Online Evolving Spiking Neural Networks for Incremental Air Pollution Prediction

Piotr S. Maciąg, Marzena Kryszkiewicz, Robert Bembenik
Institute of Computer Science
Warsaw University of Technology
Nowowiejska 15/19, 00-665, Warsaw, Poland
piotr.maciag@pw.edu.pl, mkr@ii.pw.edu.pl, r.bembenik@ii.pw.edu.pl

***Abstract*—In this article, we offer a novel model, Online evolving Spiking Neural Network for Incremental Prediction (OeSNN-IP), of forecasting from data streams, which we employ to air pollution prediction. OeSNN-IP makes predictions of pollution values and learns from a given number of recent values in streams of pollution and weather data rather than only from the most recent value from each data stream. In the proposed prediction method, older stream values have less influence on predicted values than newer ones. Moreover, we contribute to the theory of evolving spiking neural networks by offering a new fast and effective technique for encoding input values into order values of input neurons and weights of synapses linking input and output neurons. Also, we formulated the tight upper bound on the Euclidean distance between vectors of synapses weights of an output neuron and of a candidate output neuron, which simplifies the selection of a similarity threshold used in the learning phase of OeSNN-IP when a candidate output neuron is compared with output neurons. The experiments conducted on Warsaw-Ursynow pollution data show highly competitive results of the OeSNN-IP prediction as compared to the results obtained using state-of-the-art methods and algorithms.**

***Index Terms*—evolving spiking neural networks, incremental air pollution prediction, online learning, data stream, encoding technique**

I. Introduction

We are increasingly aware of the factors that contribute to the climate change and how they can influence our everyday lifes. One of the ways allowing to minimize the negative effects of the pollution on our health is the ability to effectively and accurately predict pollution levels so as to employ appropriate security measures in a timely manner. In the case of air pollution, excessive amounts of pollutants can significantly increase the risk of severe diseases such as: stroke, asthma, lung cancer and cardiopulmonary diseases [1]–[5]. Effective air pollution prediction systems can help in planning and thus in extenuating the negative effects of the pollution. While there exist many models of air pollution forecasting, which can effectively predict future pollution values (e.g. [6]–[8]), there is still shortage of such models or methods of air pollution prediction which can simultaneously learn and make predictions in an online mode.

(Corresponding authors: Piotr S. Maciąg, Marzena Kryszkiewicz, Robert Bembenik)

An evolving Spiking Neural Network (eSNN) is a type of an Artificial Neural Network designed for classification problems, typically working with a variety of data encoding techniques (such as temporal encoding algorithms [9] or Gaussian Receptive Fields encoding (GRFs) [10], [11]). The distinctive feature of eSNN is the repository of output neurons, which, in the network training phase, is updated with candidate output neurons created for each learning example of data.

The original architecture and learning algorithms of eSNN were designed to be first taught with the training data and next deployed as classifiers in real-world scenarios [12]. In [13], an extension called Online evolving Spiking Neural Network (OeSNN) for classification in streaming data was introduced. OeSNN architecture is characterized by limited size of the repository of output neurons, which enables efficient classification of data stream values.

In this article, we introduce a novel model, Online evolving Spiking Neural Network for Incremental Prediction (OeSNN-IP), which can effectively and efficiently predict future air pollution values and learn from streams of pollution and weather data. Our model differs from OeSNN in that it is capable of forecasting real values, rather than decision classes.

Our contribution provided in this article is as follows:

- We introduce the OeSNN-IP incremental model and employ it for air pollution prediction. OeSNN-IP makes predictions of pollution values and learns from a given number of recent values in streams of pollution and weather data rather than only from the most recent value from each data stream.
- We contribute to the theory of evolving spiking neural networks by offering a new fast and effective technique for encoding input values into order values of input neurons and weights of synapses linking input and output neurons.
- We propose a method for determining values of OeSNN-IP network in such a way so that older stream values have less influence on predicted values than newer ones.
- We simplify the determination of a similarity threshold between vectors of synapses weights of candidate output neurons and vectors of synaptic weights of output neurons, which are already present in the output repository,

by providing the tight upper bound on the Euclidean distances between these vectors.

- In the experiments, we show that the proposed prediction approach provides results highly competitive to the results reported in the literature for state-of-the-art methods and algorithms. Moreover, as we demonstrate in the experimental evaluation, the OeSNN-IP model using the the proposed encoding technique is faster and more effective than its variant using the GRFs encoding.

II. The Proposed Online Evolving Spiking Neural Network OeSNN-IP for Incremental Prediction of Air Pollution

A. Problem Formulation

Let F denote a pollution feature (such as ozone or PM_{10} pollution), whose values are to be forecast. Additionally, let $\mathcal{C} = \{C_1, C_2, \dots, C_M\}$ denote the set of conditional features whose values are known in advance and do not need to be forecast (such as modeled temperature or wind parameters at consecutive hours or on consecutive days). Let $\mathbf{x}^{(F)}$ and $\mathbf{x}^{(C)}$ denote data streams of feature F and feature $C \in \mathcal{C}$, respectively. Moreover, let $x_t^{(F)}$ and $x_t^{(C)}$ denote a value of feature F and a value of feature C, respectively, at time point t. By $\mathcal{W}^{(F)}$ and $\mathcal{W}^{(C)}$, we denote windows of values in streams $\mathbf{x}^{(F)}$ and $\mathbf{x}^{(C)}$, respectively. We assume that the number of values in a window is the same for all features and is equal to the value of the $\mathcal{W}_{size}$ parameter. At time point t, $\mathcal{W}^{(F)}$ will contain the sequence of values: $x_{t-\mathcal{W}_{size}}^{(F)}, \dots, x_{t-1}^{(F)}$, while window $\mathcal{W}^{(C)}$ will contain the sequence of values: $x_{t-\mathcal{W}_{size}}^{(C)}$, $\dots$, $x_{t-1}^{(C)}$.

The problem considered in this paper is formulated as follows: given windows $\mathcal{W}^{(F)}, \mathcal{W}^{(C_1)}, \dots, \mathcal{W}^{(C_M)}$ of values of data streams at time point t, incrementally forecast pollution values based on online spiking neural network for time horizons $t+1, \dots, t+\Delta t$, where Δt is a user-given value.

B. The Proposed Architecture of OeSNN-IP

The proposed architecture of OeSNN-IP and its example use is presented in Fig. 1. OeSNN-IP consists of an input layer and an output layer. The input layer consists of as many separate groups of input neurons as the number of features used for predicting. A group of input neurons for feature F is denoted by $\mathbf{NI}^{(F)}$, while a group of input neurons for any feature $C \in \mathcal{C}$ is denoted by $\mathbf{NI}^{(C)}$, respectively. The number of input neurons in each group is the same and is determined by a user-given parameter NI_{size}. The output layer constitutes repository $\mathbf{NO}$ of output neurons. The maximal number of output neurons in $\mathbf{NO}$ is determined by a user-given parameter NO_{size}. The output repository is updated at each time point with one candidate output neuron.

Values of data streams at time points $1, \dots, T_{init}$, where T_{init} is a user given value, are used to initialize OeSNN-IP. Proper forecast with OeSNN-IP starts from time point $t = T_{init}$. At each time point t, the prediction of F value is made for all prediction time points $t+1, \dots, t+\Delta t$. By $y_{t,t_{pred}}^{(F)}$ we

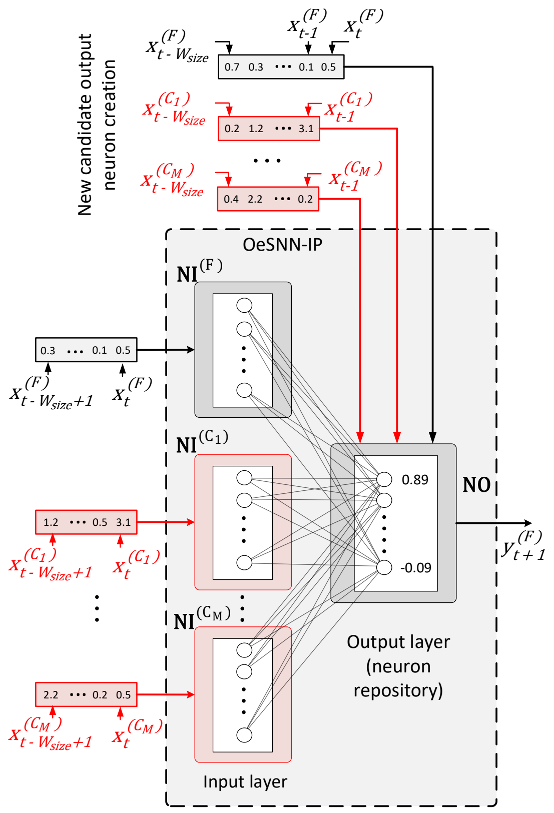


Fig. 1. The proposed OeSNN-IP architecture and an example of pollution value prediction for prediction time point $t+1$. OeSNN-IP consists of two layers. The first layer contains groups $\mathbf{NI}^{(F)}$, $\mathbf{NI}^{(C_1)}$, ..., $\mathbf{NI}^{(C_M)}$ of input neurons, where each group contains NI_{size} input neurons. The second layer constitutes repository $\mathbf{NO}$ of at most NO_{size} output neurons.

denote the OeSNN-IP prediction of feature F value for time point $t + t_{pred}$, where $t_{pred} = 1, 2, \dots, \Delta t$. Additionally, $\mathbf{y}_t^{(F)}$ will denote a vector of pollution predictions for time points $t+1, \dots, t+\Delta t$ and $\mathbf{Y}^{(F)}$ will be a matrix whose first row contains $\mathbf{y}_{T_{init}}^{(F)}$, second row contains $\mathbf{y}_{T_{init}+1}^{(F)}$ etc.

C. The Proposed Input Layer Encoding of OeSNN-IP

The aim of the input layer is to encode values present in windows for the purposes of learning and forecasting with OeSNN-IP. Let $\mathcal{A} = \{F\} \cup \mathcal{C}$. In this subsection, we will offer a new technique of encoding values in window $\mathcal{W}^{(A)}$, where A is any feature in $\mathcal{A}$. Let $\mathcal{W}^{(A)} = [\omega_1, \dots, \omega_{\mathcal{W}_{size}}]$, where ω_1 is the oldest value in window $\mathcal{W}^{(A)}$, while $\omega_{\mathcal{W}_{size}}$ is the newest value in this window. Additionally, u-th value, where $u \in [1, \dots, \mathcal{W}_{size}]$ of window $\mathcal{W}^{(A)}$ will be also referred to as $\mathcal{W}^{(A)}[u]$. Each value ω_u in $\mathcal{W}^{(A)}$ will be encoded into, so called, *order values* of NI_{size} input neurons of group $\mathbf{NI}^{(A)}$. The order values obtained for time point t will be then used for calculating weights of synapses between input neurons and the candidate output neuron created for this time point and will influence predictions $\mathbf{y}_t^{(F)}$ of the OeSNN-IP.

Now, we will focus on presenting our value encoding method. Let $I_{min}^{(A)}$ and $I_{max}^{(A)}$ be the minimal value and the maximal value, respectively, in stream $\mathbf{x}^{(A)}$ up to the current time point t. $I^{(A)}_{min}$ and $I^{(A)}_{max}$ are first obtained from $x^{(A)}_1, \dots, x^{(A)}_{T_{init}}$ initial values of data stream $\mathbf{x}^{(A)}$, in which T_{init} is a user-given initialization parameter, and then are updated with each new incoming data stream value. Given values $I^{(A)}_{min}$ and $I^{(A)}_{max}$, we define a *center value* (denoted by $\mu^{(A)}_{n_j}$) for an input neuron $n_j \in \mathbf{NI}^{(A)}$ as shown in Eq. (1):

$$\mu^{(A)}_{n_j} = I^{(A)}_{min} + (j - 0.5) \cdot width^{(A)}, \tag{1}$$

where $width^{(A)} = \frac{I^{(A)}_{max} - I^{(A)}_{min}}{NI_{size}}$.

Proposition 1. *Let $\omega_u \in \mathcal{W}^{(A)}$ and μ_{n_j} be the center value of $n_j \in \mathbf{NI}^{(A)}$. Value ω_u is closest to center value μ_{n_j} $\iff$*

$$j = \begin{cases} \left\lfloor \dfrac{\omega_u - I^{(A)}_{min}}{width^{(A)}} \right\rfloor + 1, & \textit{if } \omega_u < I^{(A)}_{max}, \\ \left\lfloor \dfrac{\omega_u - I^{(A)}_{min}}{width^{(A)}} \right\rfloor = NI_{size}, & \textit{if } \omega_u = I^{(A)}_{max}. \end{cases} \tag{2}$$

Let $\omega_u \in \mathcal{W}^{(A)}$ and center value $\mu^{(A)}_{n_j}$ of input neuron $n_j \in \mathbf{NI}^{(A)}$ be closest to feature value ω_u among center values of all input neurons in $\mathbf{NI}^{(A)}$. Additionally, let $l = min(j - 1, NI_{size} - j)$. We define function *rank* for pairs (an input neuron in $\mathbf{NI}^{(A)}$, ω_u) as follows:

- $rank(n_j, \omega_u) = 0$
- If $\omega_u < \mu_{n_j}$, then:
  - $rank(n_{j-k}, \omega_u) = 2 \cdot k - 1$ for $k = 1, \dots, l$,
  - $rank(n_{j-l-k}, \omega_u) = 2 \cdot l - 1 + k$ for $k = 1, \dots, j - 1 - l$,
  - $rank(n_{j+k}, \omega_u) = 2 \cdot k$ for $k = 1, \dots, l$,
  - $rank(n_{j+l+k}, \omega_u) = 2 \cdot l + k$ for $k = 1, \dots, NI_{size} - j - l$
- If $\omega_u \geq \mu_{n_j}$, then:
  - $rank(n_{j-k}, \omega_u) = 2 \cdot k$ for $k = 1, \dots, l$,
  - $rank(n_{j-l-k}, \omega_u) = 2 \cdot l + k$ for $k = 1, \dots, j - 1 - l$,
  - $rank(n_{j+k}, \omega_u) = 2 \cdot k - 1$ for $k = 1, \dots, l$,
  - $rank(n_{j+l+k}, \omega_u) = 2 \cdot 2 \cdot l - 1 + k$ for $k = 1, \dots, NI_{size} - j - l$.

Please note that the rank function does not involve calculation of any distances among window value ω_u and the center values of input neurons in $\mathbf{NI}^{(A)}$, but rank values it generates correspond to these distances, as we claim in Proposition 2.

Proposition 2. *Let $\omega_u \in \mathcal{W}^{(A)}$ and $n_j, n_k \in \mathbf{NI}^{(A)}$.*

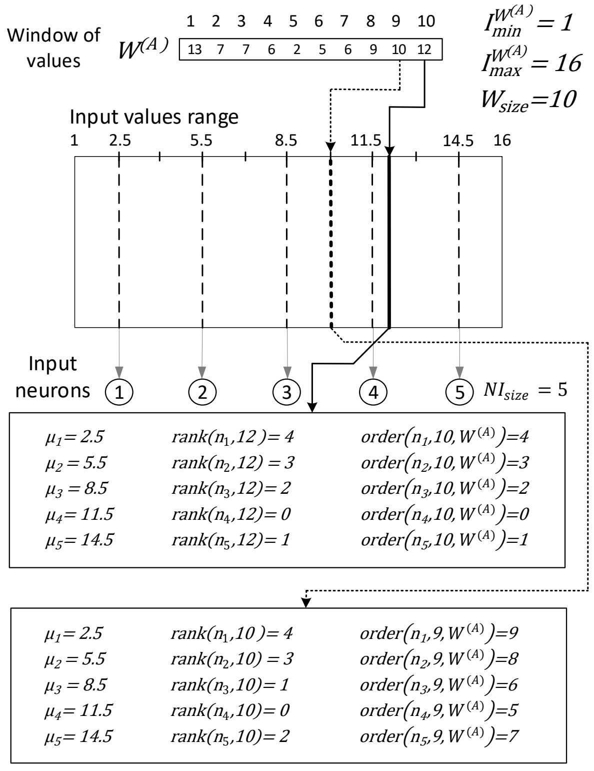
- *$rank(n_j, \omega_u) \in [0, 1, \dots, NI_{size} - 1]$.*
- *$rank(n_j, \omega_u) \neq rank(n_k, \omega_u)$ for $i \neq j$.*
- *$|\mu_{n_j} - \omega_u| < |\mu_{n_k} - \omega_u| \implies rank(n_j, \omega_u) < rank(n_k, \omega_u)$.*
- *$rank(n_j, \omega_u) < rank(n_k, \omega_u) \implies |\mu_{n_j} - \omega_u| \leq |\mu_{n_k} - \omega_u|$.*

Now, we define function *order* for each input neuron n_j in $\mathbf{NI}^{(A)}$ and u-th value in window $\mathcal{W}^{(A)}$, as follows:

$$order(n_j, u, \mathcal{W}^{(A)}) = rank(n_j, \mathcal{W}^{(A)}[u]) + (\mathcal{W}_{size} - u) \cdot NI_{size} \tag{3}$$

Corollary 1. *Let $n_j, n_k \in \mathbf{NI}^{(A)}$ and $u, u' \in [1, \dots, \mathcal{W}_{size}]$.*

- *$order(n_j, u, \mathcal{W}^{(A)}) \in [0, 1, \dots, u \cdot NI_{size} - 1]$.*
- *$order(n_j, u, \mathcal{W}^{(A)}) \neq order(n_k, u', \mathcal{W}^{(A)})$ for $i \neq j$'or $u \neq u'$.*
- *$u > u' \implies order(n_j, u, \mathcal{W}^{(A)}) < order(n_k, u', \mathcal{W}^{(A)})$.*
- *$order(n_j, u, \mathcal{W}^{(A)}) < order(n_k, u', \mathcal{W}^{(A)}) \implies u \geq u'$.*



$\mu_1 = 2.5$	$rank(n_1, 12) = 4$	$order(n_1, 10, W^{(A)}) = 4$
$\mu_2 = 5.5$	$rank(n_2, 12) = 3$	$order(n_2, 10, W^{(A)}) = 3$
$\mu_3 = 8.5$	$rank(n_3, 12) = 2$	$order(n_3, 10, W^{(A)}) = 2$
$\mu_4 = 11.5$	$rank(n_4, 12) = 0$	$order(n_4, 10, W^{(A)}) = 0$
$\mu_5 = 14.5$	$rank(n_5, 12) = 1$	$order(n_5, 10, W^{(A)}) = 1$

$\mu_1 = 2.5$	$rank(n_1, 10) = 4$	$order(n_1, 9, W^{(A)}) = 9$
$\mu_2 = 5.5$	$rank(n_2, 10) = 3$	$order(n_2, 9, W^{(A)}) = 8$
$\mu_3 = 8.5$	$rank(n_3, 10) = 1$	$order(n_3, 9, W^{(A)}) = 6$
$\mu_4 = 11.5$	$rank(n_4, 10) = 0$	$order(n_4, 9, W^{(A)}) = 5$
$\mu_5 = 14.5$	$rank(n_5, 10) = 2$	$order(n_5, 9, W^{(A)}) = 7$

Fig. 2. The proposed method of encoding values in window $\mathcal{W}^{(A)}$.

An example of encoding is presented in Fig. 2.

According to the proposed encoding technique, the order values of input neurons for each subsequent value in window $\mathcal{W}^{(A)}$, starting from the newest value to the oldest in the window, is increasing. This property in combination with the model of output neurons of OeSNN-IP will allow us to control the influence of values in $\mathcal{W}^{(A)}$ on values of synapses weights and directly on OeSNN-IP prediction results. The most recent values in windows have the highest impact on the prediction of the network, while the impact of the oldest values in windows is less significant.

D. The Proposed Output Layer of OeSNN-IP

In our proposed learning model of OeSNN-IP, for each new input pollution value $x^{(F)}_t$ of data stream $\mathbf{x}^{(F)}$, a new candidate output neuron n_c is created, initialized and used to update repository $\mathbf{NO}$ of output neurons. The synapses are created between candidate n_c and each input neuron in the input layer. The weights of synapses are initialized based on encoding of values present in windows $\mathcal{W}^{(F)}, \mathcal{W}^{(C_1)}, \dots, \mathcal{W}^{(C_M)}$. Let A be any feature in $\mathcal{A} = \{F\} \cup \mathcal{C}$ and $\mathcal{W}^{(A)}$ be a window of $\mathcal{W}_{size}$ values of feature A. For each time point t, where $t > \mathcal{W}_{size}$, $\mathcal{W}^{(A)}$ consists of values $x^{(A)}_{t-\mathcal{W}_{size}}, \dots, x^{(A)}_{t-1}$, and a new candidate output neuron is created. The vector of weights of synapses connecting candidate output neuron n_c with input neurons in groups $\mathbf{NI}^{(F)}, \mathbf{NI}^{(C_1)}, \ldots, \mathbf{NI}^{(C_M)}$ will be denoted by $\mathbf{w}_{n_c}$ and defined as $[w^{(F)}_{n_{j_1}n_c}, \ldots, w^{(F)}_{n_{j_{NI_{size}}}n_c}$, $w^{(C_1)}_{n_{j_1}n_c}, \ldots, w^{(C_1)}_{n_{j_{NI_{size}}}n_c}, \ldots, w^{(C_M)}_{n_{j_1}n_c}, \ldots, w^{(C_M)}_{n_{j_{NI_{size}}}n_c}]$, where $w_{n_j n_c}$ denotes the weight of the synapse connecting n_c with input neuron n_j in group $\mathbf{NI}^{(A)}$. Weight $w^{(A)}_{n_j n_c}$ is calculated according to Eq. (4):

$$w^{(A)}_{n_j n_c} = \sum_{u=1}^{\mathcal{W}_{size}} mod^{order(n_j, u, \mathcal{W}^{(A)})}. \tag{4}$$

We assume that each candidate output neuron n_c (as well as each output neuron n_i in $\mathbf{NO}$) is characterized by the following attributes: its output value v_{n_c} (v_{n_i}), update time point τ_{n_c} (τ_{n_i}) and update counter M_{n_c} (M_{n_c}). In the case of candidate output neuron n_c, $v_{n_c} = x^{(F)}_t$, $\tau_{n_c} = t$ and $M_{n_c} = 1$.

After the candidate output neuron is created and initialized, it is used to update repository $\mathbf{NO}$ of output neurons. Candidate n_c is either merged with one of the output neurons already present in $\mathbf{NO}$ based on a similarity condition of synapses weights, or replaces one of the output neurons in $\mathbf{NO}$ if the number of neurons in $\mathbf{NO}$ reached limit NO_{size}, or is simply added to $\mathbf{NO}$ otherwise. If n_c is merged with output neuron n_s in $\mathbf{NO}$, then its synaptic weights vector $\mathbf{w}_{n_s} = [w^{(F)}_{n_{j_1}n_s}, \ldots, w^{(F)}_{n_{j_{NI_{size}}}n_s}, w^{(C_1)}_{n_{j_1}n_s}, \ldots, w^{(C_1)}_{n_{j_{NI_{size}}}n_s}$, $\ldots, w^{(C_M)}_{n_{j_1}n_s}, \ldots, w^{(C_M)}_{n_{j_{NI_{size}}}n_s}]$, output value v_{n_s}, update time point τ_{n_s} and update counter M_{n_s} of n_s are updated according to formulae given in Eq. (5):

$$\begin{aligned} &\mathbf{w}_{n_s} \leftarrow (\mathbf{w}_{n_s} \cdot M_{n_s} + \mathbf{w}_{n_c})/(M_{n_s} + 1), \\ &v_{n_s} \leftarrow (v_{n_s} \cdot M_{n_s} + v_{n_c})/(M_{n_s} + 1), \\ &\tau_{n_s} \leftarrow (\tau_{n_s} \cdot M_{n_s} + \tau_{n_c})/(M_{n_s} + 1), \\ &M_{n_s} \leftarrow M_{n_s} + 1. \end{aligned} \tag{5}$$

In fact, candidate output neuron n_c is merged with output neuron n_s such that the Euclidean distance, denoted by $Dist_{n_c,n_s}$, between synapses weights vector $\mathbf{w}_{n_c}$ and synapses weights vector $\mathbf{w}_{n_s}$ is minimal and sufficiently small. The question is how to define sufficiently small distance? We propose to compare $Dist_{n_c,n_s}$ with $simTr \cdot \overline{Dist}$ value, where $simTr$ is a user-given parameter and $\overline{Dist}$ is the tight upper bound on the Euclidean distances between any possible candidate output neuron and any output neuron in $\mathbf{NO}$. In Proposition 3, we show how to calculate the value of the $\overline{Dist}$ bound.

Proposition 3.

$$\overline{Dist} = \left[\sum_{A \in \mathcal{A}} \sum_{j=1}^{NI_{size}} \left(\sum_{u=0}^{W_{size}-1} (mod^{j+NI_{size}\cdot u} - mod^{NI_{size}\cdot(u+1)-j-1}) \right)^2 \right]^{\frac{1}{2}}, \mathcal{A} = \{F\} \cup \mathcal{C} \tag{6}$$

The prediction of pollution values for time points $t + 1, \ldots, t + \Delta t$ is made by OeSNN-IP based on encoding of values in prediction windows $\mathcal{WP}^{(F)}, \mathcal{WP}^{(C_1)}, \ldots, \mathcal{WP}^{(C_M)}$. For predicting value $y^{(F)}_{t,1}$ of feature F for time point $t + 1$, prediction window $\mathcal{WP}^{(F)}$ equal to $[x^{(F)}_{t-\mathcal{W}_{size}+1}, \ldots, x^{(F)}_t]$ as well as each prediction window $\mathcal{WP}^{(C)}$, where $C \in \mathcal{C}$, equal to $[x^{(C)}_{t-\mathcal{W}_{size}+1}, \ldots, x^{(C)}_t]$ are used.

For predicting a value of feature F for next time points, prediction window $\mathcal{WP}^{(F)}$ will contain both some known values of feature F as well as some predicted values of F, while prediction windows of conditional features will contain only known values. More precisely, for predicting value $y^{(F)}_{t,t_{pred}}$ of feature F for time point $t + t_{pred}$, where $t_{pred} = 2, \ldots, \Delta t$, prediction window $\mathcal{WP}^{(F)}$ equal to $[x^{(F)}_{t-\mathcal{W}_{size}+t_{pred}}, \ldots, x^{(F)}_t, y^{(F)}_{t,1}, \ldots, y^{(F)}_{t,t_{pred-1}}]$, as well as each prediction window $\mathcal{WP}^{(C)}$, where $c \in \mathcal{C}$, equal to $[x^{(C)}_{t-\mathcal{W}_{size}+t_{pred}}, \ldots, x^{(C)}_{t,t_{pred-1}}]$ are applied.

In our OeSNN-IP model, the predicted value of feature F for time point $t+t_{pred}$, where $t_{pred} = 1, \ldots, \Delta t$, is determined as the average of the output values of the output neurons whose *postsynaptic potentials* calculated for $t + t_{pred}$ are maximal. *Postsynaptic potential* PSP_{n_i} of output neuron n_i in $\mathbf{NO}$ for time point $t + t_{pred}$ is calculated according to Eq. (7):

$$PSP_{n_i} = \sum_{A \in \mathcal{A}} \sum_{n_j \in \mathbf{NI}^{(A)}} \sum_{u=1}^{\mathcal{W}_{size}} w^{(A)}_{n_j n_i} \cdot mod^{order(n_j, u, \mathcal{WP}^{(A)})}, \tag{7}$$

where $w^{(A)}_{n_j n_i}$ is the weight of the synapse between input neuron n_j and output neuron n_i at time point t and $\mathcal{WP}^{(A)}$ contains values of feature A used for predicting $y^{(F)}_{t,t_{pred}}$.

E. The Proposed OeSNN-IP Prediction Algorithm

In this subsection, we present our algorithm for incremental air pollution prediction from data streams. The main procedure of OeSNN-IP is presented in Algorithm 1. It uses the following parameters: $\mathcal{W}_{size}$, T_{init}, NI_{size}, NO_{size}, mod, $simTr$. It is assumed that $T_{init} > \mathcal{W}_{size}$ and that first T_{init} values of each data stream are known. Algorithm 1 starts with calculation of the upper bound $\overline{Dist}$ and initialization of windows of input values for feature F and all features $C \in \mathcal{C}$. Each window is initialized with first $\mathcal{W}_{size}$ values of the respective data stream. Then, the minimal I_{min} and maximal I_{max} values of first T_{init} values of each data stream are obtained. Next, in steps 6 to 14, the initial OeSNN-IP learning is carried out for each input value at time point t, where $t \in W_{size} + 1, \ldots, T_{init} - 1$, in the following way: (i) the encoding based on the content of each window $\mathcal{W}^{(A)} = [\mathbf{x}_{t-W^{(A)}_{size}}, \ldots, \mathbf{x}^{(A)}_{t-1}]$, where $A \in \{F\} \cup \mathcal{C}$, is carried out (see the InputLayerEncoding procedure presented in Algorithm 2), (ii) new candidate output neuron n_c is created, initialized (see the InitCandidateOutputNeuron procedure presented in Algorithm 3), and used to update repository $\mathbf{NO}$ (see the UpdateRepository procedure presented in

Algorithm 4), and (iii) the content of each window $\mathcal{W}^{(A)}$, where $A \in \{F\} \cup \mathcal{C}$, is updated with value $x_t^{(A)}$.

After the initialization of OeSNN-IP, the network will work alternately in the following two stages for each time point $t \geq T_{init}$:

1) The OeSNN-IP learning will be carried out in the same way as during the initialization described above.
2) The prediction phase starts from initializing the content of each prediction window $\mathcal{WP}^{(A)}$, where $A \in \{F\} \cup \mathcal{C}$, with $\mathcal{W}^{(A)}$. Next, the prediction with OeSNN-IP is performed for each time point $t + t_{pred}$, where $t_{pred} = 1, \ldots, \Delta t$, in the following way: (i) the encoding based on the current content of each window $\mathcal{WP}^{(A)}$, where $A \in \{F\} \cup \mathcal{C}$ is carried out (see the InputLayerEncoding procedure presented in Algorithm 2), (ii) OeSNN-IP predicts value $y_{t,t_{pred}}^{(F)}$ (see procedure PredictValue presented in Algorithm 5) , and (iii) the content of window $\mathcal{WP}^{(F)}$ is updated with value $y_{t,t_{pred}}^{(F)}$, while the content of each window $\mathcal{WP}^{(C)}$, where $C \in \mathcal{C}$, is updated with value $x_{t+t_{pred}}^{(C)}$ of data stream $\mathbf{x}^{(C)}$. Finally, the vector $\mathbf{y}_t^{(F)} = [y_{t,1}^{(F)}, \ldots, y_{t,\Delta t}^{(F)}]$ of predicted values of feature F is appended to prediction results $\mathbf{Y}^{(F)}$ obtained so far.

Now, we will describe the procedures used by OeSNN-IP (see Algorithm 1): InputLayerEncoding, InitCandidateOutputNeuron, UpdateRepository and PredictValue.

The InputLayerEncoding procedure (see Algorithm 2) performs encoding of values present in the set of windows $\mathcal{W}^{(F)}$, $\mathcal{W}^{(C_1)}, \ldots, \mathcal{W}^{(C_M)}$ (each of which is either a window containing some $\mathcal{W}_{size}$ consecutive values from a respective data stream or is a prediction window) that were provided as arguments of this procedure. First, minimal $I_{min}^{(A)}$ and maximal $I_{max}^{(A)}$ values, where $A \in \{F\} \cup \mathcal{C}$, are updated based on the newest value in window $\mathcal{W}^{(A)}$. The updated values of $I_{min}^{(A)}$ and $I_{max}^{(A)}$ are used to calculate current width $width^{(A)}$ and central value $\mu_{n_j}^{(A)}$ for each attribute A in $\{F\} \cup \mathcal{C}$ and for each input neuron n_j in $\mathbf{NI}$. Based on this information and content of the passed windows, the order value is calculated for each input neuron according to Eq. (3).

The InitCandidateOutputNeuron procedure (see Algorithm 3) is responsible for initialization of each new candidate output neuron n_c (which is used to update repository $\mathbf{NO}$ of output neurons). First, InitCandidateOutputNeuron creates synapses between n_c and each input neuron in $\mathbf{NI}$, and then calculates their weights based on order values of input neurons according to Eq. (4). Next, output value v_{n_c} of the candidate is assigned value $x_t^{(F)}$, its update time point τ_{n_c} is set to t value, and M_{n_c} is set to 1.

The UpdateRepository procedure (see Algorithm 4), starts with calculation of the distance between vector $\mathbf{w}_{n_c}$ of synapses weights of candidate output neuron n_c and vector $\mathbf{w}_{n_i}$ of synapses weights of each output neuron n_i in $\mathbf{NO}$. Then, it determines this output neuron n_s in $\mathbf{NO}$ for which the distance is the smallest. If output repository $\mathbf{NO}$ is not empty and $Dist_{n_c,n_s}$ does not exceed $simTr \cdot \overline{Dist}$, then the values of attributes of output neuron n_s are updated with the values of attributes of candidate output neuron n_c according to formulae given in Eq. (5). Otherwise, if the current number of output neurons in $\mathbf{NO}$ is less than NO_{size}, then n_c is added to $\mathbf{NO}$. Else, n_c replaces an output neuron $n_{oldest} \in \mathbf{NO}$ whose value of update time $\tau_{n_{oldest}}$ is the smallest in comparison with the update times of other output neurons in $\mathbf{NO}$.

Algorithm 1 OeSNN-IP

```
Input: $\mathbf{x}^{(F)}, \mathbf{x}^{(C_1)}, \ldots, \mathbf{x}^{(C_M)}$ - streams of input data.
    Global constants: $\mathcal{W}_{size}$, $NO_{size}$, $NI_{size}$, $mod$, $simTr$, $T_{init}$
    Global variables: $I_{min}^{(F)}, I_{max}^{(F)}, I_{min}^{(C_1)}, I_{max}^{(C_1)}, \ldots, I_{min}^{(C_M)}, I_{max}^{(C_M)}$
Output: $\mathbf{Y}^{(F)}$ - a matrix with predicted pollution values.
 1: Calculate $\overline{Dist}$ according to Eq. (6).
 2: for $A \in \{F\} \cup \mathcal{C}$ do
 3:     $\mathcal{W}^{(A)} \leftarrow [x_1^{(A)}, \ldots, x_{\mathcal{W}_{size}}^{(A)}]$
 4:     Obtain $I_{min}^{(A)}, I_{max}^{(A)}$ based on $\mathcal{W}^{(A)}$
 5: end for
 6: for $t \leftarrow \mathcal{W}_{size} + 1$ to $T_{init} - 1$ do

        {*————— OeSNN-IP learning —————*}
 7:     InputLayerEncoding($\mathcal{W}^{(F)}$, $\mathcal{W}^{(C_1)}$, ..., $\mathcal{W}^{(C_M)}$)
 8:     Create a candidate output neuron $n_c$
 9:     $n_c \leftarrow$ InitCandidateOutputNeuron($n_c$, $x_t^{(F)}$)
10:     UpdateRepository($n_c$)
11:     for all $A \in \{F\} \cup \mathcal{C}$ do
12:         Update $\mathcal{W}^{(A)}$ with $x_t^{(A)}$
13:     end for

14: end for
15: $t \leftarrow T_{init} - 1$
16: repeat
17:     $t \leftarrow t + 1$

        {*————— OeSNN-IP learning —————*}
18:     InputLayerEncoding($\mathcal{W}^{(F)}$, $\mathcal{W}^{(C_1)}, \ldots, \mathcal{W}^{(C_M)}$)
19:     Create a candidate output neuron $n_c$
20:     $n_c \leftarrow$ InitCandidateOutputNeuron($n_c$, $x_t^{(F)}$)
21:     UpdateRepository($n_c$)
22:     for all $A \in \{F\} \cup \mathcal{C}$ do
23:         Update $\mathcal{W}^{(A)}$ with $x_t^{(A)}$;
24:     end for

        {*————— OeSNN-IP predicting —————*}
25:     for all $A \in \{F\} \cup \mathcal{C}$ do $\mathcal{WP}^{(A)} \leftarrow \mathcal{W}^{(A)}$ end for
26:     for $t_{pred} \leftarrow 1$ to $\Delta t$ do
27:         InputLayerEncoding($\mathcal{WP}^{(F)}$, $\mathcal{WP}^{(C_1)}$, ...,
                $\mathcal{WP}^{(C_M)}$)
28:         $y_{t,t_{pred}}^{(F)} \leftarrow$ PredictValue( )
29:         Update $\mathcal{WP}^{(F)}$ with $y_{t,t_{pred}}^{(F)}$
30:         Append $y_{t,t_{pred}}^{(F)}$ to $\mathbf{y}_t^{(F)}$
31:         for all $C \in \mathcal{C}$ do
32:             Update $\mathcal{WP}^{(C)}$ with $x_{t+t_{pred}}^{(C)}$
33:         end for
34:     end for
35:     Append $\mathbf{y}_t^{(F)}$ to $\mathbf{Y}^{(F)}$;

36: until there is $x_{t+1}^{F}$ in $\mathbf{x}^{(F)}$
37: return $\mathbf{Y}^{(F)}$
```

Algorithm 2 InputLayerEncoding($\mathcal{W}^{(F)}$, $\mathcal{W}^{(C_1)}$, ..., $\mathcal{W}^{(C_M)}$)

```
Input: $\mathcal{W}^{(F)}, \mathcal{W}^{(C_1)}, \dots, \mathcal{W}^{(C_M)}$ - windows of values of each feature in $\{F\} \cup \mathcal{A}$
1: for all $A \in \{F\} \cup \mathcal{C}$ do
2:     if $I_{min}^{\mathcal{W}^{(A)}} > \mathcal{W}^{(A)}[W_{size}]$ then
3:         $I_{min}^{\mathcal{W}^{(A)}} \leftarrow \mathcal{W}^{(A)}[W_{size}]$
4:     end if
5:     if $I_{max}^{\mathcal{W}^{(A)}} < \mathcal{W}^{(A)}[W_{size}]$ then
6:         $I_{max}^{\mathcal{W}^{(A)}} \leftarrow \mathcal{W}^{(A)}[W_{size}]$
7:     end if
8:     if $I_{min}^{\mathcal{W}^{(A)}}$ or $I_{max}^{\mathcal{W}^{(A)}}$ changed then
9:         $width^{(A)} \leftarrow \frac{I_{max}^{(A)} - I_{min}^{(A)}}{NI_{size}}$
10:        for all $n_j \in \mathbf{NI}^{(A_k)}$ do
11:            $\mu_{n_j}^{(A)} \leftarrow I_{min}^{\mathcal{W}^{(A)}} + (j - 0.5) \cdot width^{(A)}$
12:        end for
13:    end if
14:    for $u \leftarrow 1$ to $W_{size}$ do
15:        for all $n_j \in \mathbf{NI}^{(A)}$ do
16:            Calculate $order(n_j, u, \mathcal{W}^{(A)})$ according to Eq. (3)
17:        end for
18:    end for
19: end for
```

Algorithm 3 InitCandidateOutputNeuron(n_c, $x_t^{(F)}$)

```
Input: $n_c$ - a candidate output neuron to be initialized; $x_t^{(F)}$ - value of data stream $\mathbf{x}^{(F)}$ at time point $t$
Output: $n_c$ - an initialized candidate output neuron
1: for all $A \in \{F\} \cup \mathcal{C}$ do
2:     for all $n_j \in \mathbf{NI}^{(A)}$ do
3:         Create synapse between $n_j \in \mathbf{NI}^{(A)}$ and $n_c$
4:         $w_{n_j n_c} \leftarrow 0$
5:         for $u \leftarrow 1$ to $W_{size}$ do
6:             $w_{n_j n_c}^{(A)} \leftarrow w_{n_j n_c}^{(A)} + mod^{order(n_j, u, \mathcal{W}^{(A)})}$
7:         end for
8:     end for
9: end for
10: $v_{n_c} \leftarrow x_t^{(F)}$; $\tau_{n_c} \leftarrow t$; $M_{n_c} \leftarrow 1$
11: return $n_c$
```

Algorithm 4 UpdateRepository(n_c)

```
Input: $n_c$ - a newly created candidate output neuron
1: if $|\mathbf{NO}| > 0$ then
2:     for all $n_i \in \mathbf{NO}$ do
3:         $Dist_{n_c,n_i} \leftarrow distance(\mathbf{w}_{n_c}, \mathbf{w}_{n_i})$
4:     end for
5:     $n_s \leftarrow$ an output neuron in $\mathbf{NO}$ such that
           $Dist_{n_c,n_s} = min\{Dist_{n_c,n_i} \mid n_i \in \mathbf{NO}\}$
6: end if
7: if $|\mathbf{NO}| > 0 \land Dist_{n_c,n_s} \leq simTr \cdot \overline{Dist}$ then
8:     $\mathbf{w}_{n_s} \leftarrow (\mathbf{w}_{n_c} + M_{n_s} \cdot \mathbf{w}_{n_s})/(M_{n_s} + 1)$
9:     $v_{n_s} \leftarrow (v_{n_c} + M_{n_s} \cdot v_{n_s})/(M_{n_s} + 1)$
10:    $\tau_{n_s} \leftarrow (\tau_{n_c} + M_{n_s} \cdot \tau_{n_s})/(M_{n_s} + 1)$
11:    $M_{n_s} \leftarrow M_{n_s} + 1$
12: else if $|\mathbf{NO}| < NO_{size}$ then
13:    Insert $n_c$ to $\mathbf{NO}$;
14: else
15:    $n_{oldest} \leftarrow$ an output neuron in $\mathbf{NO}$ such that
           $\tau_{n_{oldest}} = min\{\tau_{n_i} | i = 1, \dots, NO_{size}\}$
16:    Replace $n_{oldest}$ with $n_c$ in $\mathbf{NO}$
17: end if
```

The PredictValue procedure (see Algorithm 5) starts with setting postsynaptic potentials of all output neurons in **NO** to 0. Next, the postsynaptic potentials of all output neurons in **NO** are calculated based on the current order values of the input neurons and weights of synapses between each input neuron in **NI** and each output neuron in **NO** according to Eq. (7). Finally, the returned predicted pollution value is determined as the average of the output values of those output neurons whose postsynaptic potentials are maximal.

Algorithm 5 PredictValue( )

```
Output: $v_{pred}$ - predicted pollution value
1: for all $n_i \in \mathbf{NO}$ do $PSP_{n_i} \leftarrow 0$ end for
2: for all $A \in \{F\} \cup \mathcal{C}$ do
3:     for all $n_j \in \mathbf{NI}^{(A)}$ do
4:         for all $n_i \in \mathbf{NO}$ do
5:             for $u \leftarrow 1$ to $W_{size}$ do
6:                 $PSP_{n_i} \leftarrow PSP_{n_i} +$
                       $w_{n_j n_i}^{(A)} \cdot mod^{order(n_j, u, \mathcal{W}^{(A)})}$
7:             end for
8:         end for
9:     end for
10: end for
11: $\mathbf{NO}_{max} \leftarrow \{n_i \mid n_i \in \mathbf{NO} \land PSP_{n_i}$ is maximal$\}$
12: $v_{pred} \leftarrow \frac{\sum_{n_i \in \mathbf{NO}_{max}} v_{n_i}}{|\mathbf{NO}_{max}|}$
13: return $v_{pred}$
```

III. Experimental Analysis

In this section, we provide the results of the experimental evaluation of our proposed OeSNN-IP model for air pollution prediction. Our implementation of OeSNN-IP is available in the GitHub repository[1]. The quality of the obtained results is measured using the following prediction quality measures: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) and Index of Agreement (IA), whose definitions are provided beneath:

$$\text{MAE}_{t_{pred}} = \frac{\sum_{t=T_{init}}^{T} |x_{t+t_{pred}} - y_{t,t_{pred}}|}{T - T_{init}}. \tag{8}$$

$$\text{RMSE}_{t_{pred}} = \sqrt{\frac{\sum_{t=T_{init}}^{T} (x_{t+t_{pred}} - y_{t,t_{pred}})^2}{T - T_{init}}}. \tag{9}$$

$$\text{MAPE}_{t_{pred}} = \frac{100\%}{T - T_{init}} \sum_{t=T_{init}}^{T} \left| \frac{x_{t+t_{pred}} - y_{t,t_{pred}}}{y_{t,t_{pred}}} \right|. \tag{10}$$

$$\text{IA}_{t_{pred}} = 1 - \Big[\sum_{t=T_{init}}^{T} (x_{t+t_{pred}} - y_{t,t_{pred}})^2 / \sum_{t=T_{init}}^{T} \Big(\sum_{t=T_{init}}^{T} |y_{t,t_{pred}} - \overline{x}| + \sum_{t=T_{init}}^{T} |x_{t+t_{pred}} - \overline{x}| \Big)^2 \Big]^{\frac{1}{2}}. \tag{11}$$

where $t_{pred} = 1, 2, \dots, \Delta t$ denotes the length of the prediction horizon, T is the total number of observations in each data

[1]https://github.com/piotrMaciag32/OeSNN_Air_Pollution

stream decreased by Δt, and, for simplicity, $x_{t+t_{pred}}$ denotes $x^{(F)}_{t+t_{pred}}$, while $y_{t,t_{pred}}$ denotes $y^{(F)}_{t,t_{pred}}$.

In all our experiments, we used the PM_{10} pollution dataset available at [14], which was previously used for experimental evaluation in [15]. The dataset consists of daily-averaged pollution measurements from time period 2006-2008. The weather dataset for our experiments was obtained from the Polish Institute of Meteorology and Water Management [16]. All the experiments were performed with the following values of the parameters of OeSNN-IP: $T_{init} = 200$, $\mathcal{W}_{size} = 2$ days, $simTr = 0.05$, $mod = 0.9$, and $\mathbf{NO}_{size} = 200$. Each data stream consisted of 1096 input values. The error assessment of predicted results was conducted for input values starting from $T_{init} + 1$ value.

In the first experiment, we compared the results of air pollution prediction obtained with OeSNN-IP to the results provided in [15]. In Table I, we present the prediction results reported in [15] for $t_{pred} = 1$ day for the following methods: RBF - Radial Basis Function neural network, SVR - Support Vector Regression, EN - Elman networks, MLP - Multilayer Perceptron neural network, ARX - Auto-Regressive with eXogenous input predictor, the combination of these methods with wavelet decomposition of input signals as well as the results obtained with the proposed OeSNN-IP approach and the NeuCube implementation of eSNN, which was proposed in [17].

Table I presents the results of the first experiment obtained using only the data stream of pollution feature F and the number of input neurons $NI_{size} = 70$. As follows from this table, in terms of all the used prediction quality measures, OeSNN-IP outperforms all the compared methods (RBF, SVR, EN, MLP, ARX) when they are used without the wavelet decomposition as well as MLP and ARX methods when they are used with the wavelet decomposition (denoted as MLP + W and ARX + W). In the case of the remaining three methods: RBF, SVR and EN used with the wavelet decomposition (denoted as RBF + W, SVR + W and EN + W), OeSNN-IP is highly competitive. Also, OeSNN-IP outperforms the NeuCube implementation of eSNN [17], which is available without the wavelet decomposition.

Based on the results presented in Table I, we can conclude that OeSNN-IP provides better prediction results than the other compared methods in terms of RMSE and MAE errors. For the MAPE error, the results obtained with OeSNN-IP are second best, while in terms of the IA agreement measure, OeSNN-IP, SVR + W and EN + W methods provide the same value of this prediction quality measure.

In Fig. 3, we present the results of the next experiment, which compares the obtained RMSE error for OeSNN-IP for $t_{pred} = 1, 2, \ldots, 12$ days with varying number of conditional weather features. Since the proposed OeSNN-IP approach is sensitive to correlations between pollution feature and conditional features, we show the RMSE error in four cases: I. when only pollution feature (F) was used for prediction; II. when pollution feature (F) and conditional wind speed feature (C_1) were used; III. when pollution feature (F) as well as

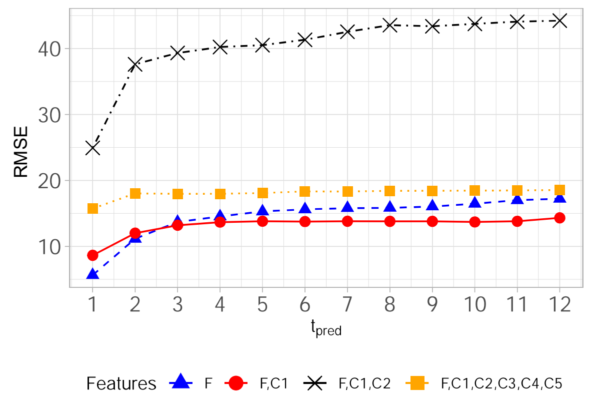


Fig. 3. Comparison of the obtained RMSE results for OeSNN-IP for $\Delta t = 12$ days. The features are as follows $F-$ PM_{10}, C_1- wind speed, C_2- avg. temp., C_3- precipitation, C_4- humidity, C_5- cloudiness.

two conditional features: wind speed (C_1) and daily average temperature (C_2) were used; IV. when pollution feature (F) and all conditional weather features were used for pollution prediction. It follows from Fig. 3 that the best prediction results in terms of RMSE were obtained when using only air pollution feature (F) in short time horizon ($t_{pred} = 1$ or $t_{pred} = 2$), while in longer time horizon ($t_{pred} \geq 3$), the best prediction results were obtained when using both F and wind speed (C_1).

Finally, we compared the efficiency and effectiveness of OeSNN-IP with its variant which uses the GRFs technique (previously used in eSNN and OeSNN networks [10], [13]) instead of our proposed encoding technique (see subsection II-C). In this experiment, all the parameters were the same as previously, with the exception for NI_{size} parameter whose values were changed from 40 to 300. Fig. 4 presents the prediction quality (in terms of the RMSE error) and runtimes of both variants of OeSNN-IP for the Warsaw-Ursynow dataset. The performed experiments show that the proposed encoding technique is more computationally efficient and assures lower values of RMSE error in prediction than the GRFs encoding technique.

IV. Conclusions

In this article, we offered the novel model, called Online evolving Spiking Neural Network for Incremental Prediction (OeSNN-IP), capable of forecasting from data streams for several time points ahead. We employed it to air pollution prediction based on earlier values of an air pollution data stream and on weather data streams. OeSNN-IP makes predictions of pollution values and learns from a given number of recent values in streams of pollution and weather data rather than only from the most recent value from each data stream. We proposed a method for determining values of OeSNN-IP network in such a way so that older stream values have less influence on predicted values than newer ones. In addition, we offered the new fast and effective technique for encoding input values into input neurons order values and synapses

TABLE I

COMPARISON OF THE OBTAINED AIR POLLUTION PREDICTION RESULTS FOR THE PROPOSED OESNN-IP APROACH AND SEVERAL OTHER APPROACHES (THE RESULTS FOR METHODS MARKED WITH * ARE PROVIDED IN [15]). THE RESULTS PRESENTED IN THE TABLE ARE REPORTED FOR $t_{pred} = 1$ DAY.

	*RBF**	*SVR**	*EN**	*MLP**	*ARX**	*RBF + W**	*SVR + W**	*EN + W**	*MLP + W**	*ARX + W**	*NeuCube*	*OeSNN-IP*
Error MAE_1 [μg/m^3]	9.88	8.66	9.48	10.48	10.56	4.92	4.11	4.32	5.13	6.43	11.64	**4.04**
Error RMSE_1 [μg/m^3]	16.43	14.69	15.44	16.72	17.11	7.24	5.93	6.16	6.94	8.46	16.26	**5.65**
Error MAPE_1 %	31.34	27.47	31.12	35.42	35.49	17.31	**13.99**	14.50	18.12	20.22	46.78	14.65
Agreement IA_1	0.74	0.73	0.73	0.69	0.67	0.94	**0.95**	**0.95**	0.94	0.84	0.62	**0.95**

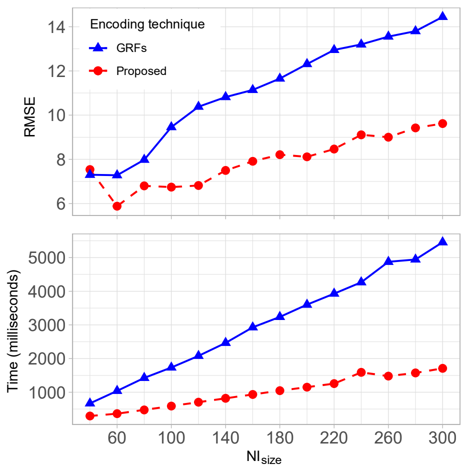


Fig. 4. The RMSE error and runtime of two variants of OeSNN-IP: (i) with the proposed encoding technique, and (ii) with the GRFs encoding technique ($t_{pred} = 1$ day; only pollution feature F was used for the prediction).

weights. Moreover, we formulated the tight upper bound on the Euclidean distance between synapses weights vector of an output neuron and synapses weights vector of a candidate output neuron. We applied this result to simplify the selection of a similarity threshold used in the learning phase of the OeSNN-IP model when a candidate output neuron is compared with output neurons in the repository. We carried experiments related to prediction of future air pollution values using Warsaw-Ursynow pollution data stream and corresponding weather data streams. The experiments showed that OeSNN-IP prediction is at least highly competitive results to a number of state-of-the-art methods and algorithms. We also proved that the proposed encoding technique is more efficient and ensures better prediction quality than the GRFs encoding technique, which is commonly used for evolving spiking neural networks.

REFERENCES

[1] M. Martuzzi, F. Mitis, I. Iavarone, and M. Serinelli, "Health impact of pm10 and ozone in 13 Italian cities," *WHO Regional Office for Europe*, p. 133, 2006.

[2] K. Naddafi, M. S. Hassanvand, M. Yunesian, F. Momeniha, R. Nabizadeh, S. Faridi, and A. Gholampour, "Health impact assessment of air pollution in megacity of Tehran, Iran," *Iranian Journal of Environmental Health Science & Engineering*, vol. 9, no. 1, p. 28, Dec 2012.

[3] P. S. Maciag, "Efficient discovery of top-k sequential patterns in event-based spatio-temporal data," in *Proceedings of the 2018 Federated Conference on Computer Science and Information Systems, FedCSIS 2018, Poznań, Poland, September 9-12, 2018*, 2018, pp. 47–56.

[4] P. S. Maciąg, N. Kasabov, M. Kryszkiewicz, and R. Bembenik, "Air pollution prediction with clustering-based ensemble of evolving spiking neural networks and a case study for London area," *Environmental Modelling & Software*, vol. 118, pp. 262 – 280, 2019.

[5] A. J. Badyda, J. Grellier, and P. Dąbrowiecki, *Ambient PM2.5 Exposure and Mortality Due to Lung Cancer and Cardiopulmonary Diseases in Polish Cities*. Cham: Springer International Publishing, 2017, pp. 9–17.

[6] A. Kurt and A. B. Oktay, "Forecasting air pollutant indicator levels with geographic models 3 days in advance using neural networks," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7986 – 7992, 2010.

[7] X. Feng, Q. Li, Y. Zhu, J. Hou, L. Jin, and J. Wang, "Artificial neural networks forecasting of pm2.5 pollution using air mass trajectory based geographic model and wavelet transformation," *Atmospheric Environment*, vol. 107, pp. 118 – 128, 2015.

[8] X. Mao, T. Shen, and X. Feng, "Prediction of hourly ground-level pm2.5 concentrations 3 days in advance using neural networks with satellite data in eastern China," *Atmospheric Pollution Research*, vol. 8, no. 6, pp. 1005 – 1015, 2017.

[9] B. Petro, N. Kasabov, and R. M. Kiss, "Selection and optimization of temporal spike encoding methods for spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2019.

[10] J. L. Lobo, I. Oregi, A. Bifet, and J. D. Ser, "Exploiting the stimuli encoding scheme of evolving spiking neural networks for stream learning," *Neural Networks*, vol. 123, pp. 118 – 133, 2020.

[11] P. S. Maciag, M. Kryszkiewicz, R. Bembenik, J. L. Lobo, and J. D. Ser, "Unsupervised anomaly detection in stream data with online evolving spiking neural networks," *CoRR*, vol. abs/1912.08785, 2019. [Online]. Available: http://arxiv.org/abs/1912.08785

[12] J. L. Lobo, J. D. Ser, A. Bifet, and N. Kasabov, "Spiking neural networks and online learning: An overview and perspectives," *Neural Networks*, vol. 121, pp. 88 – 100, 2020.

[13] J. L. Lobo, I. Laña, J. Del Ser, M. N. Bilbao, and N. Kasabov, "Evolving spiking neural networks for online learning over drifting data streams," *Neural Networks*, vol. 108, pp. 1 – 19, 2018.

[14] Chief Inspectorate For Environmental Protection. (2015) Warsaw-Ursynow pollution data 2006-2008. [Online]. Available: http://powietrze.gios.gov.pl/pjp/current?lang=en#

[15] K. Siwek and S. Osowski, "Improving the accuracy of prediction of pm10 pollution by the wavelet transformation and an ensemble of neural predictors," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 6, pp. 1246 – 1258, 2012.

[16] Polish Institute of Meteorology and Water Management - National Research Institute. (2017) Warsaw-Bielany weather data 2006-2008. [Online]. Available: http://powietrze.gios.gov.pl/pjp/current?lang=en#

[17] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Netw.*, vol. 52, pp. 62–76, Apr. 2014.