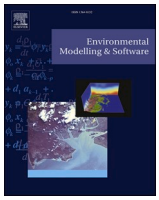
Environmental Modelling and Software 152 (2022) 105387

Contents lists available at ScienceDirect

Environmental Modelling and Software

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



Hazardous atmospheric dispersion in urban areas: A Deep Learning approach for emergency pollution forecast

Mouhcine Mendil [a,*], Sylvain Leirens [a], Patrick Armand [b], Christophe Duchenne [b]

[a] Univ. Grenoble Alpes, CEA, Leti, F-38000, Grenoble, France
[b] CEA, DAM, DIF, F-91297, Arpajon, France

ARTICLE INFO

Keywords:
Deep learning
Hazardous pollutant
Dispersion simulation
Surrogate model
Synthetic data

ABSTRACT

Today, Computational Fluid Dynamics approaches have a high level of spatial/temporal accuracy in modelling atmospheric transport and dispersion in very complex environments. Several numerical models require, however, heavy computational resources and prolonged simulation time up to several days. This time constraint is specifically crucial for intervention planning in case of accidental or malevolent toxic releases in a city. In this paper, we propose to use synthetic data generated by a realistic 3-D transport/dispersion simulator, to train a learning framework called **MCxM**. The latter relies on a sequence of masking and correction operations to progressively apply the spatial constraints and underlying physics of transport and dispersion. The learning phase uses the urban geometry of the French city Grenoble. We then test the effectiveness of the trained **MCxM** in a different French city: Paris. The results show that the **MCxM**'s forecasts are virtually instantaneous and generalize successfully to unseen conditions.

1. Introduction

1.1. Context

Air pollutants are substances of different natures that raise many issues for the population. Being exposed to such pollutants can provoke several effects from minor discomforts (e.g. bad odor) to serious health problems (such as cancer, lung infection and birth defects).

Particularly, accidental or hostile emissions of contaminants are highly dangerous events that necessitate an immediate intervention strategy from the authorities to protect the population and limit the material and environmental casualties. For this purpose, dispersion modelling is used to forecast the concentration of a pollutant at different distances and directions from the sources. The models can compare exposures to some selected benchmark, such as a state pollution standard or a level with a known health effect, to provide recommendations to decision makers (e.g., US legislation (United States Environmental Protection Agency)). For example, in an emergency planning study (Lindell, 1995), it was noted that hazardous plumes emanating from a waste incinerator in the USA could arrive in populated areas before people could evacuate or while they are evacuating. A shelter-in-place strategy would be recommended instead (Sorensen, 2004).

State of the art models of air pollution transport and dispersion, which belong to *Computational Fluid Dynamics* (CFD) family, are characterized by large memory needs and computational complexity. They usually require high-performance hardware and several hours of runtime (Zannetti, 1990). Accidental pollutant release would ideally require an immediate response from the authorities to contain any potential hazard for the population and the environment. In this regard, the runtime overhead of atmospheric transport and dispersion simulations may hinder them from being consistent with decision making under time constraints, especially if such simulations have to be executed several times for sensitivity or uncertainty evaluations.

1.2. Related work

In the past decade, many researchers explored the idea of leveraging *Machine Learning* (ML) to predict atmospheric pollution (Aguilera et al., 2011; Zhang et al., 2020; Araujo et al., 2020). This trend stems from the ability of *Artificial Intelligence* (AI) to approximate extremely complex systems from observational data. Compared to CFD, a trained *Artificial Neural Network* (ANN) is generally time-efficient and less demanding in

Abbreviations: ADE, Advection-Diffusion equation; AI, Artificial Intelligence; ANN, Artificial Neural Network; CFD, Computational Fluid Dynamics; DL, Deep Learning; DNN, Deep Neural Network; ML, Machine Learning; MSE, Mean Squared Error; PMSS, Parallel Micro-SWIFT-SPRAY.

* Corresponding author.
E-mail address: mouhcine.mendil@gmail.com (M. Mendil).

https://doi.org/10.1016/j.envsoft.2022.105387
Received 15 October 2021; Received in revised form 21 March 2022; Accepted 29 March 2022
Available online 1 April 2022

terms of computational resources. These properties make it a suitable solution for predicting the exposure to a hazardous release, where the assessment time frame is a serious restriction. A comprehensive review (Cabaneros et al., 2019) listed recent research activities on air quality, with recommendations on how to build ANNs for predicting long term air pollution. Regarding short term pollution events, Lauret et al. (2016) solved the *Advection-Diffusion equation* (ADE) using cellular automata whose transition function is learned by an ANN. The wind velocity field is, however, considered known in advance as an input variable. In another study, Wang et al. compared two ML models using the Prairie Grass dataset which refers to a typical hazardous gas release with low stack emission in a flat terrain (Wang et al., 2018). With 20 meteorological inputs, the selected neural network architecture (two hidden layers of 38 and 4 neurons respectively) showed the best performance for a learning phase of 4.4 s and a prediction time of 8 ms. A subsequent study (Ni et al., 2020) complemented Wang's work (identical dataset) by adding two *Deep Learning* (DL) models to the benchmark. Results showed a significant leap in performance with *Deep Neural Networks* (DNNs) compared to classic ML models for a learning phase of 406.3 s and a prediction time of 36 ms.

In the aforementioned works, DL has a better performance compared to classic ML techniques in fitting atmospheric dispersion data. However, the used datasets are associated with simple flat terrains, whose properties are completely different from those in urban areas. The learning problem is therefore simpler, and its feasibility cannot generalize to complex urban geometries.

Few studies assessed the performance of AI applied to fluid mechanics in presence of turbulence. Lauret et al. (2014) used the same combination of cellular automata with ANNs to solve the ADE (Lauret et al., 2016), but this time with a new consideration: the impact of a simple obstacle on the turbulent diffusion coefficient. This parameter is learned and estimated by another ANN prior to solving the ADE. The obtained results showed that the proposed model runs 1.5 times faster than CFD calculations and the diffusion coefficient is fairly predicted. The main drawback here is the assumption that the wind velocity field is a precomputed (via CFD) input of the ANN. Yet, transport simulation requires a significant amount of time (for example, for FLUENT 6.3, 99% of the time is dedicated to wind field computation and 1% for the dispersion (Vendel, 2011)). In a study entitled "Hidden Fluid Mechanics" (Raissi et al., 2018), a DNN was trained to predict the concentration and latent quantities (such as velocity and pressure) by including into the cost function residual errors from the conservation equations (mass and momentum). A showcase of 2-D and 3-D experiments revealed the capacity of DL techniques to capture with high accuracy the turbulence caused by various obstacles. However, the algorithm (or alternatively the discovery of partial differential equations affected by boundary conditions) has to be trained for every new terrain, making it highly unpractical given the geometrical diversity of urban domains.

1.3. Contributions and organization

To the best of our knowledge, there is a serious lack of ML-based surrogate models that provide a fast and reliable assessment of pollution incidents in urban areas. Regarding this limitation, our main contributions are the following:

1. We propose a novel learning framework called **MCxM** to estimate the concentration level of pollutants released in urban areas. In case of an emergency event, such surrogate model is provided with actual meteorological and topographical data to infer the pollutant dispersion.
2. Unlike purely data-centric approaches in DL, we propose a physically interpretable design of our learning framework. Mainly, two key stages are considered: (a) Masking: straightforward enforcement of the spatial constraints generated by the urban topology on the airflow. (b) Correction: modelling of the effects due to the obstacles (buildings) and progressive estimation of the pollutant distribution.
3. We propose to use synthetic data produced by a realistic 3D multi-scale atmospheric dispersion simulator (PMSS) for model training and testing. Specifically, the simulations are configured to cover several wind conditions and source locations in the cities of Grenoble and Paris (France).

The rest of the paper is organized as follows. The proposed learning approach is provided in Section 2. In Section 3, we explain how the training dataset is generated and give details about the test procedure. Section 4 discusses the obtained results, highlighting the benefits and limitations of the proposed framework. Finally, conclusions are drawn in Section 5.

2. Methods

2.1. Problem scope

The aim of this paper is to provide a ML-based surrogate model of atmospheric transport and dispersion that is able to quickly and accurately predict the pollutant's concentration dose released by accident in an urban area.

The studied area is a neighborhood of a surface up to few squared kilometers. The pollutants' doses are computed over an integration time window of maximum a couple of hours. Therefore, we consider only small-scale turbulence in the dispersion phenomena, due to mechanical phenomena for the most part (Zannetti, 1990). Mechanical turbulence comes from surface rugosity (mainly of the obstacles) and manifests as a deflection and/or a vortex circulation of the airflow at the interface of obstacles (Hunter et al., 1992).

We summarize the previous considerations in the three following hypotheses to characterize the dispersion conditions:

- *Hypothesis 1*: the release conditions consist in an instantaneous emission of the pollutant from a single point source (x_0, y_0, z_0) at human height.
- *Hypothesis 2*: the weather conditions are restricted to the wind speed and direction over the urban canopy.
- *Hypothesis 3*: the environment consists of obstacles (buildings) of different heights and shapes located inside the urban area.

Within these hypotheses, we propose, in the sequel, a learning approach based on DL to forecast the integrated pollutant concentration on a 2-D section at the same height z_0 as the emission source. A brief overview of DL is provided before describing the learning workflow.

2.2. Introduction to DL

A DNN is a non-linear statistical data modelling system that transforms a set of inputs into a set of outputs through multiple layers (sequence) of computations. The goal of DNNs is to infer from data the underlying phenomena that process the inputs values into the resulting outputs. For instance, they are used for pattern recognition and regression in various fields such as physics, health and finance.

In a DNN, the predictor variables are assigned to the input layer (L_0). Fig. 1 illustrates the operations performed in all subsequent layers. Each output of a given layer L_i is the result of applying a non-linear function (called activation) to a linear combination of L_i's inputs (which are also the outputs of the preceding layer L_{i-1}). Thanks to the universal approximation property (Hornik et al., 1989), this structure is able to fit any non-linear function by adjusting the weights and offsets of the linear combinations in the network. The second property that allows DNNs to learn efficiently is the principle of parsimony. Parsimonious models are presumably simple models with great explanatory predictive power. They explain data with a minimum number of predictor variables

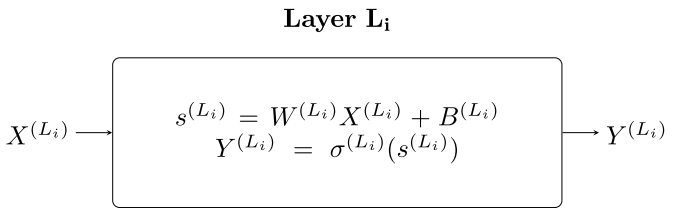


Fig. 1. Composition of a DNN's layer: $X^{(Li)}$ is the layer's input vector. Note that $X^{(Li)}$ equals the output of the previous layer L_{i-1}. $W^{(Li)}$ is the matrix whose elements represent the weight assigned to the synapses connecting the input of neurons to the output of neurons in layer L_i. $B^{(Li)}$ is the bias vector associated with the output neurons of layer L_i and $\sigma^{(Li)}$ a non-linear activation function.

(hypotheses). By limiting the number of input variables, we avoid building complex models that ultimately lead to overfitting.

The learning process consists in calibrating the parameters of the network (weights $W^{(Li)}$ and biases $B^{(Li)}$) to decrease the model's error. The error function represents a distance metric between the model's output and observational data. In the learning phase, the parameters are updated by applying a gradient-descent algorithm using the training set. The latter step, called back-propagation, uses automatic differentiation to compute the gradients of the error function with respect to the weights and biases of the DNN. Automatic differentiation relies on combining the derivatives of the constituent operations through the chain rule that gives the derivative of the overall composition, enabling accurate evaluation at machine precision (Baydin et al., 2018).

After each training cycle, the DNN model is validated on a disjoined set of data (validation set). Lastly, the performance of the model is evaluated on another set (test set), never used during training or validation. The validation step is necessary to adjust the complexity of the DNN, that is the number of layers, neurons and connections (called synapses) between neurons of consecutive layers. Vapnik (1999) introduced the notion of model capacity, which conceptually represents the space of functions the DNN can fit. Increasing the number of layers and/or their neural density enables it to fit more complex non-linear transformations. However, the obtained function may model the random noise in the training data rather than the governing principles. This problem, called overfitting, results in building models that explain well the data at hand, but fail in out-of- sample predictions. On the other hand, DNNs with low capacity are impractical to solve complex tasks and tend to underfit. This trade-off is also known in statistics as the bias-variance trade-off. Both overfitting and underfitting models lack the ability to forecast the correct output.

2.3. MCxM framework

2.3.1. Overview

Fig. 2 represents an example of the learning approach, namely **MCxM**, composed of a sequence of four masking and three correction stages (that we explain later) that incrementally build the pollution field through multiple non-linear transformations. The input data are as follows:

- the wind's speed ν and direction θ above the urban canopy and
- the 2-D binary map of the urban area, denoted by $\mathbf{M_C}$, that encodes the presence of obstacles as 0 and the fluid zones as 1. Note that this is a 2-D section at height z_0 of a 3-D building map.

The output is a prediction of a 2-D integrated concentration map at the source height z_0, over a window [0, T] starting from the instant of release up to a duration T.

Note that the position of the emission source varies from one experiment to the other (for example, possible source locations are represented by red dots in Fig. 4 and Fig. 5). In spite of that, the source location is considered at the center of the computation domain at every stage of our approach. With this assumption, the source coordinates are not needed as input parameters, thereby simplifying the learning model. The counterpart is a pre-processing effort of the urban area maps and the associated concentration fields to center them on the location of the source (further details in Section 3.2).

We are in presence of a multivariate regression task with two heterogeneous input data types: a column vector and a matrix. A straightforward approach to data fusion would be to vectorize the urban area map and concatenate it with the wind vector. This results in mixing high level physical parameters with the local spatial representation of the urban terrain. Preliminary experiments showed that this solution performs poorly as heterogeneous data sources require different processing. An alternative relies on creating two separate processing streams that construct, after a succession of learned operations, mergeable data representations (Zhang et al., 2018; Tatarchenko et al., 2015; Audebert et al., 2017). This approach offers better performances but increases the learning complexity. Besides, in our case, it is clear that constructing a two-dimensional array based only on two wind scalars will lead to a poor

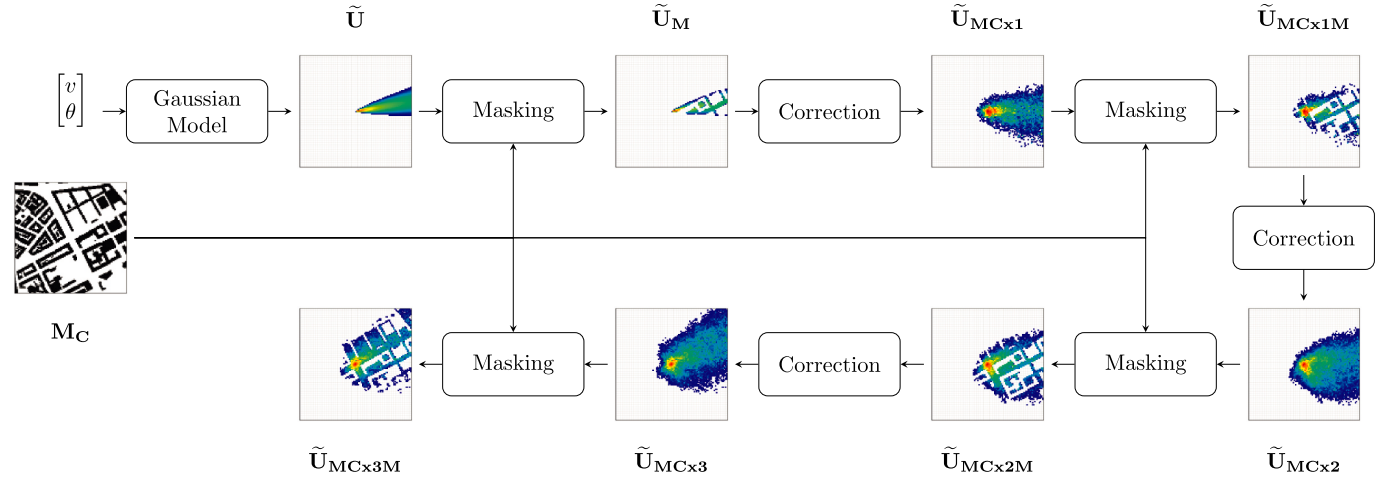


Fig. 2. Illustration of different masking and correction stages of the learning approach: the two-dimensional array $\widetilde{U}$ is obtained through integration of the instantaneous concentrations produced by the Gaussian plume model. A first masking stage applies the urban spatial constraints on $\widetilde{U}$ (data fusion) to produce $\widetilde{U}_M$. $\widetilde{U}_{MC}$ is the result of a correction stage that approximates the non-linear effects on the airflow due to the buildings, but infringes the spatial constraints. The latter are reinstated with the following masking stage. By stacking two additional correction and masking stages, we obtain $\widetilde{U}_{MC\times 2}$ and $\widetilde{U}_{MC\times 3}$.

and redundant representation. Instead, we propose to build a physics-informed representation from the meteorological data using the Gaussian plume model.

2.3.2. Gaussian plume model

The Gaussian plume model describes the three-dimensional concentration field generated by a point source under stationary meteorological conditions. The dispersion is considered here in an ideal flat environment, principally caused by the airflow transport. Given $\vec{x}$ the co-linear axis with the wind direction vector (at angle θ), the Gaussian plume formula is as follows:

$$c(x, y, z_0, t) = \frac{Q}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{(x - vt)^2}{2\sigma_x^2}\right) \exp\left(-\frac{y^2}{2\sigma_y^2}\right) \quad (1)$$

where $c(x, y, z_0, t)$ is the concentration at (x, y, z_0) and instant t due to the emission at (x_0, y_0, z_0), Q is the emission rate, v is the wind speed and (σ_x, σ_y) are the horizontal standard deviations of the spatial Gaussian plume distribution which are computed following Pasquill-turner correlation formula: $\sigma_x = \sigma_y = a\,x^b + c$ (Hanna et al., 1982). The numerical expression of these parameters is given is Section 3.2.

Thereafter, we calculate the integrated concentration over a time window of duration T, for the 2-D section at the elevation of the source $z = z_0$:

$$c_I(x, y, z_0) = \int_0^T c(x, y, z_0, t)dt. \quad (2)$$

The obtained cartography of the concentration doses, noted $\widetilde{U}$, is so far unaffected by the urban geometry.

2.3.3. Masking and corrections

The masking stage accounts for some physical obstacles of the urban area. It operates an element-wise matrix multiplication between the binary map $\mathbf{M_C}$ and a pollutant distribution $\widetilde{U}_X$ to zero the concentration level where obstacles are located:

$$\widetilde{U}_{XM} = \widetilde{U}_X \odot \mathbf{M_C}, \quad (3)$$

where $\widetilde{U}_X \in \{\widetilde{U}, \widetilde{U}_{MC\times1}, \widetilde{U}_{MC\times2}, \widetilde{U}_{MC\times3},\}$

The first masking applied to the Gaussian plume $\widetilde{U}$ is a way to merge the heterogeneous input data (wind scalars and 2-D urban map) by restricting the pollution to the fluid zones encoded in $\mathbf{M_C}$. Subsequent masking stages are applied to the "corrected" concentration fields (we explain this below) with the same objective of enforcing the spatial constraints associated to the urban area. However, the masking process itself does not capture any physical interaction between the airflow and the buildings, such as trajectory change and mechanical turbulence. Furthermore, it introduces many non-physical artifacts and abrupt discontinuities in the concentration field (e.g. see $\widetilde{U}_M$ and $\widetilde{U}_{MC\times1}$ in Fig. 2). The correction stages are introduced specifically to mitigate these inconsistencies.

In the correction stage, the physics related to the transport and dispersion of the pollutant are approximated. It operates learned non-linear transformations on masked pollutant distributions ($\widetilde{U}_M$, $\widetilde{U}_{MC\times1M}$ and $\widetilde{U}_{MC\times2M}$) and outputs their corrected counterparts ($\widetilde{U}_{MC\times1}$, $\widetilde{U}_{MC\times2}$ and $\widetilde{U}_{MC\times3}$ respectively). One consequence of the correction stage is the modification of the area reached by the pollutant. It is mainly due to the buildings deflecting the airflow that transports the pollutant outside of its initial range. This observation is important for two reasons: first, the pollutant distribution likely requires several iterations of correction before reaching its final extent. Second, there is a need for an additional masking after each correction to apply spatial constraints to the newly polluted areas. The latter update is necessary because the binary masking only identifies obstacles within the reach of the pollutant, that becomes obsolete as the pollutant distribution extends after every correction stage.

We end up with a succession of masking and correction stages that progressively model the impact of different obstacles on the expanding polluted areas. Hence the architecture's name **MCxM**, where the number after **x** stands for the length of the masking(M)/correction(C) sequence.

When using the **MCxM** framework, each correction stage can consist of a separate DNN but requires a joint training during the learning phase, i.e. the weights and biases have to be tuned together. For the sake of consistency and interpretability, we propose to share the same weights and biases between the correction DNNs in all stages, as a way to have them model "universal" physics of atmospheric transport and dispersion. This is also a regularization technique to reduce the computational complexity. An example of architecture for the correction DNN is given hereafter and the length of the masking and correction sequence will be motivated in Section 4.

2.3.4. Example of a correction DNN

When dealing with 2-D data, most common learning architectures either use image processing operations (such as convolutions) or process encoded data in less complex subspaces before recasting them into the original space. For the sake of illustration, we will use an encoder/decoder DNN represented in Fig. 3 to learn the correction operations. This is not necessarily the optimal architecture but only a practical example. More generally, the proposed approach has the advantage of being interoperable with various non-linear ML models.

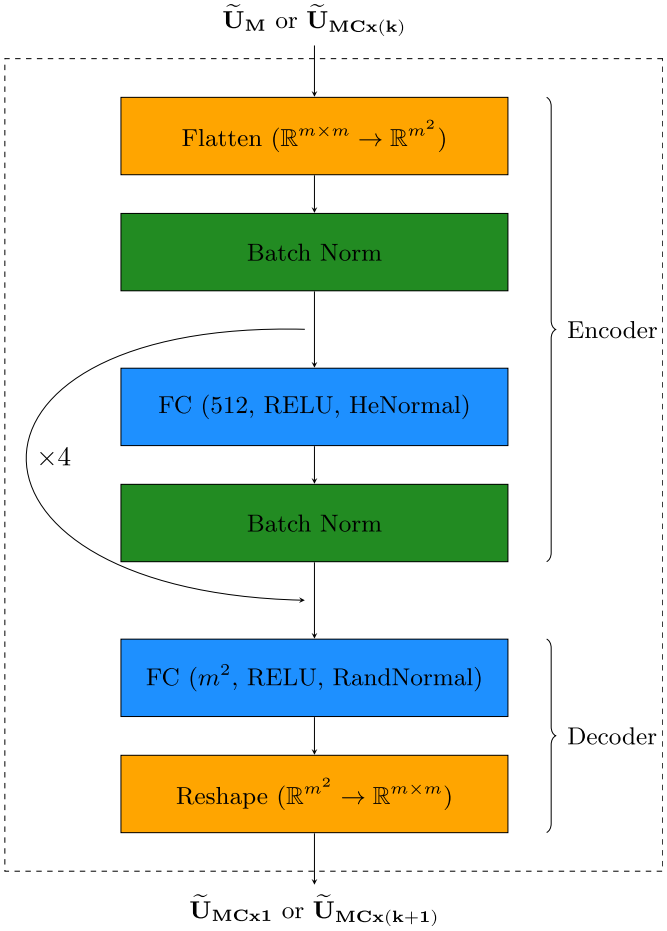


Fig. 3. Correction encoder/decoder architecture. $\widetilde{U}_M \in \mathbb{R}^{m\times m}$ (resp. $\widetilde{U}_{MC\times(k)}$) is the input array and $\widetilde{U}_{MC\times1} \in \mathbb{R}^{m\times m}$ (resp. $\widetilde{U}_{MC\times(k+1)}$) is the output array. The encoder represents and processes the input in a lower dimension. Afterwards, the decoder reconstructs the data in the original space.

The encoder block starts with a vectorization of a $m \times m$ matrix. With $m = 100$ for example, it creates a considerable amount of features ($m^2 = 10{,}000$) to be processed. The following hidden layers use significantly fewer features (512 units per layer, which will be motivated in Section 4) by transforming them into compact/high-level representations.

In the hidden layers of the encoder and decoder blocks, we use the Rectified Linear Unit (ReLU) as non-linear activation function (Agarap, 2018). As mentioned before, stacking multiple hidden layers is what makes DL approximate complex non-linear transformations. However, if deeper networks are more powerful, they also suffer from vanishing (or exploding) gradients problems that seriously limit the learning performance (Glorot and Bengio, 2010). The use of ReLU (instead of saturated functions like sigmoid) partially mitigates this problem. Additionally, we rely on batch normalization (Ioffe and Szegedy, 2015) and He initialization (HeNormal) (He et al., 2015) that achieve and maintain comparable variances of incident and outgoing weights throughout the encoder DNN. The batch normalization layers are placed before and after each of the four hidden layers of the encoder network. Finally, the decoder reconstructs the data in the original space through a full connected layer of m^2 features, rearranged in a matrix $m \times m$.

3. Experiment settings: study case of hazardous pollutant release in French urban areas

3.1. Synthetic data and CFD simulation

The dataset in DL is of a vital importance. A DNN mainly finds recurring patterns on the training data to build a governing relationship between the input and output variables. The more situations are covered, the more powerful the inference capability the model will have. In case of transport and dispersion modelling, data can be collected from 1) real size experiments or 2) small scale experiments in wind tunnels. The former solution is expensive (data acquisition devices, release systems, gases to be released, workforce and site availability for the experiment) and needs long acquisition campaigns to capture natural cycles (e.g., day-night and seasons). Moreover, it is impossible to control the weather conditions. The latter alternative is not only costly but also suffers from scaling adjustments since the experiments are usually carried out in a 1/50 scale. The database can also be generated from CFD calculations, given that most sophisticated models provide a very high accuracy. In the simulation environment, the weather characteristics, the properties of the chemical release and the geometry of the terrain can be configured with no constraints. Synthetic data have been used in many studies (Bolón-Canedo et al., 2013; Ayturan et al., 2018) and are extremely efficient to train AI systems.

We use *Parallel Micro-SWIFT-SPRAY* (PMSS) as the numerical modelling system of transport and dispersion. It is a parallel (space and time decomposition) version of Micro-SWIFT-SPRAY composed of:

- Micro-SWIFT: an analytically modified mass consistent interpolator over complex terrain. Given topography, meteorological data and buildings, a mass consistent 3-D wind field is generated. It is also able to derive diagnostic turbulence parameters to be used by Micro-SPRAY inside the flow zones modified by obstacles.
- Micro-SPRAY: a Lagrangian particle dispersion model able to take into account the presence of obstacles. The dispersion of a pollutant is simulated following the trajectories of a large number of fictitious particles. The trajectories are obtained by integrating in time the particle velocity which is the sum of a transport component defined by the local averaged wind provided generally by Micro-SWIFT, and a stochastic component, standing for the dispersion due to the atmospheric turbulence.

PMSS modelling system has been thoroughly validated against wind tunnel and full-scale experimental results (Oldrini et al., 2017; Castelli et al., 2018). The parallelism was shown to be very efficient, from a multicore laptop up to clusters with several hundreds or thousands of cores in the case of a high-performance computing center (Oldrini et al., 2019; Armand et al., 2021). In terms of computational time, to give an order of magnitude considering a spatial domain of 1 km^3 with a horizontal and vertical (close to the ground) resolution of 2 m, PMSS computational time is around 5 min for the flow and 10 more minutes for tracking a plume for 15 min using an optimal number of cores on a standard work station.

3.2. Pollution Data

We have selected two cities to experiment the **MCxM** framework: Grenoble and Paris. The learning phase (training and validation) of the **MCxM** is performed on synthetic pollution data in Grenoble's city center, while the generalization capabilities (test) are challenged in the district of Opera in Paris. Paris and Grenoble are two French cities characterized by a typical European architecture where the majority of buildings are 20m–25m tall and date from the 19th and 20th centuries. Their urban density – measured as the building square footage divided by land area – is nearly similar: 57% for Grenoble's city-center versus 41% for the Opera district in Paris. The buildings in Grenoble center are slightly more scattered than in Opera (see Figs. 4 and 5). Overall, the selected urban areas are comparable but still exhibit some architectural differences; this enables to show how adaptive the **MCxM** is in predicting pollution exposure in new environments. The configuration of PMSS for training, validation and test is summarized in Table 1 and described in details in the following paragraphs.

The training and validation set (80% and 20% respectively) is composed of 14,796 instances of integrated concentration over $T = 2$ h, generated by PMSS for the following configuration:

- As shown in Fig. (4), the urban area that constitutes the computation domain in PMSS is the center of the French city Grenoble located in the bounding box $x, y \in [913301, 914301] \times [6457391, 6458391]$, expressed in Lambert 93 coordinate reference system. It is an 500×500 shaped grid with a space resolution of 2m.
- 274 hypothetical different locations of the point source are considered.
- 54 stationary weather conditions are considered, built from a combination of 18 values of wind direction θ [°] $\in \{0, 20, 40, \ldots, 340\}$ and 3 values of the wind speed v [m.s^{-1}] $\in \{1.5, 3, 6\}$. We assume neutral atmospheric stability conditions (Pasquill's class D) to compute the semi-empiric standard deviations σ_x and σ_y of the Gaussian plume such that $a = 0.068$, $b = 0.908$ and $c = 0$.

For each experiment, a unique point source is considered to produce an instantaneous emission of a unit mass of the pollutant (gas or particulate matter). For a given initialization (source location and wind conditions), PMSS simulates the wind and dispersion fields for 2 h. Other unspecified attributes (such as temperature or air pressure) are assumed to be constant.

Similarly, the test set is composed of 11,988 instances generated with the same logic given the following configuration:

- The computation domain of PMSS is the Opera district of Paris located in the bounding box $x, y \in [650250, 6863050] \times [651450, 6864050]$ (see Fig. (5)). It is an 600×500 shaped grid with a space resolution of 2m.
- 222 hypothetical different locations of the point source.
- Same 54 stationary weather conditions used for Grenoble.

As mentioned before, the proposed learning procedure considers integrated concentration maps centered on the release source. It is a way to simplify the model by reducing the dimension of the input space (source coordinates are not input variables). Therefore, we process the

Fig. 4. Left: street map of a part of the french city Grenoble and the considered bounding box for the simulation domain; Right: binary encoding of the building topography, where 0 (black) stands for impermeable areas and 1 (white) otherwise. The different source locations are pinpointed with the red dots. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

Fig. 5. Left: street map of the Opera district in Paris and the considered bounding box for the simulation domain; Right: binary encoding of the building topography, where 0 (black) stands for impermeable built areas and 1 (white) otherwise. The different source locations are pinpointed with the red dots. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

Table 1
Design of experiments for training, validation and test.

Location	Train and Validation	Test
	Grenoble city center	Opera District of Paris
Grid Coordinates (Lambert 93)	$x \in [913{,}301, 914{,}301]$ $y \in [6457391, 6458391]$	$x \in [650{,}250, 6863050]$ $y \in [651{,}450, 6864050]$
Resolution	$2m$	$2m$
Height of buildings	$20m$–$25m$	$20m$–$25m$
Area	$1km2$	$1.2km2$
Density	41%	57%
Weather conditions	θ [°] $\in \{0, 20, 40, \ldots, 340\}$ v [m.s^{-1}] $\in \{1.5, 3, 6\}$ Pasquill stability class D: $a = 0.068$, $b = 0.908$, $c = 0$	θ [°] $\in \{0, 20, 40, \ldots, 340\}$ v [m.s^{-1}] $\in \{1.5, 3, 6\}$ Pasquill stability class D: $a = 0.068$, $b = 0.908$, $c = 0$
Pollution source	274 locations uniformly distributed in fluid zones	222 location uniformly distributed in fluid zones

data to extract bounding boxes of dimensions 400m × 400m centered on the source. One of the consequences is the elimination of all source points that fall less than 200m from the borders (274 → 92 sources in Grenoble and 220 → 102 in Opera). The training/validation set is reduced to 4968 instances, and the test set to 5508.

Afterwards, the training/validation data are augmented by rotation with respect to three angles: 90°, 180° and 270°. These rotations preserve the shape of the used square arrays, so that no padding is required. With this, the training/validation set counts approximately 20,000 instances.

4. Results and discussion

All of the following results were obtained by using, during the training phase, the *Mean Squared Error* (MSE) as loss function and *RMSProp* (Tieleman Hintonet al., 2012) as the optimization algorithm with a batch size of 16. Additionally, two regularization strategies were used to make learning tractable and improve the generalization error: 1. Exponential decay of the learning rate: we start the gradient descent algorithm with a higher learning rate (to accelerate the training and avoid spurious local minima) that is decreased in an exponential fashion (to help the network converge to a local minimum and avoid

oscillation). 2. Early stopping: the validation error with respect to the training epoch is a U-shaped curve (assuming a sufficient representational capacity of the model to overfit). During the training phase, we stop the training when the validation error reaches a minimum, achieving the best tradeoff between underfitting and overfitting.

4.1. Hyper-parameters tuning

As mentioned before, the learning capacity of a DNN is directly connected to its hyper-parameters. A poor tuning leads either to underfitting or overfitting, both reducing the generalization performance of the trained model. We propose a straightforward approach based on a grid search to select the number of occurrence and the composition of the correction block(s). The goal here is not an exhaustive search of the optimal hyper-parameters, but just to tune them enough to demonstrate the feasibility of the proposed model. First, we want to find a suitable hyper-parameter setting for the correction block introduced in Section 2. The DNN is trained for different combinations of hidden layers count and their neural density. After convergence, the MSE is evaluated on the validation dataset, normalized with respect to the highest obtained error. Compiling the results in Fig. 6, we observe that the lowest MSE is achieved by an encoder composed of four hidden layers, each containing 512 neurons. Compared to the considered configurations, this encoder configuration yields the best estimator of pollution concentration subsequent to a point hazardous release.

Secondly, we investigate what impact the length of the masking/correction sequence has on the prediction quality. Recall that the succession of correction and masking stages enables to gradually construct the integrated concentration field that accounts for the urban topography. Fig. 7 illustrates the normalized MSE (with respect to the highest error) on the Grenoble's validation dataset with eight **MCxM** architectures of different masking/correction sequence lengths. Note that each correction operation is built with the hyper-parameters selected previously, *i.e.* an encoder composed of four hidden layers each containing 512 neurons. The lowest error is obtained when using a sequence of three consecutive masking and correction stages.[1]

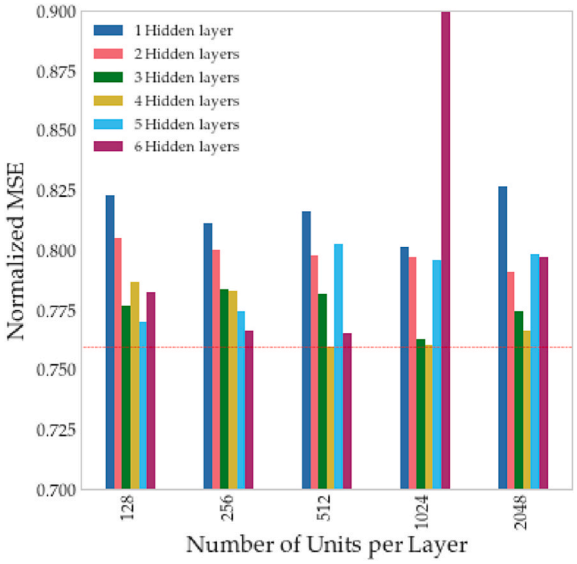


Fig. 6. Hyper-parameters benchmark for the encoder component of the correction stage.

[1] This sequence ends necessarily with a masking stage, even if it is not shown in Fig. 7.

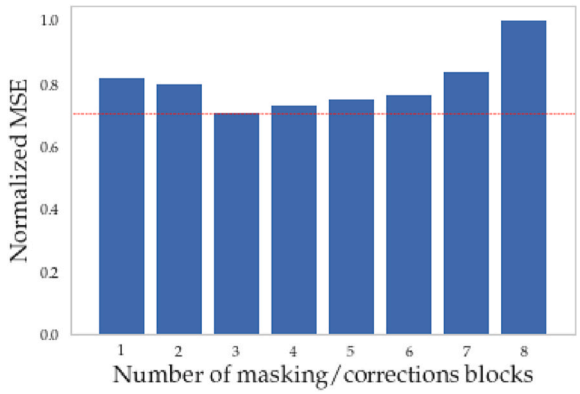


Fig. 7. Masking/correction sequence benchmark. All the correction blocks share the same weights and biases and are each composed of an encoder containing four hidden layers including individually 512 units.

In the remainder of the paper, we consider the **MCx3M** model whose three correction blocks are identical, constituted of four hidden layers and 512 neurons.

4.2. Performance

The **MCx3M** model is trained on 16,000 synthetic examples of accidental pollution dispersion in the city of Grenoble.

The wind direction θ and speed v are input variables used to compute the time-integrated Gaussian plume. The latter represents the concentrations field as if no obstacles were hindering the dispersion of the pollutant. As a reference for upcoming comparisons, we consider the Gaussian model as a **baseline**. The proposed **MCx3M** has to achieve at least better performance than a simplistic Gaussian plume and approximate the turbulence dynamic introduced by the urban geometry.

Fig. 8 represents different prediction stages of the trained **MCx3M** in a new area of Grenoble not encountered during the training phase. The Gaussian plume ($\widetilde{U}$) is narrow and assumes a flat terrain. The first masking ($\widetilde{U}_M$) applies the spatial constraints on the support of the Gaussian plume while the correction ($\widetilde{U}_{MC\times 1}$) increases its dispersion. After the second masking ($\widetilde{U}_{MC\times 1M}$), we can clearly see that the emission source is located in a street intersection. The next correction stage ($\widetilde{U}_{MC\times 2}$) splits the pollutant mass into four directions, with a higher concentration into the streets in the south-east and north-east. The dispersion range is again extended, reaching new areas where the masking ($\widetilde{U}_{MC\times 2M}$) will reveal new intersections to the following correction stage ($\widetilde{U}_{MC\times 3}$) and so on. The three correction stages progressively alter the concentration field such that their normalized average relative deviations[2] from the Gaussian plume are 57%, 96% and 105% respectively. Note that there is a slower deviation rate of the correction in the last stage compared to the first ones, hinting for a convergence process. At the end, this iterative process yields a good approximation of the actual concentration field as simulated by PMSS (considered as ground truth).

Now we evaluate the generalization capacity of the **MCx3M** model

[2] The normalized average relative deviation from the Gaussian plume of each correction stage n is $\frac{1}{n_{rows}\times n_{cols}} \sum_{i,j}^{n_{rows},n_{cols}} \frac{|\widetilde{U}_{MC\times n}(i,j)-\widetilde{U}(i,j)|}{\widetilde{U}(i,j)}$ where (n_{rows}, n_{cols}) are the dimensions of the concentration field matrices, $\widetilde{U}(i,j)$ is the Gaussian plume value at coordinates (i, j) and $\widetilde{U}_{MC\times n}(i,j)$ is the output of correction stage n at coordinates (i, j).

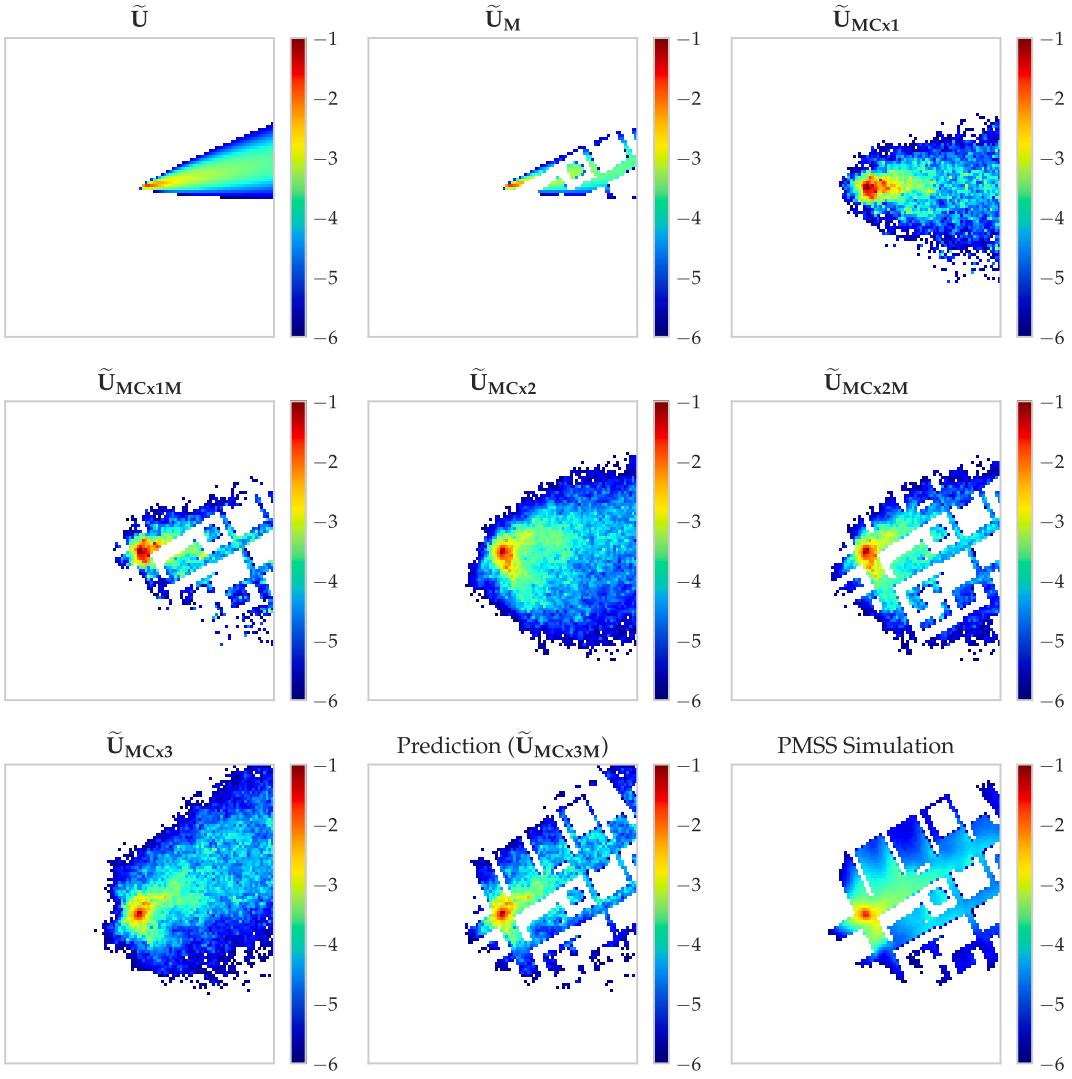


Fig. 8. Example of transformation stages within a trained **MCx3M** model for an unseen release in the city of Grenoble. The integrated concentration fields [unit. s m^{-3}/unit. released] are represented in logarithmic scale.

by computing the prediction error in the city of Paris. The obtained prediction MSE averaged over the whole testset is 2.3 times lower than the MSE of the baseline method (Gaussian model). Consequently, the consecutive masking and correction stages of the **MCx3M** drastically improve the precision of prediction in new urban areas. In Fig. 9, we illustrate four examples of predicted (by the **MCx3M** model) and simulated (by PMSS, considered as the ground truth) integrated concentrations fields over 2 h. Each example corresponds to a hypothetical accidental release by a single point source in a specific location in Paris. Because of the complex topography of urban areas, the airflow can be very turbulent. In particular, the interactions between the buildings and the airflow as well as the effect of street intersections that change the trajectory of the pollutant are accurately modeled by the trained model. The total pollutant mass deviation[3] of the predictions relatively to the simulations are 2%, 12%, −1% and 17% for examples (*a*), (*b*), (*c*) and (*d*) respectively. The small values indicate a good approximation of the total pollutant mass by the **MCx3M**. Additionally the pollutant mass distribution throughout the whole domain is faithfully reproduced by the predictions. The learning model can therefore successfully determine the high risk areas (risky concentration levels of the pollutant that require evacuation for example).

Besides, the **MCx3M** is extremely fast as the computation duration per prediction is 0.75 ms (on a Latitude LE5490 laptop, Intel Core i5, 16 Gbytes RAM). The prediction is virtually instantaneous compared to CFD simulations, even with PMSS which provides a simplified CFD solution in a short time of 15 min for comparable spatial domains. In addition to the accuracy of the prediction, the **MCx3M** speed is adapted for emergency planning in the event of a hazardous release, especially if the model has to be run thousands of times for uncertainty evaluation.

4.3. Limitations and perspectives

Based on the previous results, the proposed **MCxM** framework has shown promising speed and accuracy for modelling transport and dispersion of pollution in urban areas. However, we have only demonstrated such performance in a specific set of hypotheses so far. First, the weather conditions over the urban canopy are considered stationary for the time required by the plume to cross the simulation domain, which is not true in practice. Secondly, the **MCxM** is necessarily sub-optimal because it approaches a 3-D problem based on 2-D data (2-D section

[3] The total pollutant mass M^{Total} is computed by integrating the concentration field over the surface of the 400m × 400m bounding box centered on the source. The total pollutant mass deviation of the prediction relatively to the simulation is $\frac{M^{Total}_{prediction} - M^{Total}_{simulation}}{M^{Total}_{simulation}}$.

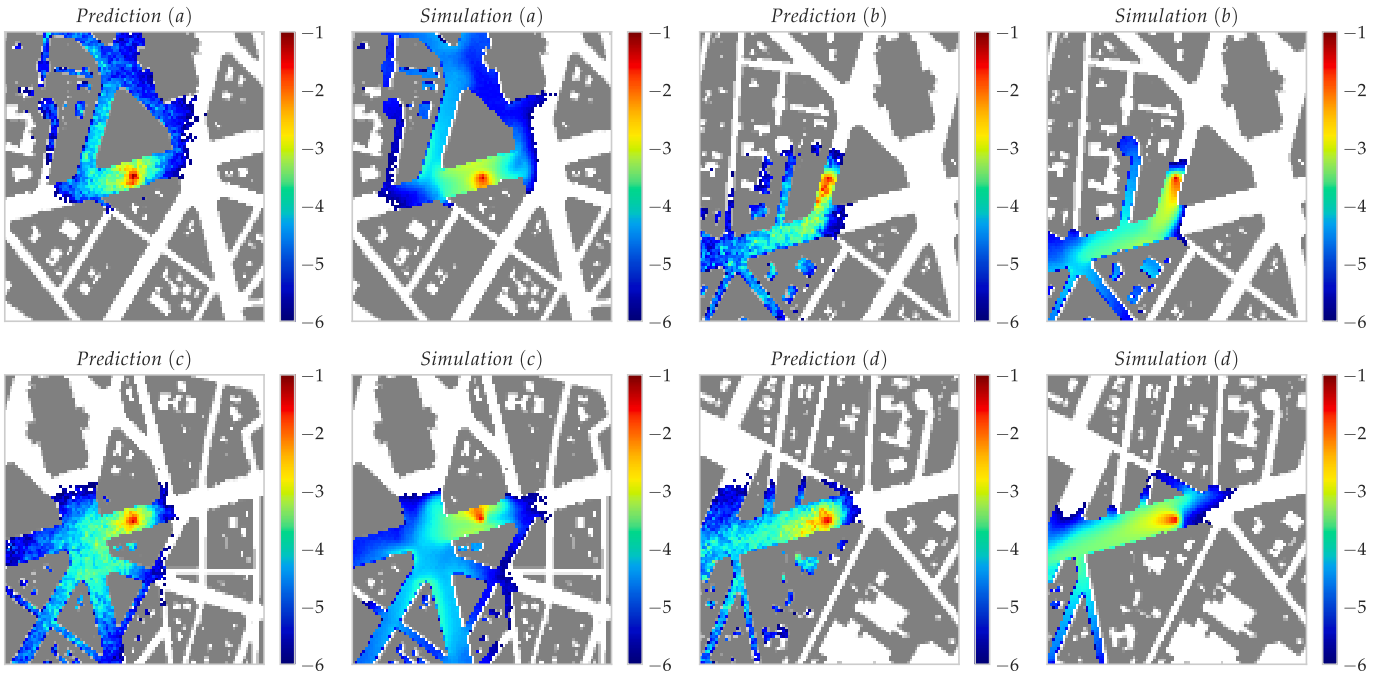


Fig. 9. Examples of predicted (**MCx3M**) and simulated (considered ground-truth) integrated concentration field [unit. s m^{-3}/unit. released] in Paris (logarithmic scale).

of an urban map). A 2-D section of the city map at the input of the **MCxM** does not represent the vertical urban geometry. Therefore, the vertical transport and dispersion phenomena over (or across) the buidings and obstacles cannot be correctly approximated by the learning model. As an example, a closed 2-D contour (such as a private courtyard) at the considered height section should unequivocally be inaccessible to the pollutant from the 2-D perspective of the learning model. However, we observed in many **MCxM** predictions that such areas are still reachable by the air flow. In reality, the learning model does not rely on any physical justification to such behavior but only on the statistical likelihood because it learned from 2-D sections of 3-D concentration fields simulated by PMSS. In that case, if the model's prediction is accurate, it is certainty due to overfitting or randomness rather than a successful approximation of the vertical transport and dispersion dynamics. Besides, we have trained and tested the **MCxM** in urban areas characterized by an almost flat natural topography (no hills, mountains or canyons). The vertical dispersion would otherwise have an even higher impact that is not embedded in the proposed 2-D model.

To address these limitations, we plan in future works to extend the **MCxM** framework by 1. accounting for the mass conservation during the masking and correction stages 2. considering time-changing weather conditions over the urban canopy, 3. integrating atmosphere stability conditions into the model and the simulations, 4. jointly considering multiple sections of the urban area's 3-D geometry, in order to enable 3-D predictions of the pollution concentration fields and 5. experimenting our framework with a larger variety of topographies.

5. Conclusion

In this paper, we propose an atmospheric transport and dispersion surrogate learning model as a fast and accurate decision making tool during hazardous pollution events in urban areas. The **MCxM** learning framework starts by approximating the initial concentration field as a Gaussian plume. The urban geometry is introduced during the masking operation, followed by a correction stage that applies the transport and dispersion physics accordingly to the perceived spatial constraints. These two stages are then iteratively repeated until the final pollution map is obtained. For the sake of illustration, the correction consists of an encoder/decoder DNN whose parameters were fine-tuned based on a grid search. We note however that the **MCxM** framework is also compatible with other DNN architectures such as convnets.

The data used in this work are synthetic, generated by the PMSS modelling system. PMSS is constituted by the parallel versions of the SWIFT 3D diagnostic flow model and the SPRAY 3D Lagrangian particle dispersion model. The training/validation dataset (approx. 20,000 instances) is produced in the French city of Grenoble under different weather and emission conditions. Once the learning phase is completed, the prediction error of the **MCx3M** was averaged over 5000 instances of hypothetical accidental release events in a district of Paris, whose urban geometry was never encountered during the training. The learned model exhibits a good accuracy and virtually instantaneous prediction time (0.75 ms) compared to usual CFD simulators.

The actual **MCxM** forecasts a 2-D horizontal concentration field at the height of the pollution source. In future works, we plan to fully utilize the 3-D simulations of PMSS to teach the proposed learning framework how to jointly predict the 3-D horizontal and vertical distributions of the pollutant. Also, we plan to extend the actual framework to consider rigorous mass conservation, more complex weather and atmospheric stability conditions, and a larger variety of urban, natural and mixed terrains.

Software and/or data availability section

The findings of this study are obtained using a research software developed by the authors. It is written in Python 3.7 and uses Tensorfow 2.2 library to implement the proposed deep-learning-based surrogate model. The codebase is about 1000 lines covering data processing and the **MCxM** framework. The learning phase is operated on a 32 GBytes NVIDIA Quadro M4000 GPU. The trained model is deployed and runs on a Latitude LE5490 laptop, Intel Core i5, 16 Gbytes RAM. The software is a property of the French Alternative Energies and Atomic Energy Commission (CEA) and is not currently shared publicly.

Raw data for model training, validation and test consist of *ASCII* files of total size 60 GBytes. Each file encodes the integrated pollutant concentration field as a matrix whose vertical and horizontal axes correspond to the geographic coordinates (more details in section 3). They are

generated by PMSS, a 3D multi-scale flow and dispersion modelling system dedicated to the complex atmospheric environment. It is a commercial software developed for fifteen years by the French Alternative Energies and Atomic Energy Commission and by ARIA Technologies. All data supporting the findings of this study are available from the corresponding author on request.

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.

Acknowledgment

The authors would like to acknowledge the financial support of the Cross-Disciplinary Program on Numerical Simulation of CEA, the French Alternative Energies and Atomic Energy Commission.

References

Agarap, A.F., 2018. Deep Learning Using Rectified Linear Units (Relu) arXiv preprint arXiv:1803.08375.
Aguilera, P.A., Fernández, A., Fernández, R., Rumí, R., Salmerón, A., 2011. Bayesian networks in environmental modelling. Environ. Model. Software 26 (12), 1376–1388.
Araujo, L.N., Belotti, J.T., Alves, T.A., de Souza Tadano, Y., Siqueira, H., 2020. Ensemble method based on artificial neural networks to estimate air pollution health risks. Environ. Model. Software 123, 104567.
Armand, P., Oldrini, O., Duchenne, C., Perdriel, S., 2021. Topical 3d modelling and simulation of air dispersion hazards as a new paradigm to support emergency preparedness and response. Environ. Model. Software 143, 105129.
Audebert, N., Le Saux, B., Lefèvrey, S., 2017. Fusion of heterogeneous data in convolutional networks for urban semantic labeling. In: 2017 Joint Urban Remote Sensing Event. JURSE), IEEE, pp. 1–4.
Ayturan, Y.A., Ayturan, Z.C., Altun, H.O., 2018. Air pollution modelling with deep learning: a review. Int. J. Environ. Pollut. Environ. Model. 1 (3), 58–62.
Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M., 2018. Automatic differentiation in machine learning: a survey. J. Mach. Learn. Res. 18.
Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A., 2013. A review of feature selection methods on synthetic data. Knowl. Inf. Syst. 34 (3), 483–519.
Cabaneros, S.M., Calautit, J.K., Hughes, B.R., 2019. A review of artificial neural network models for ambient air pollution prediction. Environ. Model. Software 119, 285–304.
Castelli, S.T., Armand, P., Tinarelli, G., Duchenne, C., Nibart, M., 2018. Validation of a Lagrangian particle dispersion model with wind tunnel and field experiments in urban environment. Atmos. Environ. 193, 273–289.
Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, pp. 249–256.
Hanna, S.R., Briggs, G.A., Hosker Jr., R.P., 1982. Handbook on Atmospheric Diffusion, Tech. Rep. National Oceanic and Atmospheric Administration, Oak Ridge, TN (USA).
He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034.
Hornik, K., Stinchcombe, M., White, H., et al., 1989. Multilayer feedforward networks are universal approximators. Neural Network. 2 (5), 359–366.
Hunter, L., Johnson, G., Watson, I., 1992. An investigation of three-dimensional characteristics of flow regimes within the urban canyon, Atmospheric Environment. Part B. Urban Atmosphere 26 (4), 425–432.
Ioffe, S., Szegedy, C., 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. PMLR, pp. 448–456.
Lauret, Pierre, Heymes, Frederic, Laurent, Aprin, Johannet, Anne, Gilles Dusserre, Lapebie, Emmanuel, Antoine, Osmont, 2014. Atmospheric turbulent dispersion modeling methods using machine learning tools. Chem. Eng. Transact. 36, 517–522.
Lauret, P., Heymes, F., Aprin, L., Johannet, A., 2016. Atmospheric dispersion modeling using Artificial Neural Network based cellular automata. Environ. Model. Software 85, 56–69.
Lindell, M.K., 1995. Assessing emergency preparedness in support of hazardous facility risk analyses: application to siting a US hazardous waste incinerator. J. Hazard Mater. 40 (3), 297–319.
Ni, J., Yang, H., Yao, J., Li, Z., Qin, P., 2020. Toxic gas dispersion prediction for point source emission using deep learning method, Human and Ecological Risk Assessment. Int. J. 26 (2), 557–570.
Oldrini, O., Armand, P., Duchenne, C., Olry, C., Moussafir, J., Tinarelli, G., 2017. Description and preliminary validation of the pmss fast response parallel atmospheric flow and dispersion solver in complex built-up areas. Environ. Fluid Mech. 17 (5), 997–1014.
Oldrini, O., Armand, P., Duchenne, C., Perdriel, S., 2019. Parallelization performances of pmss flow and dispersion modeling system over a huge urban area. Atmosphere 10 (7), 404.
Raissi, M., Yazdani, A., Karniadakis, G.E., Aug. 2018. Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data arXiv:1808.04327 [physics, stat].
Sorensen, J., 2004. Planning for protective action decision making: evacuate or shelter-in-place. J. Hazard Mater. 109 (1–3), 1–11.
Tatarchenko, M., Dosovitskiy, A., Brox, T., 2015. Single-View to Multi-View: Reconstructing Unseen Views With a Convolutional Network. CoRR abs/1511.06702, 1, p. 2, 2.
Tieleman, T., Hinton, G., et al., 2012. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA: Neural network. mach. learn. 4 (2), 26–31.
United States Environmental Protection Agency. National Ambient Air Quality Standards (NAAQS) table. https://www.epa.gov/criteria-air-pollutants/naaqs-table.
Vapnik, V.N., 1999. An overview of statistical learning theory. IEEE Trans. Neural Network. 10 (5), 988–999.
Vendel, F., 2011. Modélisation de la dispersion atmosphérique en présence d'obstacles complexes: application à l'étude de sites industriels. Ph.D. thesis.
Wang, R., Chen, B., Qiu, S., Zhu, Z., Wang, Y., Wang, Y., Qiu, X., 2018. Comparison of machine learning models for hazardous gas dispersion prediction in field cases. Int. J. Environ. Res. Publ. Health 15 (7), 1450.
Zannetti, P., 1990. Air Pollution Modeling: Theories, Computational Methods and Available Software. Computational Mechanics Publications.
Zhang, L., Xie, Y., Xidao, L., Zhang, X., 2018. Multi-source heterogeneous data fusion. In: 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD). IEEE, pp. 47–51.
Zhang, B., Zhang, H., Zhao, G., Lian, J., 2020. Constructing a pm2. 5 concentration prediction model by combining auto-encoder with bi-lstm neural networks. Environ. Model. Software 124, 104600.