method that combines optimization algorithm called Improved Manta-Ray Foraging Optimization (IMRFO) with Relevance Vector Machine (RVM) model in order to predict monthly pan evaporation. Similarly to the results presented in Adnan et al. (2022a, 2023a), the offered RVM-IMRFO was proven to be highly effective in making predictions, especially when compared with the models that did not apply any optimization algorithm (such as the standard Multilayer Perceptron MLP network). Other innovative approaches combining heuristic methods with time series single predictors are: ANFIS-WCAMFO presented in Adnan et al. (2021) or SVM-FFAPSO (Adnan et al., 2022b).

The other effective prediction approach (adopted by us in this study) is to apply ensemble models and time series decomposition. For example, Yang et al. (2023) proposed approach in which the two data decomposition models (i.e. Variational Mode Decomposition, VMD, and Complete Ensemble Empirical Mode Decomposition with Adaptive Noise, CEEMDAN) were linked with two predictors: Deep Belief Network (DBN) and Gradient Boosted Regression Tree (GBRT) in order to forecast multi-time-step water streamflow. Moreover, the tested prediction models were combined into an ensemble using Bayesian Model Averaging (BMA) technique. The experiments presented in Yang et al. (2023) showed that the offered ensemble model gives better prediction quality than singleton models. A similar BMA ensemble model was applied to $PM_{2.5}$ pollution prediction in Zhou et al. (2020).

1.3. Our approach

In order to effectively predict air pollution, in this work we propose an advanced stacking ensemble model that is characterized as follows:

1. First, decomposition of air pollution time series data using the CEEMDAN method (Torres et al., 2011) is used. The decomposed modes of time series are combined with original air pollution and weather time series in order to obtain training and testing datasets.
2. Subsequently, the construction of a stacking ensemble that in the first layer consists of a set of predictors, each of which is taught using the same training dataset and makes its own air pollution prediction.
3. The second layer of the stacking ensemble is composed of either a bagging ensemble of predictors or a single predictor. Such predictors are taught solely using prediction values obtained from the first stacking layer. As we presented in the experiments, the prediction quality tends to outperform other models when the bagging ensemble of **evolving Spiking Neural Networks (eSNNs)** is used.

Previous research showed that **eSNN networks** could be effectively used for prediction in various tasks and problems. Examples of the research problems that have already successfully adapted eSNNs include classification in streaming data (Lobo et al., 2018), air pollution prediction (Maciąg et al., 2019; Maciąg et al., 2020), traffic estimation (Laña et al., 2019, 2018) and anomaly detection (Maciąg et al., 2021; Demertzis et al., 2019). To the distinctive characteristics of eSNNs belong an evolving repository of output neurons that in the network's training process is updated with a new candidate output neuron created for each new training instance of data. Another distinctive eSNNs feature is the

method for calculating synapses' weights between output neurons and neurons in either internal or input layer using spikes' ranks obtained for the input values encoding (Kasabov, 2014). Thus, eSNNs are suitable for incremental, theoretically, life-long learning.

eSNNs not only adopt spiking neuronal models, synapses' learning rules and encoding methods, but also provide mechanisms (such as an evolving repository of output neurons) that are particularly useful in various classification and prediction engineering tasks. Such a design of eSNNs is inspired by the *Evolving Connectionist Systems* design paradigm (Kasabov, 1998). The connection weights of each output neuron that represent a prototype area of the input data, can be presented as a fuzzy rule, providing explainability, thus departing from the black box neural network systems (Maciąg et al., 2019).

1.4. Contributions of the article

In this paper, we provide the following primary contributions:

- We propose an advanced air pollution prediction method that consists of air pollution time series decomposition, stacking and bagging ensembles of predictors. As we showed, the proposed method is especially effective when the bagging eSNNs are applied in the second layer of the stacking ensemble.
- We offer learning and prediction algorithms with the bagging ensembles of eSNNs. The results of experiments suggest that bagging eSNNs in some cases perform substantially better than a single eSNN. In this work, the proposed algorithms are adapted for air pollution prediction, but they can also be adapted to other prediction problems.
- To the best of our knowledge, for the first time in the literature we provide the computational and space complexity analysis of the learning and prediction algorithms of eSNN networks. We extend this analysis also to the proposed bagging ensemble model.
- We experimentally compare the results obtained using our proposed bagging ensembles of eSNNs with other seven state-of-the-art predictors. We use the Warsaw air pollution and weather datasets for this purpose. The experimental results show that the proposed approach gives better prediction quality than other broadly used predictors.

In addition to the above enumerated primary contributions, the article provides also the following secondary contributions:

- We introduce a new formula for calculating an eSNN's prediction value based on weighting output values of output neurons in this eSNN.
- We develop a categorical data encoding method for an eSNN and provide a formula that allows us to calculate the tight upper bound between a vector of synapses' weights of a candidate output neuron and vectors of synapses' weights of output neurons present in the network's repository.
- In Appendix, we provide a new method for allocating Gaussian Receptive Fields (GRFs) values based on a histogram of values of a given attribute. This method can be particularly useful in a case of highly skewed values distributions of dataset's attributes/features. The GRFs allocation method is based on the encoding method proposed in Maciąg et al. (2022).

This paper is structured as follows. Section 2 summes up the related work. Section 3 gives a description of the introduced stacking ensemble model. Section 4 presents learning and prediction principles of eSNNs. In Section 5, we present the designed bagging eSNNs algorithms. In Section 6, we provide the computational and space complexity analysis of the eSNNs learning and prediction algorithms as well as its bagging ensembles. Section 7, first describes the selected datasets and then presents the results of the experiments. Section 8 concludes and discusses the work. Finally, in Appendix we provide our method for allocating GRFs based on a histogram of values of an attribute.

2. Related work

Many types of air pollution models have been recently introduced in the literature. Sometimes they are dedicated to the prediction of a specific air pollutant type, such as ozone (O_3), particulate matter PM_{10} and $PM_{2.5}$ or carbon dioxide (CO_2). Neural networks are among the most widely used types of predictors. The initial attempts to predict air pollution applied the MLP neural networks. For example, Gardner and Dorling (1999) and Kukkonen et al. (2003) used this type of neural network for the prediction of NO_2 air pollutants in London and Helsinki, respectively. A comparison of the prediction effectiveness of an MLP network with the regression trees and linear regression in predicting air quality in Christchurch, New Zealand, presented in Gardner (1999) suggests that MLP predictors perform better than the other two selected methods. In Maciąg et al. (2020), a prediction model of an online evolving spiking neural network was adapted to air pollution forecasting in streaming data. The model offered in Maciąg et al. (2020) was superior to several other types of predictors, such as the RBF neural networks and the Elman networks.

Recently, some works studied the effectiveness of using ensembles of air pollution predictors. In Siwek et al. (2009), an ensemble of various types of predictors (such as the RBF neural network, support vector regression, Elman networks, MLP neural network or Auto-Regressive with eXogenous input predictor) combined with the wavelet decomposition of input pollution and weather time series data was applied to forecast PM_{10} pollutant. The ensemble architecture presented in Siwek et al. (2009) consisted of the above-mentioned types of predictors, each providing its own forecasting result. The final ensemble forecast was calculated as an average of the results each predictor returned. Maciąg et al. (2019) proposed a clustering-based ensemble model consisting of eSNNs networks, each of which is taught using a set of air pollution time series obtained from the clustering of an input training dataset. The examples of recent applications of ensembles to air pollution prediction are presented in Van Roode et al. (2019), Zhou et al. (2020) and Guo et al. (2020).

Other studies investigated the relationship between different types of pollutants and meteorological factors in the effectiveness of air pollution prediction. For example, Shaharuddin et al. (2008) and Zaharim et al. (2009) showed that there is a significant correlation between values of PM_{10} pollution and low-frequency components of weather attributes (such as temperature, rainfall or wind speed).

In this work, we experimentally compare the proposed bagging ensemble of eSNNs with a number of the other state-of-the-art predictors offered in the literature, namely: SVM (Suthaharan, 2016), MLP neural network (Gardner and Dorling, 1998), Elastic net, linear regression and LSTM networks. We also provide the prediction results for the SNN implementation available in the SNNTorch package (Eshraghian et al., 2021). A distinctive feature of the SNNTorch implementation is its ability to combine the SNNs neuronal models (such as the Leaky-Integrate-and-Fire model) with the learning algorithms of MLP networks (such as the backpropagation algorithm). The enumerated predictors were combined with the CEEMDAM decomposition method and the proposed stacking ensemble.

The review of the current methods of air pollution prediction can be found in Siwek and Osowski (2016), Kowalski and Warchałowski (2018), Mao et al. (2017), Zaharim et al. (2009), Feng et al. (2015) and Bozdağ et al. (2020), while Cabaneros et al. (2019) provides a review of methods that specifically use neural networks to predict air pollution.

3. The proposed prediction method

In this section, we offer our proposed air pollution prediction method that combines data decomposition, stacking and bagging ensembles. In Fig. 1, we present the architecture of the proposed method.

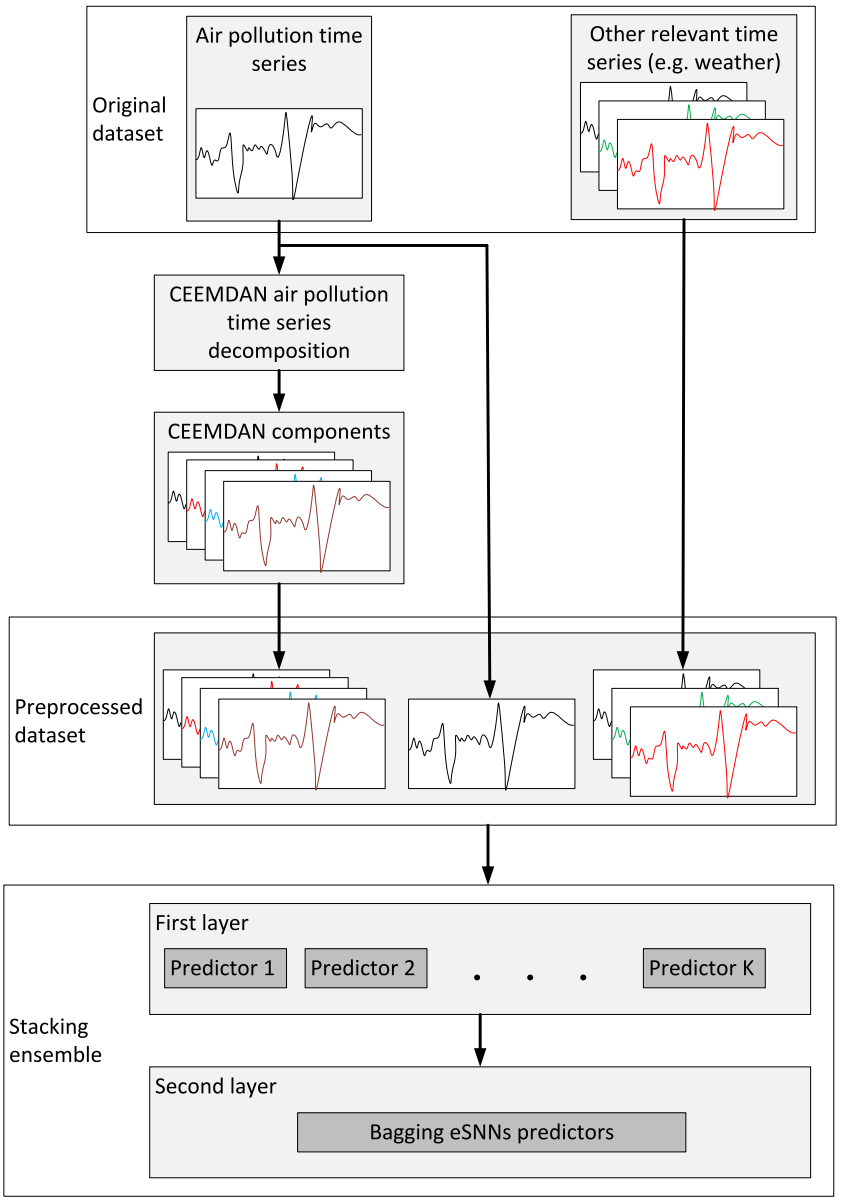


Fig. 1. The architecture of the proposed ensemble model.

3.1. Air pollution time series decomposition

Previous research (see, for example, Siwek et al. (2009), Siwek and Osowski (2012)) showed that decomposing air pollution time series into several modes and including these modes in a model's training data can significantly improve air pollution prediction quality. Such decomposition is particularly useful for air pollution prediction problem since training datasets are often composed of only a limited number of attributes (such as weather attributes only weakly correlated with air pollution). Thus, by including attributes highly correlated with air pollution we can improve prediction quality.

In our approach, for the air pollution time series decomposition we applied the CEEMDAN method (Torres et al., 2011). CEEMDAN is a method that is based on another time series decomposition method called Ensemble Empirical Mode Decomposition (EEMD) (Wu and Huang, 2009). Unlike EEMD, which adds white noise to the decomposed time series before the decomposition is conducted, CEEMDAN adds white noise to each subsequent component obtained as the result of decomposition. As the Wu and Huang (2009) showed, CEEMDAN requires less decomposition steps than EEMD and is significantly more computationally efficient. For the preprocessing of the Warsaw air pollution time series, we applied the CEEMDAN implementation available in the PyEMD Python package (Laszuk, 2023). We presented the details of the applied CEEMDAN parameters in Section 7.

Let us denote the original, acquired dataset containing air pollution and weather attributes as $\mathbf{D}_o$. After air pollution time series of $\mathbf{D}_o$ is decomposed by CEEMDAN, the obtained modes are inserted into $\mathbf{D}_o$ in order to obtain input dataset $\mathbf{D}$ ($\mathbf{D}$ is split into training $\mathbf{D}_{tr}$ and testing $\mathbf{D}_{ts}$ parts).[1] Dataset $\mathbf{D}$ consists of dataset instances D, each of which contains a vector of values of attributes $A_1, \ldots, A_m \in \mathbf{A}$ and a target value.

[1] Please note that in our approach we decompose air pollution time series of $\mathbf{D}_o$ using CEEMDAN before splitting it into training and testing parts. In a real deployment scenario, one would usually expect to separately decompose air pollution of only training dataset $\mathbf{D}_{tr}$, and subsequently, decompose also the dataset used for real-time predictions.

3.2. Stacking ensemble

After air pollution time series is decomposed by the CEEMDAN method, the stacking ensemble is constructed and used to make air pollution predictions. In the proposed approach, the stacking ensemble consists of two layers:

1. The first layer contains a set of different types of predictors, each of which being taught using the training dataset $\mathbf{D}_{tr}$ consisting of historical air pollution, weather and decomposed air pollution time series as well as target air pollution values. As the result, the first layer generates a dataset containing air pollution predictions from each predictor. In the preliminary experiments, we tested different types of predictors for the first layer and selected the best results they provide.
2. The second layer of the proposed ensemble contains the set of bagging eSNNs predictors.[2] Such nested bagging ensemble is taught solely using data obtained from the first layer.

4. Evolving spiking neural networks

As we will present in Section 7, bagging eSNNs are highly effective predictors. Thus, in this section, we first review the most important differences between eSNNs and a typical SNNs architecture. Subsequently, we provide a description of the encoding, training and prediction principles of an eSNN. The description provided in this section is essential to define the bagging eSNNs learning and prediction algorithms given in the next section.

4.1. Key differences between eSNNs and SNNs

SNNs are a type of neural networks whose learning and classification rules are highly inspired by the learning and working principles of biological neural networks (Ponulak and Kasinski, 2011; Gerstner et al., 2014; Kugele et al., 2020). In particular, the developed neuronal models of SNNs (such as the Integrated-and-Fire or Leaky-Integrated-and-Fire models Gerstner and Kistler, 2002) as well as the learning rules of SNNs, such as the Spike-Time Dependent Plasticity, are supposed to closely mimic the behavior of biological neurons. Unfortunately, this often leads to situations in which for some machine learning tasks, such as classification or prediction, SNNs can be outperformed by other types of models whose learning mechanisms are aimed to maximize the model's accuracy (Kugele et al., 2020). On the other hand, as presented by Kugele et al. (2020), SNNs use an event-driven, asynchronous model of spike emissions, and thus they can be more computationally and energy efficient than synchronous ANNs.

eSNNs adapt and modify the usually complex architecture of SNNs in order to make it more effective in performing machine learning tasks, such as classification, prediction or anomaly detection. While SNNs were designed to simulate various processes of biological neural networks, eSNNs were proven to be suitable for solving engineering tasks that require minimization of an error calculated as the difference between the results returned by the taught model and the ground-truth labels of the input dataset.

In Table 1, we provide an overview of key differences between eSNNs and SNNs.

4.2. Architecture of eSNNs

Herein, we present the adapted architecture of an eSNN network as well as its learning and prediction principles. In this work, we use an eSNN network consisting of two layers (input and output). The architecture of a singleton eSNN is presented in Fig. 2.

[2] For comparison purposes we also tested other predictors in this layer.

The aim of the input layer of an eSNN network is to encode values of attributes $\mathbf{A}$ of each dataset instance D into firing order of input neurons $\mathbf{NI}$. The firing order of an input neuron $n_j \in \mathbf{NI}$ is denoted as $order_{n_j}$. The encoding method that we apply is based on the *rank-order* encoding frequently used in spiking neural networks (Ponulak and Kasinski, 2011). According to rank-order encoding, each input value (such as an input pollution value) is transformed to a firing order of a set of input neurons.

The output layer of the network consists of a repository of neurons $\mathbf{NO}$, each of which is linked to every input neuron in $\mathbf{NI}$. Each output neuron of $\mathbf{NO}$ is assigned an output value that can be returned by the network in the prediction phase.

4.3. Input data encoding

In eSNN, a dedicated group of input neurons (denoted here as $\mathbf{NI}^{(A)}$) is used to encode input values of each attribute $A \in \mathbf{A}$ of a dataset $\mathbf{D}$. We assume that each attribute $A \in \mathbf{A}$ contains either continuous (real) values, ordinal (ordered discrete) values or nominal (unordered discrete) values. We refer to the sets of these types of attributes using the following notation: $\mathbf{A}^{(c)}$ - continuous attributes, $\mathbf{A}^{(o)}$ - ordinal attributes, $\mathbf{A}^{(n)}$ - nominal attributes. More importantly, we assume that the values of each ordinal and nominal attribute of the acquired dataset $\mathbf{D}$ are transformed in the preprocessing step into distinct and consecutive numbers starting with number 1. For example, weekdays are encoded using consecutive numbers between 1 and 7.

Previous research focused specifically on the encoding of values of continuous attributes $\mathbf{A}^{(c)}$ into the firing order of eSNN input neurons. For this purpose, one can consider adaptation of the GRFs encoding method (Lobo et al., 2020; Maciąg et al., 2021), which calculates firing times of input neurons, or the encoding technique offered in Maciąg et al. (2020), which directly calculates the firing order of input neurons without computing their firing times (the method of Maciąg et al. (2020) can also be applied to encode ordinal attributes $\mathbf{A}^{(o)}$ after their transformation into numbers denoting levels of values). Alternatively, one can consider applying the encoding method offered in Maciąg et al. (2022) that is particularly useful in the encoding of continuous values that have non-uniform distributions.

4.3.1. Encoding of continuous and ordinal values

In order to encode values of continuous and ordinal attributes one can select from different methods proposed in the literature. The examples of methods that can be effectively used with eSNN networks are presented in Fig. 3.

After the preliminary experiments and the dataset's analysis, for the encoding of values of continuous and ordinal attributes we applied the method of Maciąg et al. (2020). To the distinctive features of this method belong: (i) efficient encoding even for a large number of input neurons, (ii) lack of numerical errors when values are encoded (which are often present in the case of the other methods such as GRFs) and (iii) only one control parameter of the method: the number of input neurons encoding each continuous or ordinal attribute (we denoted this parameter as NI_{size}).

After the input neurons of attributes $\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}$ are initialized, they are used to calculate firing order for each dataset instance D in either training or testing datasets: $\mathbf{D}_{tr}$ or $\mathbf{D}_{ts}$. Let $D(A)$ denote the value of an attribute A for a dataset instance D. According to the method of Maciąg et al. (2020), for each attribute $A \in \mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}$, firing order function $order_{n_j}$ of the first firing input neuron in $n_j \in \mathbf{NI}^{(A)}$ is equal to 1 and the value of the $order$ function is incremented by 1 for each subsequent firing input neuron in $\mathbf{NI}^{(A)}$. We refer the reader to Maciąg et al. (2020) for the complete description of the encoding method employed by us to encode continuous and ordinal attributes.

In Appendix, we provide the extension of the other encoding method for the continuous and ordinal attributes proposed in Maciąg et al. (2022) that incorporates the GRFs encoding. The main difference

Table 1
Key differences between SNNs and eSNNs.

Network property	SNNs	eSNNs
Architecture of the network	The network consists of three layers: input, internal and output. Only internal layer fully mimics neuronal models of biological neurons and related learning rules (such as STDP). The aim of the input layer is to encode data, while the output layer contains neurons directly responsible for classification/prediction. Neurons in the internal layer can be organized into three dimensional structure mimicking the structure of human brain.	The architecture does not contain internal layer (only input and output layers are present). Neurons of the input layer are fully connected to the neurons of the output layer.
Data encoding methods	Uses various encoding methods depending on the type of input data (e.g. GRFs encoding or time-series encoding algorithms, such as TR, BSA or MV Tu et al. (2017))	Uses either the GRFs method or a method that directly calculates firing order of input neurons without calculation of their exact firing times.
Neuronal models	Internal layer adapts different types of neuronal models, such as IF or LIF. The behavior of input neurons is dependent on the used data encoding method. Output neurons can use simplified IF model and are assigned decision classes present in the input data.	The input layer consists of sets of input neurons, each set encoding values of a distinctive attribute of input data (the number of input neurons is a user-specified parameter NI_{size}). The output neurons can use e.g. IF neuronal model.
Applications	Medical data analysis, simulation of brain processes.	Engineering applications: classification, prediction, anomaly detection, air pollution prediction.

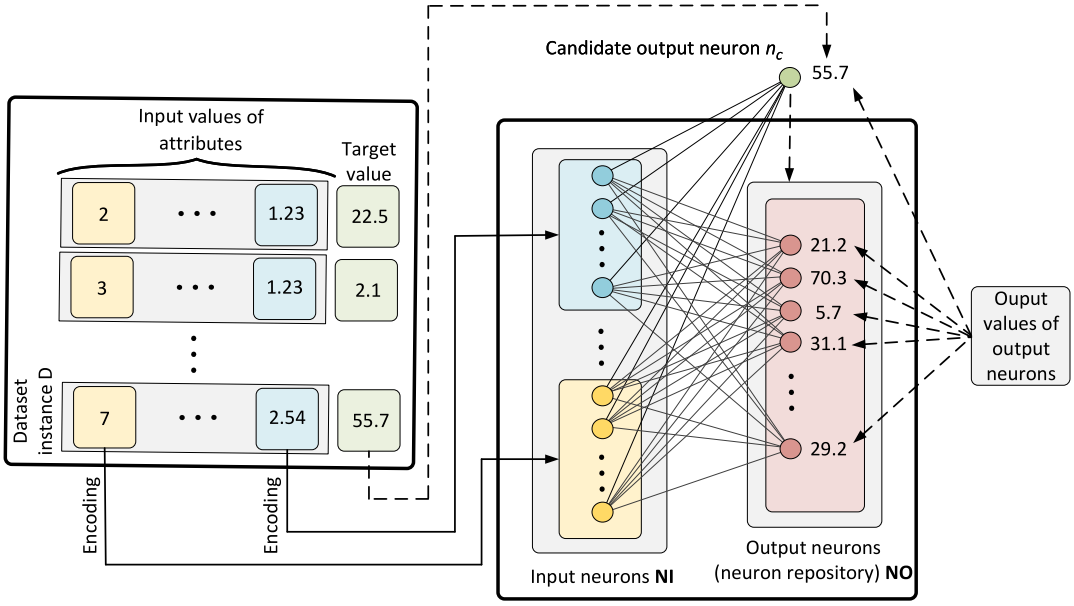


Fig. 2. Architecture of the singleton eSNN network adapted in this work.

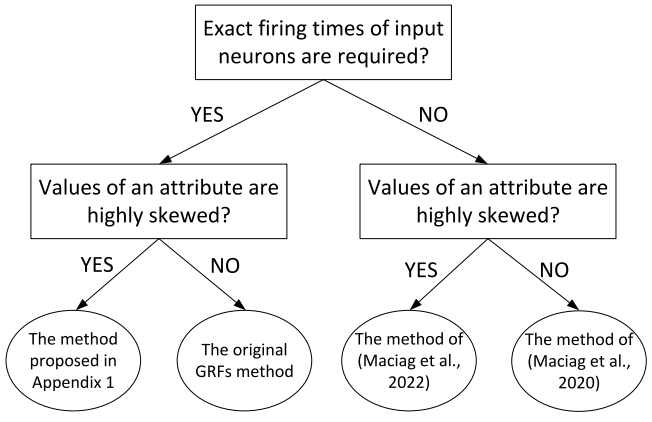


Fig. 3. Selection of encoding method for continuous and ordinal attributes considered and applied in the preliminary experiments of this study.

between the encoding method of Maciąg et al. (2022) and the extension provided in Appendix lies in the fact that the method of Maciąg et al. (2022) directly calculates the firing order of input neurons without the calculation of their exact firing times, while our extension offered in Appendix allows us to calculate such exact firing times based on the allocation of GRFs that uses a histogram of values of an input attribute.

The selection of the specific version of the encoding method (either the method of Maciąg et al. (2020) used in this work, the encoding method of Maciąg et al. (2022) or the method introduced in Appendix) should be based on the used SNN architecture. If an SNN network consists of a reservoir of input neurons connected using e.g. recurrent synapses whose weights are calculated according to the Spike-Time Dependent Plasticity (Hebb, 2005; Ponulak and Kasinski, 2011) rules, then the encoding method of Appendix will be more appropriate. Otherwise, if the SNN network consists of only two layers and it is enough to calculate only firing order of neurons rather than their exact firing times, the method of Maciąg et al. (2022) will be more efficient.

4.3.2. Encoding of nominal values

To encode values of nominal attributes $\mathbf{A}^{(n)}$ into the firing order of input neurons, we adapted the *one-hot* encoding method. The number of input neurons encoding each nominal attribute $A \in \mathbf{A}^{(n)}$ of a dataset

D is equal to the number of distinct values of that attribute in **D**. Specifically, assuming that a nominal attribute A consists of J distinct values in **D**, a set of input neurons $\mathbf{NI}^{(A)}$ will consist of J input neurons. Given dataset instance $D \in \mathbf{D}$ and a value $D[A]$ of a nominal attribute $A \in \mathbf{A}^{(n)}$ of D, only an input neuron $n_j \in \mathbf{NI}^{(A)}$ whose $j = D[A]$ will fire (that is, the value of $order_{n_j}$ will be set to 1, while the $order_{n_j}$ values for the remaining of neurons in $NI^{(A_n)}$ will be set to 0). For the sake of consistency, in this article we denote the number of input neurons encoding each attribute $A \in \mathbf{A}$ as $|\mathbf{NI}^{(A)}|$ (please note that for continuous and ordinal attributes $|\mathbf{NI}^{(A)}|$ is equal to NI_{size}, while for nominal attributes $|\mathbf{NI}^{(A)}|$ is equal to J).

Algorithm 1 presents the encoding procedure for the values of a dataset instance D.

Algorithm 1 InputLayerEncoding(D)

Input: D - a dataset instance to be encoded
1: **for each** $A \in \mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}$ **do** ▷ For each continuous or ordinal attribute
2: Calc. $order_{n_j}$ of all $n_j \in \mathbf{NI}^{(A)}$ according to the encoding method proposed in Maciąg et al. (2022)
3: **end for**
4: **for each** $A \in \mathbf{A}^{(n)}$ **do** ▷ For each nominal attribute
5: **for each** $n_j \in \mathbf{NI}^{(A)}$ **do**
6: **if** j ≠ D[A] **then**
7: $n_j \leftarrow$ does not fire
8: $order_{n_j} \leftarrow 0$
9: **else**
10: $n_j \leftarrow$ fires
11: $order_{n_j} \leftarrow 1$
12: **end if**
13: **end for**
14: **end for**

4.4. Network's training

The output layer of eSNN consists of a repository of output neurons **NO**, each of which is assigned a real value. In the network's learning process, for each dataset instance D in the training set of instances $\mathbf{D}_{tr}$, a new candidate output neuron n_c is created and used to update repository **NO**. The weights of synapses linking n_c to each input neuron $n_j \in \mathbf{NI}$ are stored in a vector $\mathbf{w}_{n_c} = [w^{(A_1)}_{n_1,n_c}, \ldots, w^{(A_1)}_{n_{|\mathbf{NI}^{(A_1)}|},n_c}, \ldots, w^{(A_m)}_{n_1,n_c}, \ldots, w^{(A_m)}_{n_{|\mathbf{NI}^{(A_m)}|},n_c}]$ and are initialized according to equation: $w^{(A)}_{n_j n_c} = mod^{order_{n_j}}$, where mod is a modulation factor whose value is specified by the user and should be in the range $(0, 1)$. If n_j does not fire, then the synapse's weight $w_{n_j n_c}$ is set to 0. Finally, n_c is assigned a target value v_{n_c} of the dataset instance D. Additionally, update counter M_{n_c} of each candidate output neuron n_c is equal to 1.

After the candidate neuron n_c is created and its synapses' weights are initialized, it is either added to the repository of output neurons **NO** or merged with one of the output neurons already existing in **NO**. Next, Euclidean distances $Dist_{n_c,n_i}$ between the vector of synapses' weights $\mathbf{w}_{n_c}$ and the vectors of synapses' weights $\mathbf{w}_{n_i}$ of each output neuron $n_i \in \mathbf{NO}$ are calculated. If there exists such an output neuron n_s for which $Dist_{n_c,n_s}$ is minimal and below the value $simTr \cdot \overline{Dist}$, then the vector $\mathbf{w}_{n_s}$ and counter M_{n_s} are updated according to Eq. (1) and n_c is discarded. Otherwise, n_c is simply inserted into **NO**. $simTr$ is a user-given similarity factor whose value is in the range [0, 1] and $\overline{Dist}$ is the tight upper bound on Euclidean distances between a vector of synapses weights of a candidate n_c and a vector of synapses weights of any output neuron in **NO** and is calculated according to our Proposition 1. Additionally, output value v_{n_s} is updated according to Eq. (1).

$$\mathbf{w}_{n_s} = \frac{\mathbf{w}_{n_s} \cdot M_{n_s} + \mathbf{w}_{n_c}}{M_{n_s} + 1}, v_{n_s} = \frac{v_{n_c} + M_{n_s} \cdot v_{n_s}}{M_{n_s} + 1}, M_{n_s} = M_{n_s} + 1. \quad (1)$$

Proposition 1. *The tight upper bound on the Euclidean distances between any possible candidate output neuron and any output neuron in* **NO** *is equal to:*

$$\overline{Dist} = \left[\sum_{A \in \mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}} \sum_{j=1}^{NI_{size}} \left(mod^{j} - mod^{NI_{size}-j-1} \right)^2 + 2 \cdot \sum_{A \in \mathbf{A}^{(n)}} mod^2 \right]^{\frac{1}{2}} \quad (2)$$

The internal state of each output neuron $n_i \in \mathbf{NO}$ is characterized by the value of a *Post-synaptic Potential*, PSP. The value of PSP_{n_i} of an output neuron n_i given current firing order of input neurons **NI** for the encoded dataset instance D is calculated according to Eq. (3).

$$PSP_{n_i} = \sum_{A \in \mathbf{A}} \sum_{n_j \in \mathbf{NI}^{(A)}} w^{(A)}_{n_j n_i} \cdot mod^{order_{n_j}}. \quad (3)$$

4.5. Prediction

Given a dataset instance D being subject to prediction and the encoding of attributes' values of D into firing order of input neurons $n_j \in \mathbf{NI}$, the first step of the prediction is the calculation of PSP values of all output neurons in **NO**. Some previous research (e.g. Lobo et al. (2018)) applied a prediction method according to which the dataset instance is assigned an output value of an output neuron in **NO** whose PSP value at first reaches a user-given PSP threshold. In order to increase prediction quality, Maciąg et al. (2020) proposed to first calculate PSP values of all output neurons and, subsequently, assign to the instance D a prediction value of an output neuron in **NO** whose PSP is maximal (if two or more neurons have the same maximal PSP values then dataset instance is assigned the average of output values of these neurons). However, as it was explained above, in the eSNN learning process, candidate output neurons are occasionally merged with output neurons in **NO** (the number of these merges for each output neurons $n_i \in \mathbf{NO}$ is equal to update counter M_{n_i}). To reflect these updates of output neurons, we propose to calculate a target value to be assigned to the dataset instance in the following way. First, a set of output neurons $\mathbf{NO}_{max}$ whose PSP values are maximal is obtained. Subsequently, the predicted value is calculated according to Eq. (4).

$$\text{Predicted-Value} = \frac{\sum_{n_i \in \mathbf{NO}_{max}} v_{n_i} \cdot M_{n_i}}{\sum_{n_i \in \mathbf{NO}_{max}} M_{n_i}}. \quad (4)$$

Contrary to, for example, Maciąg et al. (2020), Eq. (4) calculates target value based on weighted average of output values of output neurons in $\mathbf{NO}_{max}$ rather than based on arithmetic average.

5. eSNNs bagging model

The already developed eSNN learning and prediction methods tend to cause overfitting of the results and thus decrease the prediction quality when a predictor is deployed in a real-world application or a forecasting system. In Fig. 4, we illustrated such a situation for a singleton eSNN that was applied for a 1-day ahead prediction of air pollution in Warsaw using the training and testing datasets. A single eSNN tends to provide superior prediction results on a training dataset (for which the most predicted values are nearly equal to the observed values) but much more inferior on a testing dataset. This fact can be explained by a specific prediction method applied in eSNNs: each candidate output neuron (for each training instance a candidate output neuron is created) is assigned an output value (directly used for prediction) being the same as the target value of a corresponding training instance in $\mathbf{D}_{tr}$. Hence, the taught network can easily lose its ability to generalize prediction results when testing data is used.

Thus, in this article, we propose and experimentally verify learning and prediction algorithms of eSNN that apply the bootstrap aggregating (bagging) technique (Breiman, 1996; Polikar, 2012) to provide better prediction results. As we present in the experiments, the applied bagging eSNN ensembles provide substantially better prediction results than a singleton eSNN model.

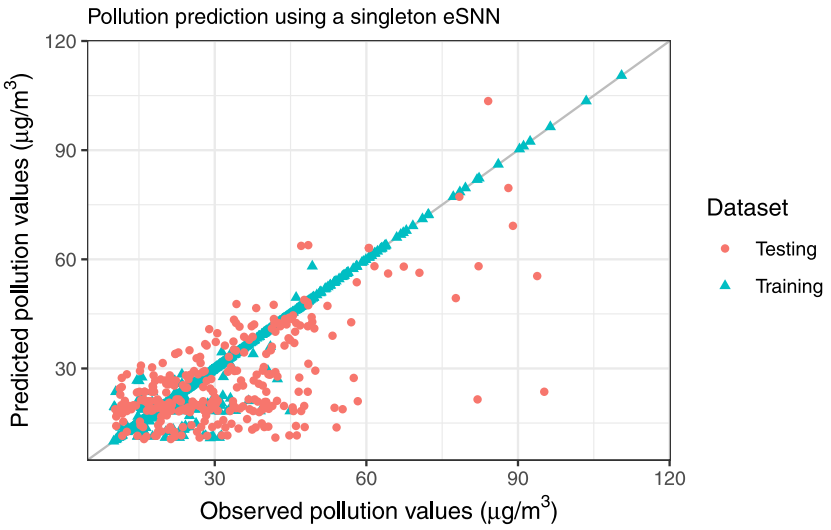


Fig. 4. An example of PM_{10} prediction results obtained using a singleton eSNN (blue triangles denote predictions for the training dataset and red dots denote predictions for the testing dataset) obtained for the Warsaw air pollution dataset.

5.1. Proposed bagging algorithm

In Algorithm 2, we present the proposed prediction procedure with the bagging ensemble of eSNN. The algorithm receives two datasets: training $\mathbf{D}_{tr}$ and testing $\mathbf{D}_{ts}$ (obtained from a random split of the original dataset $\mathbf{D}$) and the following input parameters: NI_{size} - the number of input neurons encoding continuous and ordinal attributes, mod - the modulation factor, $simTr$ - the fraction of the tight upper bound distance $\overline{Dist}$, N - the number of eSNNs in the ensemble. The algorithm consists of two phases: the first phase in which each eSNN in the ensemble is trained using a randomly generated sample of training dataset $\mathbf{D}_{tr}$ (we denote them as $\mathbf{D}_{tr}^{(i)}$, $i = 1, \ldots, N$), and the second phase in which each eSNN in the ensemble returns a prediction value for each instance of testing dataset $\mathbf{D}_{ts}$. In our implementation, the size of each training sample $\mathbf{D}_{tr}^{(i)}$ is equal to 75% of the number of dataset instances in $\mathbf{D}_{tr}$. The final prediction value assigned to an instance of $\mathbf{D}_{ts}$ is an average value obtained from all output values returned by each individual eSNN in the ensemble.

Algorithm 2 proceeds as follows. First, samples $\mathbf{D}_{tr}^{(i)}$, $i = 1, \ldots, N$ are generated (samples have the same size). Subsequently, ith eSNN in the ensemble is trained using a respective sample of instances $\mathbf{D}_{tr}^{(i)}$. For each instance $D \in \mathbf{D}_{tr}^{(i)}$, the first firing order of input neurons of eSNN_i is calculated given values of attributes of D. Next, a new candidate output neuron n_c is created and the weights of its synapses to all input neurons are initialized according to the obtained firing order of input neurons in **NI**. Please note that the synapses' weights between n_c and input neurons that do not fire are set to 0. Additionally, n_c is assigned the target value of the instance D.

After the candidate n_c is initialized, Algorithm 2 finds such an output neuron n_s in the repository **NO** for which the Euclidean distance calculated between vectors of synapses' weights $\mathbf{w}_{n_c}$ and $\mathbf{w}_{n_s}$ is the smallest. If the distance is less than or equal to $simTr \cdot \overline{Dist}$ threshold, then the vector of synapses weights of $\mathbf{w}_{n_s}$ is updated according to Eq. (1) and update counter M_{n_s} is incremented. Eventually, candidate n_c is discarded. Otherwise, if the distance $Dist_{n_c,n_i}$ is greater than $simTr \cdot \overline{Dist}$ threshold or there are no output neurons in **NO**, then n_c is inserted to **NO**.

After training of each eSNN in the ensemble, the prediction of a target value for each instance in the testing dataset $\mathbf{D}_{ts}$ is conducted. Each testing instance $D \in \mathbf{D}_{ts}$ is encoded by all eSNNs in the ensemble. The final predicted value assigned to D is the average value of all individual output values returned by each eSNN. In order to obtain the prediction for instance D by each individual eSNN network, D is first

Algorithm 2 Bagging Ensemble eSNN Algorithm

Input: $\mathbf{D}_{tr}, \mathbf{D}_{ts}, NI_{size}, mod, simTr, N$

for $i \in 1 \ldots N$ **do** ▷ *Teach each eSNN in the ensemble separately*
  $\mathbf{D}_{tr}^{(i)}$ - randomly generate i-th bootstrap sample from $\mathbf{D}_{tr}$
  Initialize input layer of $eSNN_i$
  for each training instance $D \in \mathbf{D}_{tr}^{(i)}$ **do**
    InputLayerEncoding(D) ▷ *Algorithm 1*
    Create a candidate output neuron n_c
    for each $A \in \mathbf{A}$, **each** $n_j \in \mathbf{NI}^{(A)}$ of eSNN_i **do**
      Create a synapse between $n_j \in \mathbf{NI}^{(A)}$ and n_c; $w_{n_j n_c} \leftarrow 0$
      if n_j fires **then** $w_{n_j n_c}^{(A)} \leftarrow w_{n_j n_c}^{(A)} + mod^{order_{n_j}}$ **end if**
    end for
    if $|\mathbf{NO}| > 0$ **then**
      for each $n_i \in \mathbf{NO}$ **do** $Dist_{n_c,n_i} \leftarrow distance(\mathbf{w}_{n_c}, \mathbf{w}_{n_i})$ **end for**
      $n_s \leftarrow$ an output neuron in **NO** such that $Dist_{n_c,n_s} = min\{Dist_{n_c,n_i} \mid n_i \in \mathbf{NO}\}$
    end if
    if $|\mathbf{NO}| > 0 \wedge Dist_{n_c,n_s} \leq simTr \cdot \overline{Dist}$ **then**
      $\mathbf{w}_{n_s} \leftarrow (\mathbf{w}_{n_c} + M_{n_s} \cdot \mathbf{w}_{n_s})/(M_{n_s} + 1)$
      $v_{n_s} \leftarrow (v_{n_c} + M_{n_s} \cdot v_{n_s})/(M_{n_s} + 1)$; $M_{n_s} \leftarrow M_{n_s} + 1$
    else
      Insert n_c to **NO**;
    end if
  end for
end for
for each testing instance $D \in \mathbf{D}_{ts}$ **do**
  for $i \in 1 \ldots N$ **do** ▷ *For each eSNN in the ensemble*
    InputLayerEncoding(D) ▷ *Algorithm 1*
    for each $A \in \mathbf{A}$, $n_j \in \mathbf{NI}^{(A)}$, $n_i \in \mathbf{NO}$ **do**
      if n_j fires **then** $PSP_{n_i} \leftarrow PSP_{n_i} + w_{n_j n_i}^{(A)} \cdot mod^{order_{n_j}}$ **end if**
    end for
    $\mathbf{NO}_{max} \leftarrow \{n_i \mid n_i \in \mathbf{NO} \wedge PSP_{n_i}$ is maximal$\}$
    $Value^{(i)} \leftarrow (\sum_{n_i \in \mathbf{NO}_{max}} v_{n_i} \cdot M_{n_i})/(\sum_{n_i \in \mathbf{NO}_{max}} M_{n_i})$
  end for
  $Value_D \leftarrow (\sum_{i=1}^{N} Value^{(i)})/N$ ▷ *Prediction value assigned to instance D*
end for

encoded into the firing order of input neurons of this eSNN. Afterwards, PSP values of all output neurons are calculated and the set of output neurons having the same maximal values of PSP is remembered as $\mathbf{NO}_{max}$. According to the proposed Eq. (4), eSNN returns an output value being a weighted average of output values of output neuron $n_i \in \mathbf{NO}_{max}$. The weights of output values are equal to the update counters of output neurons $n_i \in \mathbf{NO}$ (update counter of an output neuron reflects the number of candidate output neurons merged with that output neuron, and thus can be perceived as a significance of an output neuron in **NO**). Our algorithm implementation is available at the GitHub repository.[3]

6. Analysis of computational and space complexity

Since we are unaware of any other work that attempted to analyze the computational and space complexity of eSNNs' learning and prediction algorithms, we provide such analysis herein. We also analyze computational & space complexity of the proposed eSNNs bagging ensemble model.

6.1. eSNN computational complexity analysis

Computational complexity of the learning phase. Before the learning and prediction phases begin, the input layer of eSNNs needs to be initialized. Thus, we start the analysis by assessing the computational complexity of the network's initialization step.

The computational complexity of the input neurons' allocation step for each attribute $\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}$ is linearly dependent on the number of input neurons NI_{size} (the user-given parameters) as well as the dataset size $|\mathbf{D}_{tr}^{(i)}|$ and equals to $O(|\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}|) \cdot (|\mathbf{D}_{tr}^{(i)}| + NI_{size})$ (this follows from the fact that each attribute in $\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}$ has its own group of input neurons). For the nominal attributes $\mathbf{A}^{(n)}$, Algorithm 1 requires a single scan of each dataset sample. Thus, the computational complexity equals $O(|\mathbf{A}^{(n)}| \cdot |\mathbf{D}_{tr}^{(i)}|)$.

Given the above, the computational complexity of the **initialization** of input layer, O_{init}, of each eSNN network in the ensemble equals to:

$$O_{init}(|\mathbf{A}^{(n)}| \cdot |\mathbf{D}_{tr}^{(i)}| + |\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}| \cdot (|\mathbf{D}_{tr}^{(i)}| + NI_{size})). \quad (5)$$

After the input layer of each eSNN is initialized, each dataset instance is encoded into spikes and used to create a candidate output neuron that is either added to the repository **NO** or merged with one of the output neurons already existing there as presented in Section 4. The computational complexity of these steps for all $1 \ldots N$ eSNNs in the ensemble is:

$$O_L(N \cdot |\mathbf{D}_{tr}^{(i)}| \cdot (O_e + O_l)), \text{ where} \quad (6)$$

$$O_e(|\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}| \cdot NI_{size} + |\mathbf{A}^{(n)}| \cdot K), \quad (7)$$

$$O_l(|\mathbf{NO}| \cdot (|\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}| \cdot NI_{size} + |\mathbf{A}^{(n)}| \cdot K)). \quad (8)$$

In Formulae (6)–(8), by K we denote the average number of distinct values of attributes $\mathbf{A}^{(n)}$, while $|\mathbf{NO}|$ is the average number of output neurons in the repository **NO**.

Computational complexity of the prediction phase. The computational complexity of the prediction step is highly affected by the fact that each output neuron in **NO** is connected to each input neuron in the internal layer **NI**.

In order to make a prediction, each eSNN in the ensemble calculates the PSP value given the encoded dataset's instance D and selects the output value of an output neuron $n_i \in \mathbf{NO}$ whose PSP value is the greatest. Similarly to the learning phase, the prediction needs to be made for each testing instance $D \in \mathbf{D_{ts}}$ and separately for each eSNN. Thus, we can denote the computational complexity of the prediction phase as:

$$O_P(N \cdot |\mathbf{D}_{ts}| \cdot (O_e + O_p)), \text{ where} \quad (9)$$

$$O_e(|\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}| \cdot NI_{size} + |\mathbf{A}^{(n)}| \cdot K), \quad (10)$$

$$O_p(|\mathbf{NO}| \cdot |\mathbf{NI}|). \quad (11)$$

In Formula (11), $|\mathbf{NI}|$ denotes the total number of input neurons in each eSNN.

6.2. Space complexity analysis

The space complexity depends on the number of input and output neurons in each eSNN of the ensemble. Thus, the space complexity of an input layer of a single eSNN can be assessed as $O_{in}(|\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}| \cdot NI_{size} + |\mathbf{A}^{(n)}| \cdot K)$. Subsequently, the space complexity of the output layer of eSNN equals to $O_{ou}(|\mathbf{D}_{tr}^{(i)}|)$. The O_{ou} space complexity represents the worst case when each candidate output neuron created for a dataset instance $D \in \mathbf{D}_{tr}^{(i)}$ is added to the repository **NO** as a new output neuron. Thus, the space complexity of the ensemble of $1 \ldots N$ eSNNs equals:

$$O_S(N \cdot (|\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}| \cdot NI_{size} + |\mathbf{A}^{(n)}| \cdot K + |\mathbf{D}_{tr}^{(i)}|)).$$

7. Experiments

In this section, we describe the dataset selected for the experiments and the preprocessing we apply. Subsequently, we describe the selected error and agreement measures that are used to assess air pollution prediction quality. Next, we describe the experimental setup and provide the values of the parameters for which we obtained the best results. Finally, we provide the results of our experiments.

7.1. Air pollution dataset for warsaw

To verify our approach experimentally, we used the PM_{10} air pollution and weather dataset for Warsaw. The similar dataset was previously applied in the other studies (see, for example, Siwek and Osowski (2012)) that compared air pollution prediction results for several types of predictors. The dataset consists of 1096 daily air pollution observations collected over years 2006–2008 as well as several weather-related attributes: temperature, precipitation, humidity, wind speed and cloudiness. The dataset can be obtained from the website of the Chief Inspectorate of Environmental Protection of Poland (GIOS, 2015). Similarly to the results presented in Siwek and Osowski (2012), we applied our proposed ensemble for 1-day ahead prediction of air pollution.

In Fig. 5, we present the basic characteristics of the dataset. The following attributes **A** of the dataset (selected based on a significance of the correlation with the pollution level) are applied in the learning process. Continuous attributes $\mathbf{A}^{(c)}$: air pollution values observed in the current and two previous days as well as the temperature on the current day. The set of nominal attributes $\mathbf{A}^{(n)}$ consists of one attribute representing a day of the week (encoded as a number from 1 to 7). As mentioned previously, we denote acquired dataset as $\mathbf{D}_o$.

7.2. Air pollution decomposition

In order to obtain dataset **D** containing decomposed air pollution time series from $\mathbf{D}_o$, we use the CEEMDAN method with the maximum number of generated components set to 5. This is motivated by the fact that each subsequent component generated by CEEMDAN is usually less correlated with the decomposed air pollution time series and by including such weakly correlated components in the training data we can decrease the prediction quality.

In Fig. 6, we presented the original air pollution time series dataset and its generated components.

[3] https://github.com/piotrMaciag32/Bagging-eSNN-Prediction.

Mean and standard deviation of attributes

	PM_{10}	Temp.	Peri.	Hum.	Wind speed	Cloud.
mean	31.2	10.7	1.55	78.8	1.92	5.98
sd	20.06	8.53	3.73	13.3	0.95	2.2

Linear correlation coeff.

	PM_{10}	Temp.	Peri.	Hum.	Wind speed	Cloud.
PM_{10}	1.0	-0.26	-0.15	-0.02	-0.28	-0.1

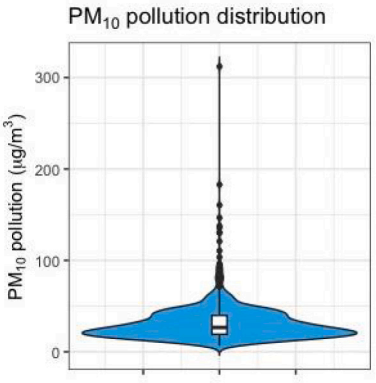


Fig. 5. Parameters of the attributes of the acquired dataset and linear correlation coefficient between continuous attributes and the distribution of the pollution values.

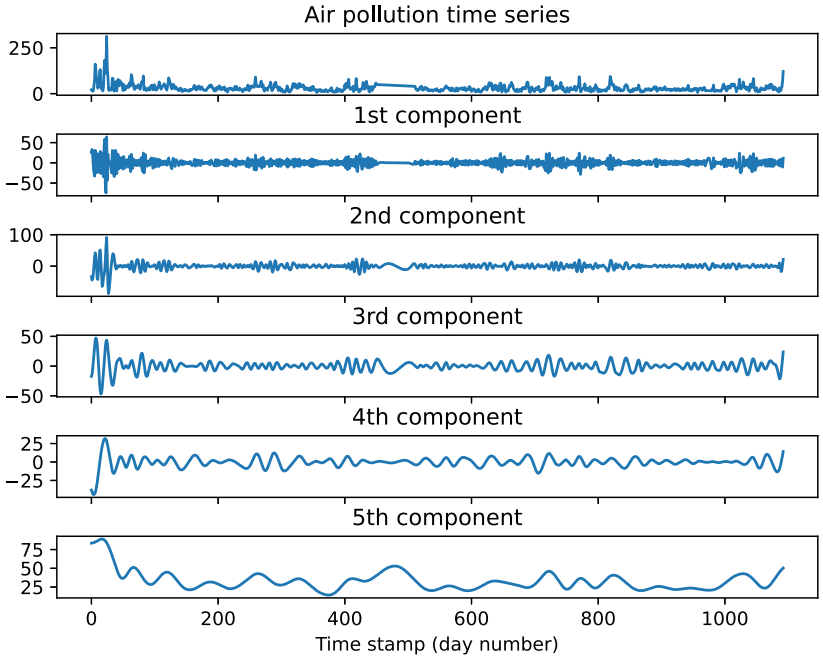


Fig. 6. The decomposed PM_{10} pollution time series data using CEEMDAN and 5 main components.

Table 2
Linear correlation coefficient between air pollution time series and its 5 components.

	1st component	2nd component	3rd component	4th component	5th component
PM_{10} air pollution	0.38	0.44	0.43	0.27	0.55

The linear correlation coefficient between the air pollution time series and its components are given in Table 2.

The obtained PM_{10} components are combined with dataset $\mathbf{D}_o$ in order to obtain dataset $\mathbf{D}$. The dataset $\mathbf{D}$ is subsequently randomly split into training $\mathbf{D}_{tr}$ and testing parts $\mathbf{D}_{ts}$ in proportion 7:3.

7.3. Selected error and agreement measures

For the assessment of the quality of prediction we selected the following three error measures for which the lower value indicates better prediction results: Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). Definitions of these error measures are presented in Eqs. (13)–(14). To assess the agreement between the predicted and observed values, the Index of Agreement, IA (also known as Willimott index) and r measures are applied (we presented this measure in Eqs. (15)–(16)). The higher values of the agreement measures indicate better prediction results. In Eqs. (13)–(14) and (15)–(16), n denotes the number of testing instances in the testing dataset $\mathbf{D}_{ts}$, o_i and p_i are a target value (observed air pollution value) and a predicted value of an ith testing instance, respectively.

The values of the MAE and RMSE errors are in the range $[0, +\infty]$, while the values of the MAPE error are in percent. The values of the IA agreement measure are in the range $[0, 1]$, where 0 implies no agreement between observed and predicted values, while 1 implies that the predicted values are equal to the observed values. The values of the r coefficient are in the range $[0, 1]$, where 0 indicates the absence of a linear correlation between the predicted and observed values and 1 indicates the strongest positive linear correlation between the predicted and observed values (each predicted value equals the respective observed value).

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |o_i - p_i|}{n}. \tag{12}$$

Table 3
Tested and selected parameters for each used model (except for the LR and EN models that do not require any input parameters). The predictors' parameters not listed beneath were set to their default values.

Predictor	Tested parameters	Selected parameters
eSNN	N = {1, 5, 10, …, 60}; simTr = {0.01, 0.011, …, 0.025}; NI_{size} = {50, 55, …, 100}	$NI_{size} = 75, simTR = 0.023$
SVR	C = {1, 2, …, 15}; kernel = {*linear*, *rbf*}	$C = 9$, kernel = *linear*
MLP	hidden-layer-size = {100, 200, 300}; max-iter = {500, 750, 1000}; α = {0.0001, 0.0004, 0.0007, 0.001} learning-rate-init = {0.001, 0.004, 0.007, 0.01}	hidden-layer-sizes = 200, max-iter = 1000, $\alpha = 0.0007$, learning-rate-init = 0.001
LSTM	batch-size = {10, 20, …, 50}; epoch = {100, 200, …, 500}	epoch = 100, batch-size = 20, lstm-layer-hidden-units = 100
SNNTorch	SNNs neurons type = {single LIF layer, layer of synaptic neurons, double layer of LIF neurons}	SNNs neurons type = single LIF layer

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(o_i - p_i)^2}{n}}. \tag{13}$$

$$\text{MAPE} = \frac{\sum_{i=1}^{n}\left|\frac{o_i - p_i}{o_i}\right|}{n} \cdot 100. \tag{14}$$

$$\text{IA} = 1 - \frac{\sum_{i=1}^{n}(o_i - p_i)^2}{\sum_{i=1}^{n}(|p_i - \overline{o}| + |o_i - \overline{o}|)^2}. \tag{15}$$

$$r = \frac{n\sum_{i=1}^{n} o_i p_i - \sum_{i=1}^{n} o_i \sum_{i=1}^{n} p_i}{\sqrt{\left[n\sum_{i=1}^{n} p_i^2 - (\sum_{i=1}^{n} p_i)^2\right]\left[n\sum_{i=1}^{n} o_i^2 - (\sum_{i=1}^{n} o_i)^2\right]}}. \tag{16}$$

7.4. Obtained air pollution prediction results

In Table 4, we show the results of the comparison of the prediction results obtained for the selected prediction models, namely:

- Single models of Linear Regression (LR), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Long-short Term Memory (LSTM)[4] network, Elastic Network (EN) and the SNNTorch SNNs implementation. For LR, SVM, MLP and EN the `sklearn` package is used, while for LSTM, we created a model using the `Keras` package.
- The above models taught using original data incorporating the CEEMDAN decomposition.
- The proposed stacking ensemble that in the first layer includes: LR, SVM, MLP, LSTM and EN, and in the second layer includes either the bagging eSNNs or a single predictor from the set of the predictors used in the first layer.

In the performed experiments, we conducted grid search procedure in order to find the best parameters of each selected model (except for the LR and EN models that do not require any input parameters). The sets of tested parameters and the selected best parameters are presented in Table 3.

In the case of SNNTorch network, the parameters other than the type of hidden neurons were set as follows:

- PSP leak value - 0.9 for LIF neurons and 0.8 for synaptic neurons.
- Learning rate - 0.001.
- Optimization algorithm - Adam.

As it can be noted from Table 4, the proposed model achieves the best prediction results when it is combined with 30 eSNNs organized in the bagging ensemble. The obtained prediction results are visualized by us in Fig. 7 using Taylor diagram. For each model, the diagram shows the linear correlation coefficient between real and predicted pollution value, the standard deviation of predicted values as well as the RMSE error. The diagram clearly shows two separate prediction groups: the first one when no decomposition is used, and the second one when the CEEMDAN components are incorporated into training and testing data. For the second group, the bagging ensemble of 30 eSNNs achieves superior results in terms of the correlation coefficient and the RMSE error.

In Fig. 8, we present the scatter plots of the observed pollution values and the predicted pollution values for the testing dataset $\mathbf{D}_{ts}$ as well as the parameters of the fitted linear regression and the linear correlation coefficient r. The presented four plots were obtained for the ensemble consisting of CEEMDAN, stacking and 1, 5, 15, 30 bagging eSNNs, respectively. It can be noticed that the use of bagging ensembles of eSNN improves the quality of prediction when compared to a single model. The violin plots illustrating the prediction results are given in Fig. 9.

Fig. 10 presents the obtained values of the error and agreement measures (in the form of a heatmap) for the changing number of eSNNs in the second layer of the ensemble (parameter N) and the eSNN's similarity threshold $simTr$ parameter. Similar results are presented in Fig. 11 for the changing number of eSNNs and the NI_{size} parameter. It can be observed from both figures that the use of as few as five eSNNs can substantially improve the error and agreement indicators when compared to a singleton eSNN. Moreover, the indicators are improving even more for the greater values of the N parameter.

7.5. Discussion

The obtained prediction results and the conducted comparison of models suggest that applying CEEMDAN decomposition method to air pollution time series data can allow us to substantially improve prediction results. This is inline with the previously reported results, such as the results of Siwek and Osowski (2016), where wavelet decomposition was applied.

Our experiments also suggest that by employing a stacking ensemble of various predictors can improve prediction quality even further. In particular, we were able to obtain slightly better prediction results when the second layer of the proposed ensemble consisted of bagging eSNNs. This can be explained by the fact, that such bagging ensemble is trained solely using data highly positively correlated with the target air pollution values. Previous work suggest that eSNNs tend to underperfrom when the training dataset contains attributes weakly correlated with target values, but can outperform other models when such attributes are entirely eliminated from the dataset.

8. Conclusions and future work

This article introduced a new approach to air pollution prediction using the ensembling method consisting of: (i) CEEMDAN air pollution time series decomposition, (ii) stacking ensemble that in the first layer contains a set of predictors, each of which making its own air pollution prediction and in the second layer contains a bagging ensemble of eSNNs. To this end, we proposed and implemented the bagging eSNNs algorithm that randomly selects a specified number of samples from a given training dataset, each of which is next used to teach a distinct eSNN in the ensemble.

The prediction results obtained with the proposed approach were compared with the prediction results obtained from other selected

[4] The network consisted of one LSTM layer followed by one Dense layer of a neuron making prediction.

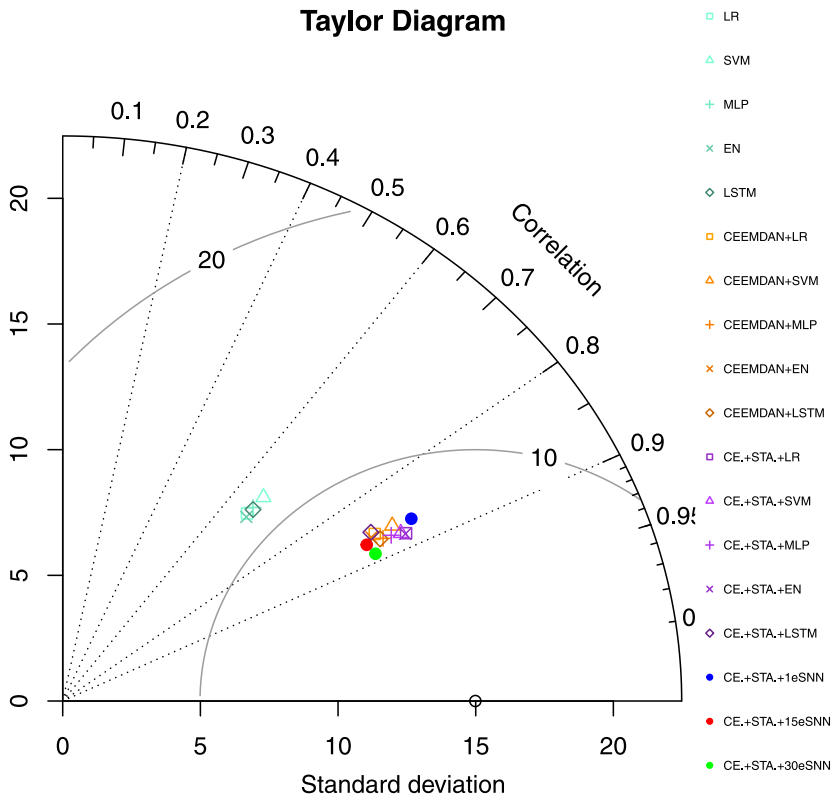


Fig. 7. Taylor diagram presenting models' comparison for the prediction results for the tested models. CE. stands for CEEMDAN decomposition; STA. stands for the first layer of stacking ensemble consisting of LR, SVM, MLP, EN and LSTM predictors.

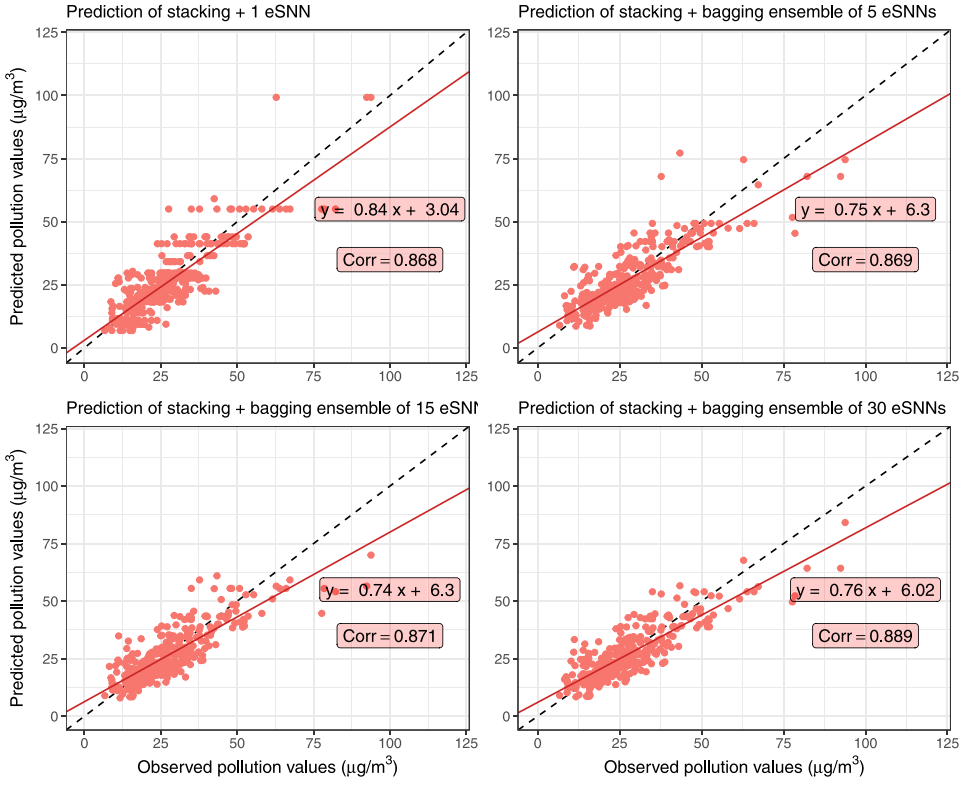


Fig. 8. Scatter plots presenting prediction results obtained for the proposed model consisting of CEEMDAN, stacking and, respectively, a single eSNN as well as bagging ensembles consisting of 5, 15 and 30 eSNNs. All plots were obtained using the CEEMDAN air pollution decomposition and the first layer of stack consisting of LR, SVM, MLP, LSTM and EN predictors. The plots contain also fitted linear models representing dependency between the observed and predicted values as well as the values of the linear correlation coefficient.

Table 4
Comparison of the obtained air pollution prediction results for the selected predictors. First layer of stacking ensemble consists of prediction results made by LR, SVM, MLP, LSTM and EN (we denoted the set of these predictors as Stacking in the table beneath).

Predictor	RMSE [μg/m^3]	MAE [μg/m^3]	MAPE%	IA
LR	11.27	8.13	37	0.77
SVM	11.21	7.91	35	0.79
MLP	11.16	7.92	33	0.78
LSTM	11.15	8	35	0.78
EN	11.19	8.05	37	0.78
SNNTorch	13.73	8.89	35	0.78
CEEMDAN + LR	7.66	5.62	25	0.92
CEEMDAN + SVM	7.69	5.56	25	0.93
CEEMDAN + MLP	7.32	5.58	24	0.93
CEEMDAN + LSTM	7.36	5.41	23	0.93
CEEMDAN + EN	7.60	5.59	25	0.92
CEEMDAN + SNNTorch	14.53	11.21	48	0.85
CEEMDAN + Stacking + Single LR	7.16	5.42	24	**0.94**
CEEMDAN + Stacking + Single SVM	7.21	5.34	23	**0.94**
CEEMDAN + Stacking + Single MLP	7.27	5.4	23	0.93
CEEMDAN + Stacking + Single LSTM	7.78	5.55	25	0.92
CEEMDAN + Stacking + Single EN	7.14	5.4	24	**0.94**
CEEMDAN + Stacking + Single eSNNs	7.71	5.84	24	0.93
CEEMDAN + Stacking + Bagging eSNNs (15)	7.42	5.39	22	0.92
CEEMDAN + Stacking + Bagging eSNNs (30)	**6.91**	**5.14**	**21**	**0.94**

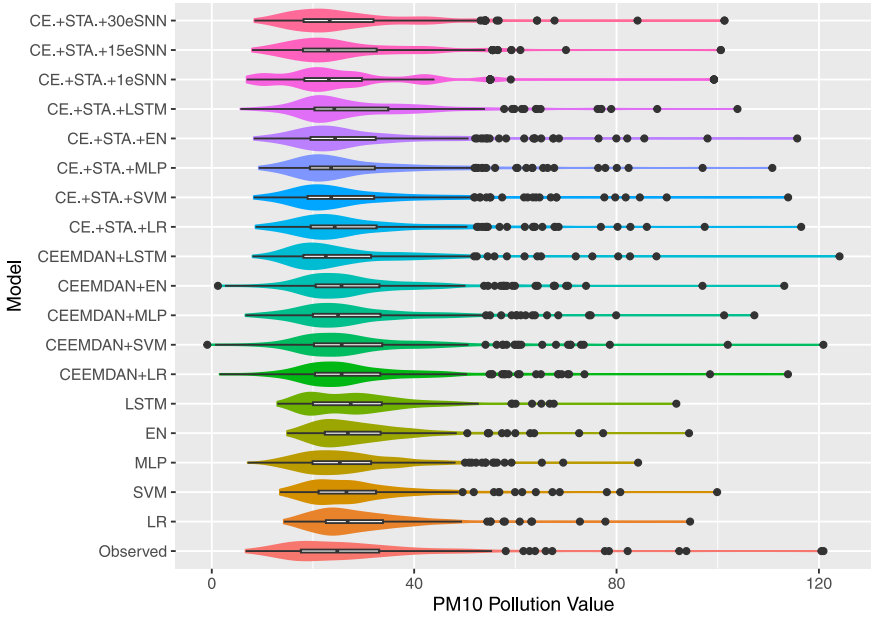


Fig. 9. Violin plots of real pollution values (observed) and predictions of the best performing prediction models tested in the experiments.

state-of-the-art predictors. As the results of the experiments showed, the proposed ensemble can provide substantially better prediction quality than the tested singleton models. Moreover, as the results of the experiments suggest, to the advantages of the proposed stacking ensemble model that uses bagging eSNNs provide more stable error and agreement measures when a range of eSNNs internal parameters is tested than when a singleton eSNN is applied. In particular, for the changing values of the parameter $simTR$, the variance of obtained measures/agreement values is much higher for a singleton eSNN than when a bagging ensemble of eSNNs is used.

In this work, we also analyzed the computational and space complexity of the eSNNs' learning and prediction algorithms. To the best of our knowledge, this is the first time when such an analysis is done. Also, we proposed a formula for calculating the tight upper bound on the Euclidean distances between synapses weights of any possible candidate output neuron and any output neuron that simplifies the selection of a similarity threshold parameter of eSNN and can be applied when training and testing datasets consist of continuous, ordinal and nominal attributes.

The work discusses the encoding methods applied to eSNNs and (in Appendix) provides the new encoding method that (i) calculates a histogram of input values of each ordinal or continuous feature and (ii) allocates the GRFs fields and input neurons encoding each attribute according to the calculated histogram of values. This allows for more effective encoding of the values of unevenly distributed features, such as highly skewed features when compared to the GRFs method used previously in the literature.

The study in its current form posses some limitations that should be addressed in the future studies. First, we would expect to conduct more extensive experiments using datasets of the other geographical locations. Second, one could expect to test different types of ensemble models with eSNNs (for example, boosting ensembles). The other potential recommendation for the future research could be applying one of the nature-inspired algorithms mentioned in Section 1 in order to search for the best parameters of eSNNs networks.

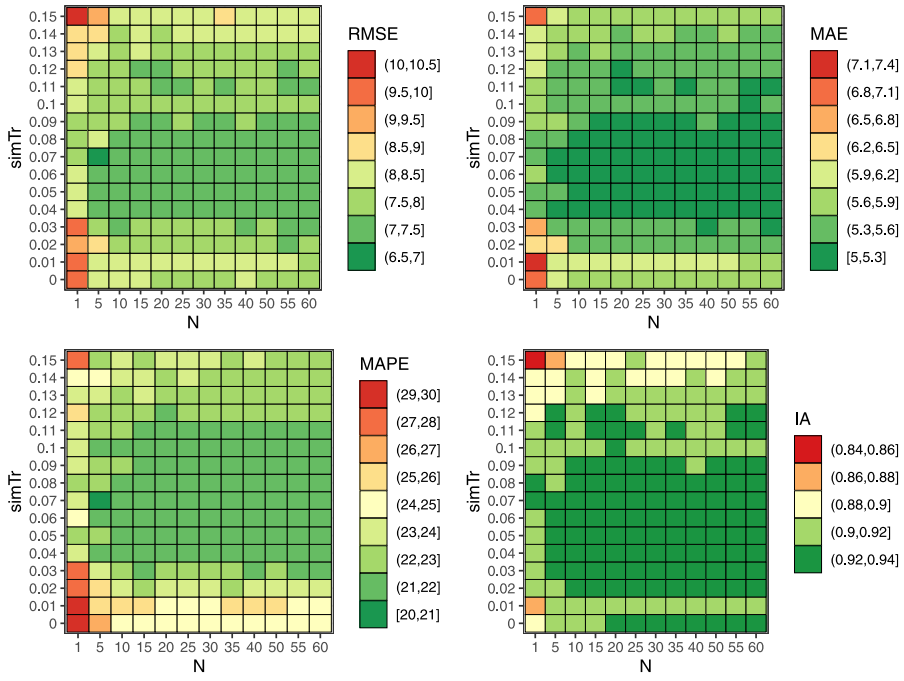


Fig. 10. The error and agreement values for the proposed ensemble model for the changing values of N (the number of eSNNs in the bagging ensemble), $simTr$ (similarity threshold of eSNN) and constant NI_{size} (number of input neurons encoding attributes $\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}$). The plots were obtained for the fixed $NI_{size} = 50$ eSNNs parameter. Please note that MAPE is given in percent.

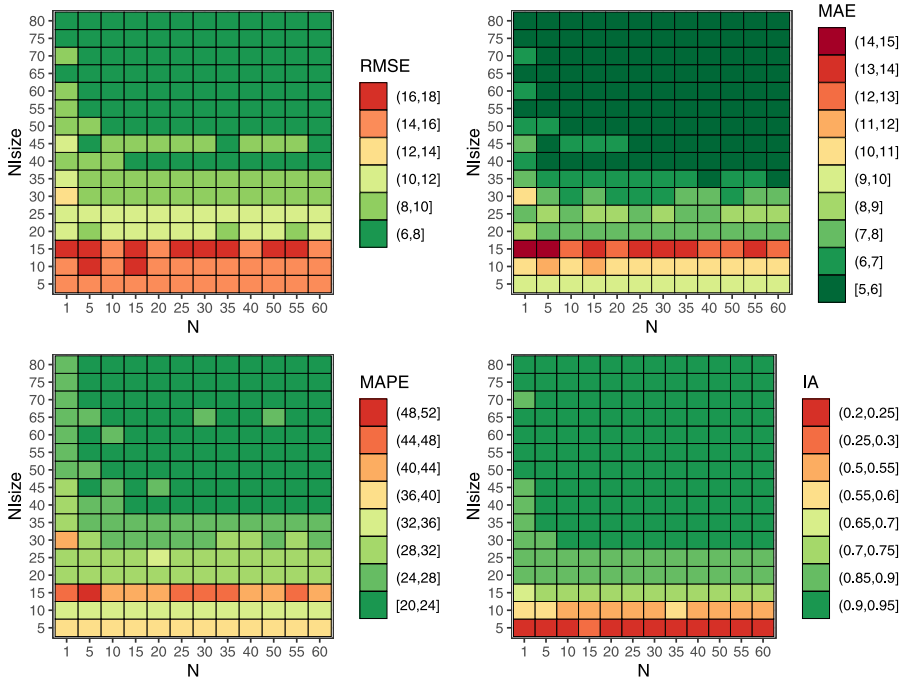


Fig. 11. The error and agreement values for the changing values of N (number of eSNNs in the ensemble) and NI_{size} (the number of input neurons in eSNN encoding attributes $\mathbf{A}^{(c)} \cup \mathbf{A}^{(o)}$). The plots were obtained for the fixed $simTr = 0.02$. Please note that MAPE is given in percent.

Nevertheless, we believe that the proposed ensemble model can be successfully used in other prediction tasks, such as prediction of electricity consumption or stock prices forecasting.

Software and Data availability

- **Name of the software**: Bagging-eSNNs
- **Author**: Piotr Maciąg, Warsaw University of Technology, Institute of Computer Science
- **Contact Information**: piotr.maciag@pw.edu.pl
- **Year of first available**: 2021
- **Programming language**: C++
- **Availability**: https://github.com/piotrMaciag32/Bagging-ensemble-eSNN
- **Cost**: free, **License**: GPLv3
- **Software required**: C++ v 14 compiler, CMake compatible environment

- **Hardware required**: Software can be run on a standard personal computer
- **Data**: datasets used for experiments are provided in the GitHub repository along the software

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The used data is shared along the software published on GitHub. Please refer to software availability declaration.

Acknowledgments

J. L. Lobo and J. Del Ser receive funding support from the Basque Government, Spain through grants KK-2023/00012 (BEREZ-IA) and IT1456-22 (MATHMODE).

Appendix. A new GRFs-based encoding method

Many types of the input data encoding methods for SNNs have been proposed and successfully applied in various domains. Herein, we first review the GRFs encoding method and subsequently provide our new encoding method for SNNs networks.

A.1. Gaussian receptive fields encoding method

One of the most commonly used encoding method is population-based encoding that applies Gaussian Receptive Fields (GRFs) in order to obtain precise spiking times of input neurons for a given input stimuli. An example of a population-based coding that uses GRFs is presented in Fig. 12. The values I^A_{min} and I^A_{max} are the minimal and the maximal values of a continuous or ordinal attribute A in a dataset, respectively, and they should be calculated in the input layer initialization step. NI_{size} is the number of GRFs (and their respective input neurons) that is specified by the user. Each GRF ($j = 1, \ldots, NI_{size}$) is used to obtain excitation value for an input value $D[A]$ according to Eq. (17). In Eq. (17), μ_j is a mean value of each GRF that is calculated according to Eq. (18), while σ is a standard deviation of each GRF that is calculated according to Eq. (19).

$$Exc^{(A)}_j = exp\Big(-\frac{1}{2}\big(\frac{D[A]-\mu^{(A)}_j}{\sigma}\big)^2\Big) \tag{17}$$

$$\mu_j = I^{(A)}_{min} + (j-0.5)\cdot width^{(A)}, \tag{18}$$

where $j = 1, \ldots, NI_{size}$ and $width^{(A)} = \frac{I^{(A)}_{max} - I^{(A)}_{min}}{NI_{size}}$. $width^{(A)}$ is a distance between mean values of GRFs.

$$\sigma = \frac{width^{(A)}}{\beta}. \tag{19}$$

β is a user given parameter that controls the standard deviation of each GRF. Given the calculated excitation values of GRFs, the firing of input neurons $j = 1, \ldots, NI_{size}$ can be obtained as follows:

$$FT_{n_j} = 1 - Exc_j(D[A]). \tag{20}$$

Let us consider an example GRFs coding given in Fig. 12. First, let us assume that the value of an attribute A to be encoded is $D[A] = 6.5$ and the minimal and maximal values of this attribute in the dataset are $I^{(A)}_{min} = 0, I^{(A)}_{max} = 10$, respectively. Additionally, let us assume that we apply $NI_{size} = 5$ input neurons to encode values of A and $\beta = 2$. The other GRFs parameters will be calculated as follows:

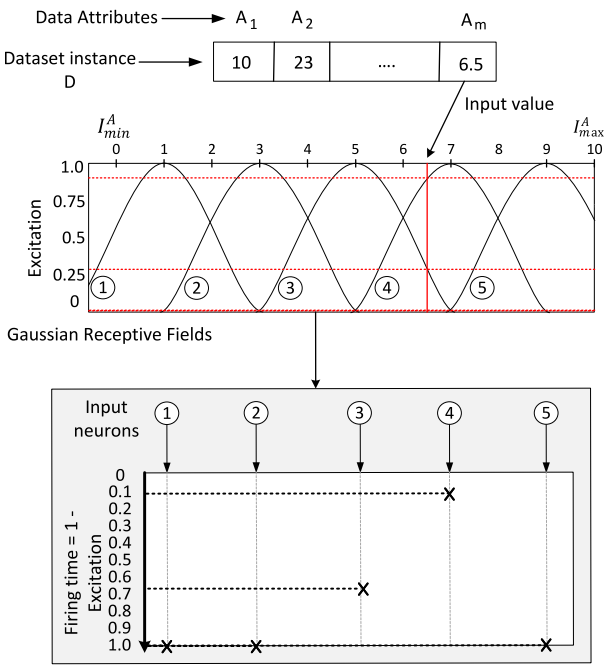


Fig. 12. The GRFs encoding method as explained in, for example, Kasabov (2014).

- The $width^{(A)} = \frac{10-0}{5} = 2$.
- The center values of GRFs calculated according to Eq. (18) are as follows: $\mu_1 = 1, \mu_2 = 3, \mu_3 = 5, \mu_4 = 7, \mu_5 = 9$.
- σ value of Eq. (19) equals to 1.
- GRFs excitation values for $D[A] = 6.5$ are as follows:
  - $Exc_1(6.5) = 2.69958E{-}07$,
  - $Exc_2(6.5) = 0.002187491$,
  - $Exc_3(6.5) = 0.324652467$,
  - $Exc_4(6.5) = 0.882496903$,
  - $Exc_5(6.5) = 0.043936934$.
- Thus, the respective firing times of input neurons are as follows:
  - $FT_{n_1}(6.5) = 1$,
  - $FT_{n_2}(6.5) = 0.997812509$,
  - $FT_{n_3}(6.5) = 0,675347533$,
  - $FT_{n_4}(6.5) = 0,117503097$,
  - $FT_{n_5}(6.5) = 0,956063066$.

A.2. Proposed encoding method

The already developed encoding methods that apply GRFs are prone to numerical errors and may not be adequate for datasets in which values of some or all attributes are of non-uniform distributions, such as normal or highly skewed distributions. Let us consider a histogram of values of the attribute A presented in Fig. 13 and the allocation of GRFs as explained in the above example. One may notice that the number of the GRFs allocated to the range $[0, 4]$ is the same as the number of GRFs allocated to the range $[6, 10]$ despite the fact that the former range has many more values than the latter one.

Thus, herein we introduce our method that allocates GRFs (and their respective input neurons) based on a histogram of input values of a given attribute. The method introduced here extends the encoding method previously introduced in Maciąg et al. (2022). However, unlike the method of Maciąg et al. (2022) that is suitable for the calculation of only firing order of input neurons based on a given input value, the method provided here allows us to calculate exact firing times of

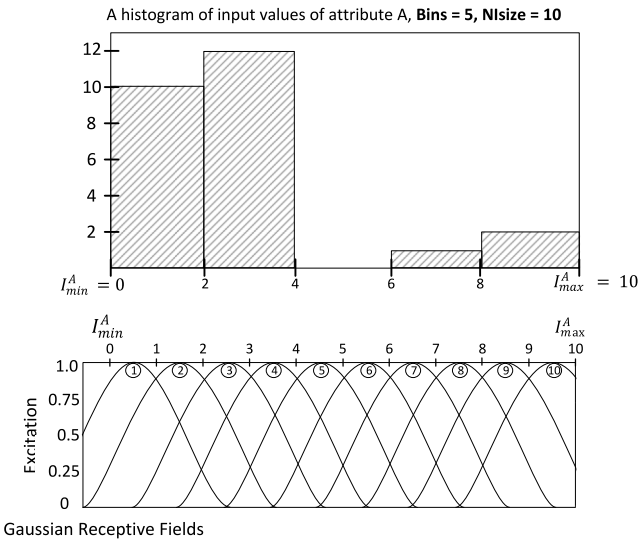


Fig. 13. A highly skewed distribution of input values of the attribute A and a uniform allocation of GRFs ($NI_{size} = 10$, the number of bins in the histogram is 5).

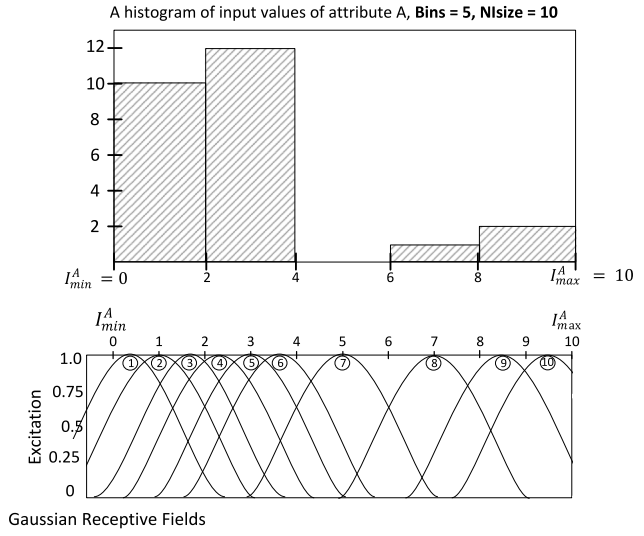


Fig. 14. A highly skewed distribution of input values and GRFs allocated according to the method presented in this appendix.

input neurons for a given input value. To the distinctive features of the method provided here belongs the fact that GRFs are not allocated uniformly over the range $[I^{(A)}_{min}, I^{(A)}_{max}]$ of values, but according to the actual distribution of values of attribute A and the calculated histogram of these values.

In addition to NI_{size} (GRFs number) and β (a single GRF width), parameters typically required by GRFs, our allocation method requires user to specify one more parameter: B - the number of bins in the histogram to be generated for values of attribute A. The number of GRFs NI_{size} should be equal to or greater than the number of bins B. In our method, first for the range of values of each bin a single GRF is allocated regardless of the number of values of A in that bin. The rest of $NI_{size} - B$ GRFs (and their respective neurons) are allocated proportionally to the number of values in each bin.

The width of each bin for the range of values of attribute A equals:

$$Bins_{width} = \frac{I^{(A)}_{max} - I^{(A)}_{min}}{B} \tag{21}$$

Thus, the range of values in each bin is as follows:

Table 5
The input and calculated parameters of our GRFs allocation method for the dataset provided in Fig. 14.

Input parameters	Values
B (no. bins)	5
NI_{size}	10
β	2
Calculated parameters	**Values**
$width$	2
$Bin_i.\Delta$	$Bin_1.\Delta = 1, Bin_2.\Delta = 0.67,$ $Bin_3.\Delta = 2, Bin_4.\Delta = 2, Bin_5.\Delta = 1$
$Bin_i.\overline{\overline{GRFs}}$	$Bin_1.\overline{\overline{GRFs}} = 2, Bin_2.\overline{\overline{GRFs}} = 3,$ $Bin_3.\overline{\overline{GRFs}} = 1, Bin_4.\overline{\overline{GRFs}} = 1, Bin_5.\overline{\overline{GRFs}} = 2$
μ_j	$\mu_1 = 0.(3), \mu_2 = 1, \mu_3 = 0.(6),$ $\mu_4 = 2.(3), \mu_5 = 3, \mu_5 = 2.(6), \mu_7 = 5,$ $\mu_8 = 7, \mu_8 = 8.5, \mu_10 = 9.5$
σ	1

- $\left[I^{(A)}_{min} + (i-1)\cdot Bin_{width}, I^{(A)}_{min} + i \cdot Bins_{width}\right)$, for $Bin_i = 1, \ldots, B-1$.
- $\left[I^{(A)}_{min} + (B-1)\cdot Bin_{width}, I^{(A)}_{max}\right]$ for Bin_B.

Let $Bin_i.\overline{\overline{Values}}$ be the number of values that are in Bin_i of the generated histogram. In addition to one GRF always allocated for each Bin, the allocation of additional number of GRFs is calculated according to Proposition 2.

Proposition 2 (*Adapted from Maciąg et al. (2022)*). *Let $Bin_i.\overline{\overline{GRFs}}, i = 1, \ldots, B$ represents the number of GRFs to be allocated to encode the values of Bin_i and let $\overline{\overline{\mathbf{D}}}$ be the number of a dataset's instances in* **D**.[5]

$$Bin_i.\overline{\overline{GRFs}} = round\left(\frac{Bin_i.\overline{\overline{Values}}}{\overline{\overline{\mathbf{D}}}} \cdot (NI_{size} - B)\right) + 1, for\ i = i \in \{1, \ldots, B\}.$$

Please note that the number of GRFs allocated according to Proposition 2 may not necessarily be equal to NI_{size}. After obtaining the number of GRFs allocated to each Bin of the histogram, first we calculate the center value μ_i of each GRF.

To this end, the *width* between center values of GRFs for each bin is calculated according to Eq. (22).

$$Bin_i.\Delta = \frac{Bins_{width}}{Bin_i.\overline{\overline{GRFs}}} \tag{22}$$

Subsequently, we calculate the center value of each GRF as follows. For bins $i \in \{1, \ldots, B\}$, the center value of GRF, $GRF_j \in Bin_i.GRFs, j \in \{1, \ldots, Bin_i.\overline{\overline{GRFs}}\}$ is calculated as follows:

$$\mu_j = Min^{(F)} + (i-1)\cdot Bins_{width} + (j - 0.5)\cdot Bin_i.\Delta. \tag{23}$$

To illustrate the proposed GRFs allocation method, let us consider the example given in Fig. 14. First, let us assume that $I^{(A)}_{min} = 0, I^{(A)}_{max} = 10$, $Bins = 5$ and $NI_{size} = 10$. Subsequently, let us assume that the distribution of values of A is as presented in the histogram in Fig. 14. In Table 5, we present the input and the calculated values of our GRFs allocation method. After allocation of GRFs, for a given input value $D[A]$, excitations, as well as firing times of input neurons, can be calculated as presented in Eqs. (17), (20).

[5] Please note that we propose to calculate histogram over the overall **D** dataset before it is split into the training and testing parts. However, in the case of big data datasets, the parameters of our method can be calculated using only the training part of the dataset or a sample of the dataset.

References

Adnan, R.M., Dai, H.-L., mirshekari chargari, M., Al-Bahrani, M., Mamlooki, M., 2022a. Prediction of the FRP reinforced concrete beam shear capacity by using ELM-CRFOA. Measurement 205, 112230. http://dx.doi.org/10.1016/j.measurement.2022.112230, URL: https://www.sciencedirect.com/science/article/pii/S0263224122014269.

Adnan, R.M., Dai, H.-L., Mostafa, R.R., Parmar, K.S., Heddam, S., Kisi, O., 2022b. Modeling multistep ahead dissolved oxygen concentration using improved support vector machines by a hybrid metaheuristic algorithm. Sustainability 14 (6), http://dx.doi.org/10.3390/su14063470, URL: https://www.mdpi.com/2071-1050/14/6/3470.

Adnan, R.M., Mostafa, R.R., Chen, Z., Parmar, K.S., Kisi, O., Zounemat-Kermani, M., 2023a. Water temperature prediction using improved deep learning methods through reptile search algorithm and weighted mean of vectors optimizer. J. Mar. Sci. Eng. 11 (2), http://dx.doi.org/10.3390/jmse11020259, URL: https://www.mdpi.com/2077-1312/11/2/259.

Adnan, R.M., Mostafa, R.R., Dai, H.-L., Heddam, S., Kuriqi, A., Kisi, O., 2023b. Pan evaporation estimation by relevance vector machine tuned with new metaheuristic algorithms using limited climatic data. Eng. Appl. Comput. Fluid Mech. 17 (1), 2192258. http://dx.doi.org/10.1080/19942060.2023.2192258, arXiv:https://doi.org/10.1080/19942060.2023.2192258.

Adnan, R.M., Mostafa, R.R., Islam, A.R.M.T., Kisi, O., Kuriqi, A., Heddam, S., 2021. Estimating reference evapotranspiration using hybrid adaptive fuzzy inferencing coupled with heuristic algorithms. Comput. Electron. Agric. 191, 106541. http://dx.doi.org/10.1016/j.compag.2021.106541, URL: https://www.sciencedirect.com/science/article/pii/S0168169921005585.

Badyda, A., Grellier, J., 2014. Screening assessment of the burden of disease due to air pollution in eleven Polish agglomerations. Eur. Respir. J. 44 (Suppl 58).

Bell, M.L., Zanobetti, A., Dominici, F., 2014. Who is more affected by ozone pollution? A systematic review and meta-analysis. Am. J. Epidemiol. 180 (1), 15–28. http://dx.doi.org/10.1093/aje/kwu115, arXiv:https://academic.oup.com/aje/article-pdf/180/1/15/17342063/kwu115.pdf.

Bozdağ, A., Dokuz, Y., Gökçek, Ö.B., 2020. Spatial prediction of PM10 concentration using machine learning algorithms in Ankara, Turkey. Environ. Pollut. 263, 114635. http://dx.doi.org/10.1016/j.envpol.2020.114635, URL: https://www.sciencedirect.com/science/article/pii/S0269749120312306.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.

Brunekreef, B., Holgate, S.T., 2002. Air pollution and health. Lancet 360 (9341), 1233–1242. http://dx.doi.org/10.1016/S0140-6736(02)11274-8, URL: https://www.sciencedirect.com/science/article/pii/S0140673602112748.

Cabaneros, S.M., Calautit, J.K., Hughes, B.R., 2019. A review of artificial neural network models for ambient air pollution prediction. Environ. Model. Softw. 119, 285–304.

DEFRA, 2018. Department for environment food and rural affairs archive data. URL: https://uk-air.defra.gov.uk/data/data-catalogue.

Demertzis, K., Iliadis, L., Bougoudis, I., 2019. Gryphon: a semi-supervised anomaly detection system based on one-class evolving spiking neural network. Neural Comput. Appl..

EEA, 2020. European environment agency - air quality in Europe — 2020 report. URL: https://www.eea.europa.eu/publications/air-quality-in-europe-2020-report.

Eshraghian, J.K., Ward, M., Neftci, E., Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Jeong, D.S., Lu, W.D., 2021. Training spiking neural networks using lessons from deep learning. arXiv preprint arXiv:2109.12894.

Feng, X., Li, Q., Zhu, Y., Hou, J., Jin, L., Wang, J., 2015. Artificial neural networks forecasting of PM2.5 pollution using air mass trajectory based geographic model and wavelet transformation. Atmos. Environ. 107, 118–128.

Gardner, M., 1999. The Advantages of Artificial Neural Network and Regression Tree Based Air Quality Models (Ph.D. thesis). University of East Anglia.

Gardner, M., Dorling, S., 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmos. Environ. 32 (14), 2627–2636. http://dx.doi.org/10.1016/S1352-2310(97)00447-0, URL: https://www.sciencedirect.com/science/article/pii/S1352231097004470.

Gardner, M., Dorling, S., 1999. Neural network modelling and prediction of hourly NOx and NO2 concentrations in urban air in London. Atmos. Environ. 33 (5), 709–719.

Gerstner, W., Kistler, W., 2002. Spiking Neuron Models: An Introduction. Cambridge University Press, New York, NY, USA.

Gerstner, W., Kistler, W.M., Naud, R., Paninski, L., 2014. Neuronal Dynamics: from Single Neurons to Networks and Models of Cognition. Cambridge University Press.

GIOS, 2015. Chief inspectorate for environmental protection - warsaw-ursynow pollution data 2006–2008. URL: http://powietrze.gios.gov.pl/.

Guo, C., Liu, G., Chen, C.-H., 2020. Air pollution concentration forecast method based on the deep ensemble neural network. Wirel. Commun. Mob. Comput. 2020, 1–13.

Hebb, D.O., 2005. The Organization of Behavior: A Neuropsychological Theory. Psychology Press.

Kasabov, N., 1998. The ECOS framework and the ECO learning method for evolving connectionist systems. J. Adv. Comput. Intell. Intell. Inf. 2 (6), 195–202.

Kasabov, N.K., 2014. NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. Neural Netw. 52, 62–76.

Kowalski, P.A., Warchałowski, W., 2018. The comparison of linear models for PM10 and PM2. 5 forecasting. In: WIT Transaction on Ecology and the Environment, Vol. 230. pp. 177–188.

Kugele, A., Pfeil, T., Pfeiffer, M., Chicca, E., 2020. Efficient processing of spatio-temporal data streams with spiking neural networks. Front. Neurosci. 14, http://dx.doi.org/10.3389/fnins.2020.00439, URL: https://www.frontiersin.org/articles/10.3389/fnins.2020.00439.

Kukkonen, J., Partanen, L., Karppinen, A., Ruuskanen, J., Junninen, H., Kolehmainen, M., Niska, H., Dorling, S., Chatterton, T., Foxall, R., Cawley, G., 2003. Extensive evaluation of neural network models for the prediction of NO2 and PM10 concentrations, compared with a deterministic modelling system and measurements in central Helsinki. Atmos. Environ. 37 (32), 4539–4550.

Laña, I., Capecci, E., Del Ser, J., Lobo, J.L., Kasabov, N., 2018. Road traffic forecasting using NeuCube and dynamic evolving spiking neural networks. In: Del Ser, J., Osaba, E., Bilbao, M.N., Sanchez-Medina, J.J., Vecchio, M., Yang, X.-S. (Eds.), Intelligent Distributed Computing XII. Springer International Publishing, Cham, pp. 192–203.

Laña, I., Lobo, J.L., Capecci, E., Del Ser, J., Kasabov, N., 2019. Adaptive long-term traffic state estimation with evolving spiking neural networks. Transp. Res. C 101, 126–144.

Laszuk, D., 2023. PyEMD's documentation — PyEMD 0.4.0 documentation. URL: https://pyemd.readthedocs.io/en/latest/index.html, [Online; accessed 10. Jul. 2023].

Lobo, J.L., Laña, I., Del Ser, J., Bilbao, M.N., Kasabov, N., 2018. Evolving Spiking Neural Networks for online learning over drifting data streams. Neural Netw. 108, 1–19.

Lobo, J.L., Oregi, I., Bifet, A., Del Ser, J., 2020. Exploiting the stimuli encoding scheme of evolving Spiking Neural Networks for stream learning. Neural Netw. 123, 118–133.

Maciąg, P.S., Kasabov, N., Kryszkiewicz, M., Bembenik, R., 2019. Air pollution prediction with clustering-based ensemble of evolving spiking neural networks and a case study for London area. Environ. Model. Softw. 118, 262–280.

Maciąg, P.S., Kryszkiewicz, M., Bembenik, R., Lobo, J.L., Del Ser, J., 2021. Unsupervised anomaly detection in stream data with online evolving spiking neural networks. Neural Netw. 139, 118–139.

Maciąg, P.S., Sitek, W., Skonieczny, L., Rybiński, H., 2022. A comparative study of short text classification with spiking neural networks. URL: https://annals-csis.org/proceedings/2022/pliks/184.pdf.

Maciąg, P.S., Kryszkiewicz, M., Bembenik, R., 2020. Online evolving spiking neural networks for incremental air pollution prediction. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8.

Mao, X., Shen, T., Feng, X., 2017. Prediction of hourly ground-level PM2.5 concentrations 3 days in advance using neural networks with satellite data in eastern China. Atmospheric Pollut. Res. 8 (6), 1005–1015.

Masood, A., Ahmad, K., 2021. A review on emerging artificial intelligence (AI) techniques for air pollution forecasting: Fundamentals, application and performance. J. Clean. Prod. 322, 129072. http://dx.doi.org/10.1016/j.jclepro.2021.129072, URL: https://www.sciencedirect.com/science/article/pii/S0959652621032613.

Polikar, R., 2012. Ensemble learning. In: Zhang, C., Ma, Y. (Eds.), Ensemble Machine Learning: Methods and Applications. Springer US, Boston, MA, pp. 1–34.

Ponulak, F., Kasinski, A., 2011. Introduction to spiking neural networks: Information processing, learning and applications. Acta Neurobiol. Exp. 71 (4), 409–433.

Shaharuddin, M., Zaharim, A., Nor, M.J.M., Karim, O.A., Sopian, K., 2008. Application of wavelet transform on airborne suspended particulate matter and meteorological temporal variations. WSEAS Trans. Environ. Dev. 4 (2), 89–98, cited By 10.

Siwek, K., Osowski, S., 2012. Improving the accuracy of prediction of PM10 pollution by the wavelet transformation and an ensemble of neural predictors. Eng. Appl. Artif. Intell. 25 (6), 1246–1258.

Siwek, K., Osowski, S., 2016. Data mining methods for prediction of air pollution. Int. J. Appl. Math. Comput. Sci. 26 (2), 467–478.

Siwek, K., Osowski, S., Garanty, K., Sowinski, M., 2009. Ensemble of predictors for forecasting the PM10 pollution. In: VXV International Symposium on Theoretical Engineering. pp. 1–5.

Suthaharan, S., 2016. Support vector machine. In: Machine Learning Models and Algorithms for Big Data Classification. Springer, pp. 207–235.

Torres, M.E., Colominas, M.A., Schlotthauer, G., Flandrin, P., 2011. A complete ensemble empirical mode decomposition with adaptive noise. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4144–4147. http://dx.doi.org/10.1109/ICASSP.2011.5947265.

Tu, E., Kasabov, N., Yang, J., 2017. Mapping temporal variables into the NeuCube for improved pattern recognition, predictive modeling, and understanding of stream data. IEEE Trans. Neural Netw. Learn. Syst. 28 (6), 1305–1317.

Van Roode, S., Ruiz-Aguilar, J., González-Enrique, J., Turias, I., 2019. An artificial neural network ensemble approach to generate air pollution maps. Environ. Monit. Assess. 191, 1–15.

Wu, Z., Huang, N.E., 2009. Ensemble empirical mode decomposition: a noise-assisted data analysis method. Adv. Adapt. Data Anal. 1 (01), 1–41.

Yang, L., Yu, H., Barzegar, R., Adamowski, J., Wen, X., 2023. Ensemble learning of decomposition-based machine learning and deep learning models for multi-time step ahead streamflow forecasting in an arid region. http://dx.doi.org/10.21203/rs.3.rs-2770415/v1.

Zaharim, A., Shaharuddin, M., Nor, M.J.M., Karim, O.A., Sopian, K., 2009. Relationships between airborne particulate matter and meteorological variables using non-decimated wavelet transform. Eur. J. Sci. Res. 27 (2), 308–312.

Zhang, J.J., Wei, Y., Fang, Z., 2019. Ozone pollution: A major health hazard worldwide. Front. Immunol. 10, http://dx.doi.org/10.3389/fimmu.2019.02518, URL: https://www.frontiersin.org/articles/10.3389/fimmu.2019.02518.

Zhou, Y., Chang, F.-J., Chen, H., Li, H., 2020. Exploring Copula-based Bayesian Model Averaging with multiple ANNs for PM2.5 ensemble forecasts. J. Clean. Prod. 263, 121528. http://dx.doi.org/10.1016/j.jclepro.2020.121528, URL: https://www.sciencedirect.com/science/article/pii/S0959652620315754.