Contents lists available at ScienceDirect

Building and Environment

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

A multi-fidelity wind surface pressure assessment via machine learning: A high-rise building case

Anina Šarkić Glumac [a,*], Onkar Jadhav [a], Vladimir Despotović [b], Bert Blocken [c,d], Stephane P.A. Bordas [e]

[a] University of Luxembourg, Interdisciplinary Centre for Security, Reliability and Trust (SnT), 6 Avenue de la Fonte, Esch-sur-Alzette, 4364, Luxembourg
[b] University of Luxembourg, 6 Avenue de la Fonte, Esch-sur-Alzette, 4364, Luxembourg
[c] Anemos BV, Spechtendreef 3, 2460 Lichtaart, Belgium
[d] Building Physics and Sustainable Design, Department of Civil Engineering, Leuven University, Kasteelpark Arenberg 40 – bus 2447, Leuven, 3001, Belgium
[e] University of Luxembourg, Department of Engineering, Institute of Computational Engineering, 6 Avenue de la Fonte, Esch-sur-Alzette, 4364, Luxembourg

ARTICLE INFO

Keywords:
Wind loading
Machine learning
LES
RANS

ABSTRACT

Computational fluid dynamics (CFD) represents an attractive tool for estimating wind pressures and wind loads on high-rise buildings. The CFD analyses can be conducted either by low-fidelity simulations (RANS) or by high-fidelity ones (LES). The low-fidelity model can efficiently estimate wind pressures over a large range of wind directions, but it generally lacks accuracy. On the other hand, the high-fidelity model generally exhibits satisfactory accuracy, yet, the high computational cost can limit the number of approaching wind angles that can be considered. In order to take advantage of the main benefits of these two CFD approaches, a multi-fidelity machine learning framework is investigated that aims to ensure the simulation accuracy while maintaining the computational efficiency. The study shows that the accurate prediction of distributions of mean and rms pressure over a high-rise building for the entire wind rose can be obtained by utilizing only 3 LES-related wind directions. The artificial neural network is shown to perform best among considered machine learning models. Moreover, hyperparameter optimization significantly improves the model predictions, increasing the R^2 value in the case of rms pressure by 60%. Dominant and ineffective features are determined that provide a route to solve a similar application more effectively.

1. Introduction

The accurate evaluation of wind pressures and wind loads on high-rise buildings represents a key point in their design process [1–4]. With the development of construction technology and engineering materials, the new generation of high-rise buildings is tending towards taller and more slender structures. This becomes a challenge as these structures are super-flexible, light-weight, have low damping ratios and low fundamental frequencies, making them very prone to wind loads [5–9]. Due to this, it is necessary to understand the detailed characteristics of wind effects on such structures.

The traditional approach for the assessment of wind loads on high-rise buildings strongly relies on atmospheric boundary layer wind tunnel practice [2,10]. However, in the last decades, there has been a burgeoning growth in the use of Computational Fluid Dynamics (CFD). This is due to the significant growth of available computational power and the fact that numerical methods have several advantages compared to wind tunnel tests. Important advantages include no dynamic similarity constraints and the provision of whole-flow field data, i.e. data in every point of the computational domain, as opposed to the limited number of sensor points in the wind tunnel (e.g., [10–13]). In addition, the costs of these CFD analyses are generally lower compared to experiments. However, routine use of CFD for design purposes still requires significant progress to capture the right balance between the accuracy of the results and the efficiency (e.g., [14,15]).

The major challenge in CFD comes from the difficulty of capturing the complex separated flows around structures that are composed of multi-scale fluctuations in the inflow turbulence and their nonlinear interactions [16]. Currently, the use of CFD for the wind flow around structures at high Reynolds numbers entails the decision of whether to apply low-fidelity simulations, such as Reynolds-averaged Navier–Stokes (RANS), or high-fidelity simulations such as Large Eddy Simulation (LES). RANS, considered as the workhorse of CFD (e.g., [12,

* Corresponding author.
E-mail addresses: anina.glumac@uni.lu (A. Sarkić Glumac), onkar.jadhav@uni.lu (O. Jadhav), vladimir.despotovic@lih.lu (V. Despotović), bert.blocken@kuleuven.be (B. Blocken), stephane.bordas@alum.northwestern.edu (S.P.A. Bordas).

https://doi.org/10.1016/j.buildenv.2023.110135
Received 28 November 2022; Received in revised form 27 January 2023; Accepted 19 February 2023
Available online 26 February 2023

16]), requires lower computational costs. However, the RANS-based simulation of wind loads is generally considered to be insufficiently accurate [10,14,15,17], because the turbulence is not resolved but modeled and as a result, flow separation, reattachment and vortex shedding in the wake of the building are often poorly reproduced, or not reproduced at all. To overcome this challenge, in LES, the Navier–Stokes equations are resolved over large turbulence scales only, while the effects of the smaller scales are modeled. This provides the ability to more accurately predict flow separation, reattachment and vortex shedding in the wake of the building, as well as the resulting mean pressures and pressure fluctuations [1,11,15,18–20]. However, this comes with a significant increase in computational cost. Thus, recently, multi-fidelity approaches have gotten more attention in trying to bridge this gap and establish synergies between the low-fidelity (RANS) and high-fidelity (LES) approaches.

Multi-fidelity models based on surrogate modeling, a subgroup of machine learning, have been successfully applied in several studies. Some examples are the optimization of a transonic aircraft wing with two levels of CFD fidelity [21]. Surrogate modeling was also applied in [22] for the rotor blade design, where the simplified code related to aerodynamics was coupled with high-fidelity numerical simulations. In the case of building-geometry optimization, Ding and Kareem [23] built the co-kriging model [21] using correlated CFD inputs with two model fidelities (RANS and LES) to minimize the competing aerodynamic objectives represented by the mean drag force coefficient and the standard deviation of the lift force coefficient.

Also, recently, other machine learning techniques have gotten more attention. In the light of specific machine learning applications in predicting pressure coefficients, the vast majority of work is related to the prediction of the pressures based on wind tunnel test data. The main motivation is grounded in the occasionally limited amount of data that can be gathered from wind tunnel tests. The main identified causes are the limited number of wind tunnel tests as being high in cost, executed with a limited number of sensors, inaccessibility of certain areas to install sensors, etc. Thus, machine learning applications were set to complement those wind tunnel tests by extending the resolution of the pressure measurement points over the facades of the buildings (e.g., [6,24]), extending the measurement set by including additional external conditions, such as different wind directions and terrain (e.g., [25–27]) or different interference scenarios [28], etc. The main quantities of interest were the mean pressure coefficient, the root mean square pressure coefficient (e.g., [6,24–26,28,29]) and the peak pressure coefficient [26] as well as the pressure time history (e.g., [6,24,27]). It is noticed that machine learning is gradually becoming more widely used in wind engineering interpolation problems, yet, the extrapolation difficulties are also highlighted [6].

However, the number of pressure related machine learning applications based on numerical datasets is still very limited. In particular, multi-fidelity machine learning approaches are of interest as they can leverage between RANS and LES to predict wind pressures. Only recently, a study was published by Lamberti and Gorle [15] explored the use of a machine learning (ML) framework to relate a large number of low-fidelity RANS to a small number of high-fidelity LES simulations to improve the prediction of the root mean square (rms) pressure coefficient (C_p') over a high-rise building. The main focus was to determine the best performing training set for a specific wind direction from the wind rose range. For most wind directions, the choice of two neighboring wind directions as training data gave the best results.

The study in present paper further explores a similar idea of a multi-fidelity machine learning framework, but shifts the focus from one specific angle to the whole wind rose. Thus, the main aim is to determine the optimal/minimal training set that provides the best wind pressure prediction for the whole wind rose. Moreover, besides already considered rms values of pressure coefficient, mean pressure coefficients are also taken into account. Usually, improvement of the prediction of wind pressures is done by testing different machine learning methods as in (e.g., [15,24,28]). Yet, besides different methods, the present study sets out to explore new machine learning techniques that might improve model's performance, such as hyperparameter optimization and feature selection. Namely, feature selection techniques are applied over the large feature set assembled based on similar studies. Feature selection can detect the set of dominant as well as ineffective features. As a result, the most optimal features can be determined for improving the prediction of the mean and rms pressure coefficients that should be used when dealing with similar problems. Thus, the open question 'What data should be used?' raised in [30] is tackled in this study. Lastly, an attempt is made to improve the prediction related to the deficiency of machine learning methods with regard to extrapolation problems. Possible improvements in this regard are demonstrated using the transfer learning technique.

This paper is organized into 6 sections. Section 2 is dedicated to the introduction of the high-rise building geometry and the numerical RANS and LES set-up. It also includes the validation of the LES results with wind tunnel measurements. Section 3 covers a brief explanation of the used machine learning, hyperparameter optimization and feature selection algorithms. It concludes with the proposition of the adaptive training strategy used to determine the optimal training set for the whole wind rose. Section 4 provides the results by determining the best performing machine learning model and optimal training data set with optimal input features. Finally, the paper closes with the limitations of the study, presented in Section 5, that can be used as possible areas of future research and a summary of the conclusions in Section 6.

2. Computational fluid dynamics models

In this section, first, the high-rise building case and inflow conditions are introduced, followed by the setup of the RANS simulations and LES simulations, and finally, the validation of the LES simulations with the wind tunnel experiments.

2.1. High-rise building model and inflow conditions

The considered high-rise building case is a numerical representation of the wind tunnel tests [31,32]. The model has a square cross-section with edges $B = 133.33\,\mathrm{mm}$, and the height of the building is $H = 400\,\mathrm{mm}$, which represents a 120 m tall building in full-scale. The building has a flat roof. Fig. 1 shows the building with the Cartesian coordinate system and the wind direction convention used in this study.

The mean incident wind profile, i.e. that measured in the empty wind tunnel at the center of the turntable [33], matches that of a power law with exponent of 0.2, as shown in Fig. 2(a). This is representative of the terrain category II [34] that can represent the approach flow in urban areas with a dominant high-rise building surrounded by sparse low-rise buildings. Such an arrangement is common at the outskirts of large cities, university campuses displaced from city centers, etc. Fig. 2(a,b) shows the incident vertical profiles of mean stream-wise wind speed (U), the stream-wise turbulence intensity (I_U) and the vertical turbulence intensity (I_W). 95% confidence intervals related to the uncertainty estimates of the experimental data are marked with vertical ticks from both marker sides. At building height, the values are 16 m/s, 13% and 11%, respectively. The wind tunnel measurements also included pressure measurements using 64 taps on the roof and 26 taps on the facades (marked in Fig. 1(b) and (c) with light gray circles).

2.2. RANS simulations

All RANS simulations are performed with the same domain and grid. The domain size is $L \times B \times H = 6.8 \times 6.8 \times 1.6\,\mathrm{m}$, where the height corresponds to the height of the wind tunnel. Moreover, the upstream length is $5.8H$, and the downstream length is $10.8H$, which is in agreement with best practice guidelines [36]. Similarly, as in [15]

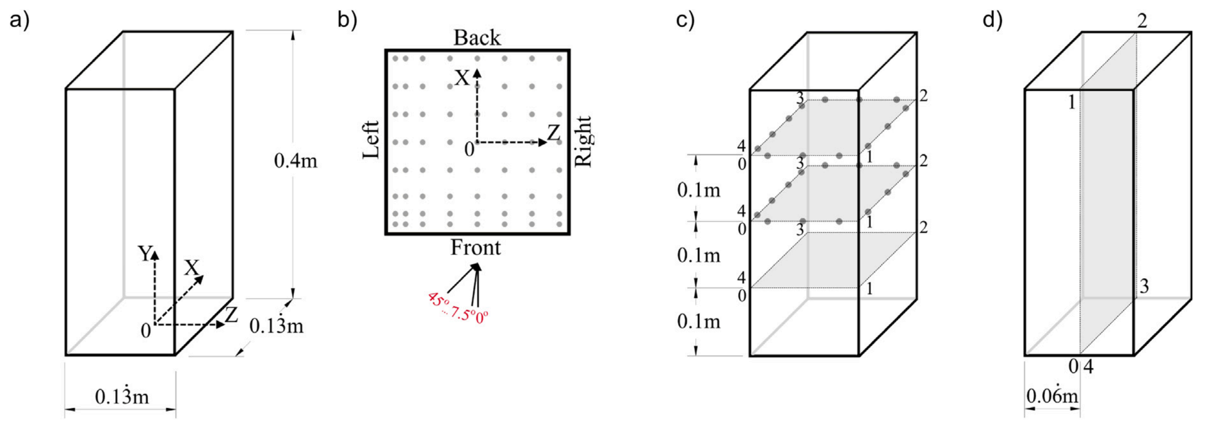

Fig. 1. Geometry of the high-rise building with (a) main dimensions and coordinate system; (b) top view with definition of wind directions; (c) position of horizontal and (d) vertical planes for data analysis. (Light gray circles represent the locations used for the assessment of the grid dependency).

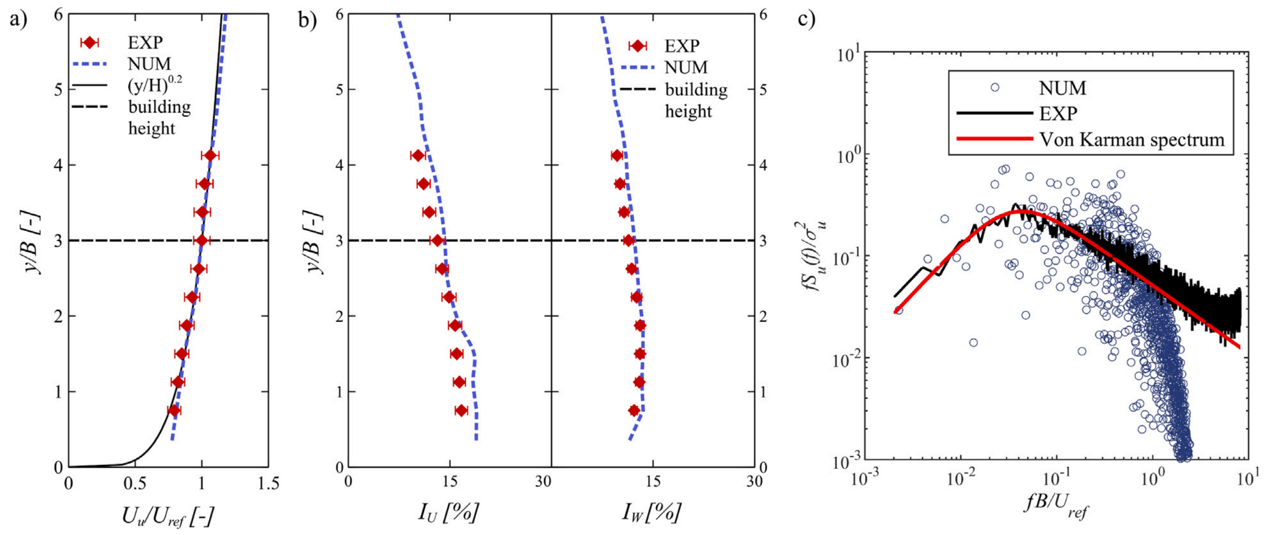

Fig. 2. (a,b) Incident vertical profiles: (a) mean streamwise velocity, (b) turbulence intensity in streamwise (I_U) and vertical (I_W) direction; (c) incident Von Karman turbulence spectrum at building height [35].

different wind directions are obtained by modifying velocity components imposed at the inflow. The grid is a structured grid containing 0.9 million hexahedral cells where refinement regions are applied near the high-rise building. This grid is referred to as the medium grid and it is as shown in Fig. 3. The number of cells along the streamwise, spanwise and vertical edges are 23, 23 and 82, respectively; the resulting y^+ is 141 on average. A grid dependency study is performed for the 0° angle. The results are compared against a coarser and a finer grid. The coarser and the finer grid are characterized by a spatial resolution that is 2 times lower, respectively higher near the building model than in the medium case. The mean pressure coefficient predicted by those three grids (coarse, medium and fine) are compared at the middle line of the roof in Fig. 1(c). Here discrepancies can be observed between the coarse grid on the one hand and the medium and fine grid on the other hand. In contrast, results of the medium and fine grid are showing a better agreement. Thus, the mean pressure coefficient of the medium and fine grid simulations are compared at 64 locations on the roof and 26 locations on the facades (light gray circles marked in Fig. 1(b)). As shown in Fig. 3(c), the comparison shows that 84.4% of the points on the building surface have a relative difference below 10% (compared to 34.7% when results of coarse and medium grid are compared). Thus, the simulations are performed with the medium grid.

The inlet boundary conditions of mean velocity U, turbulent kinetic energy k and specific turbulent dissipation rate ω are determined from the incident vertical wind tunnel profiles. The turbulent kinetic energy k is calculated from U and I_u using Eq. (1), where a is a parameter in the range between 0.5 and 1.5 [36–38]. In this study, $a = 1$ is chosen, as recommended by Tominaga et al. [36]. The specific turbulent dissipation rate ω is calculated based on the turbulence dissipation rate from Eqs. (2) and (3), with the von Karman constant $\kappa = 0.42$ and $\beta^* = 0.09$.

$$k(z) = a(I_u(z)u(z))^2 \tag{1}$$

$$\epsilon(z) = \frac{u^{*3}_{ABL}}{\kappa(z + z_0)} \tag{2}$$

$$\omega(z) = \frac{\epsilon(z)}{\beta^* k(z)} \tag{3}$$

Rough wall boundary conditions are taken into account with the standard wall function approach using the sand-grain roughness k_s, which is related to the aerodynamic roughness length z_0 by $k_s = 9.793 z_0/C_s$ [39,40]. In this study, for the ground surface values of $k_s = 0.00238$ m and $C_s = 8.2$ are used. The building walls have roughness $k_s = 0$ and $C_s = 0.5$. The outlet is treated as a pressure outlet with the constant relative pressure equal to zero and a zero-gradient boundary condition. Depending on the wind direction, the side boundaries are either inlet or outlet.

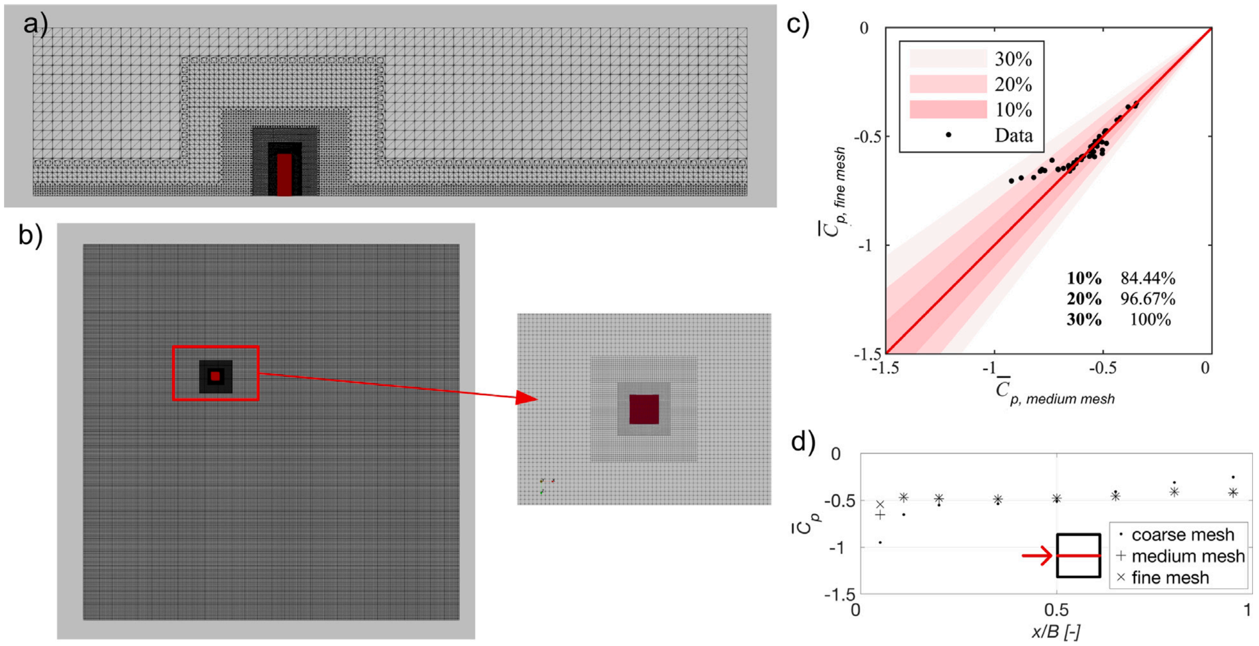

Fig. 3. (a) View of the RANS grid in the central vertical plane and; (b) a horizontal plane; (c) comparison of the mean pressure coefficient, medium against the fine grid at 64 locations on the roof and 26 locations marked with light gray dots in Fig. 1(b) and (c), the main diagonal represents a perfect fit; (d) comparison of the mean pressure coefficient for coarse, medium and fine grid at the light gray dots from Fig. 1(b) that are belonging to the middle line of the roof.

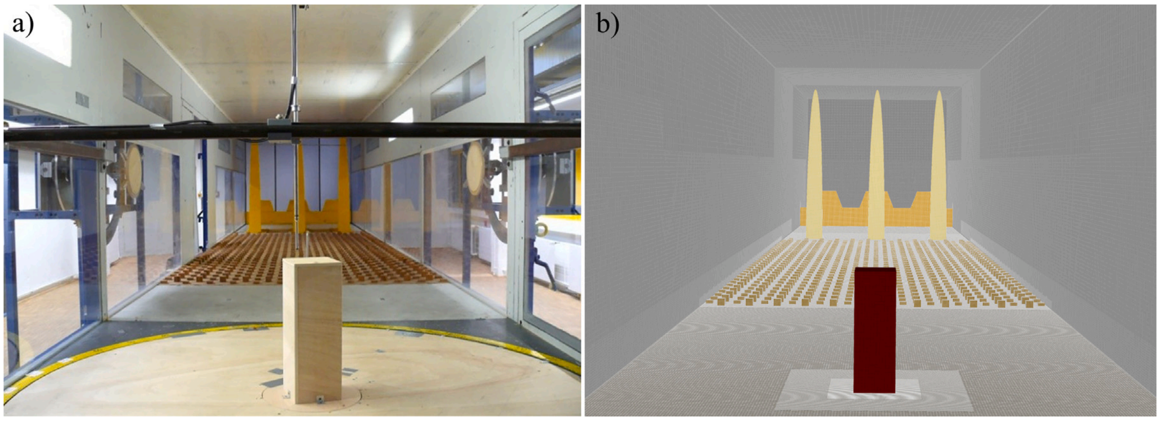

Fig. 4. Wind tunnel test section with the high-rise building model: (a) experimental and (b) computation domain [35].

The open-source code OpenFOAM (controle-volume based method) is used to perform the simulations. Different wind directions are simulated, keeping the same grid and modifying the inflow velocity components, where boundary conditions are assigned using the groovyBC library in OpenFOAM. The 3D steady RANS equations are solved in combination with the $k-\omega$–SST model [41]. The SIMPLE algorithm is used for pressure–velocity coupling, and the discretization is done using second-order numerical schemes.

2.3. LES simulations and validation

The computational domain for LES is a full numerical representation of the wind tunnel. The domain size is $L \times B \times H = 13.5 \times 1.8 \times 1.6$ m. The original geometry of the turbulence generators (castellated barrier, spires and roughness field) is replicated in the driver section (i.e., pre-cursor domain), as shown in Fig. 4.

Spatial discretization of the computational domain is performed using a predominantly hexahedral grid, see Fig. 5, where a view of the LES grid is presented. As it can be seen from Fig. 5, additional levels of refinement are applied around the obstacles. The base of the refinement strategy is cell splitting in predefined regions. The initial grid consists of purely cubic cells with edge $\delta/B = 2.6 \times 10^{-1}$ as a starting point for further refinement (coarse grid in Fig. 5(a)). The precursor domain grid is created following the guidelines in [42]. The finest zone, in the building nearby, has a cell size of $\delta/B = 1.67 \times 10^{-2}$. The resulting grid resolution is higher than the minimum required by Tominaga et al. [36]. Furthermore, at the model walls, a body-fitted structured grid of 15 boundary layers is adopted, with an expansion ratio of 1.05, as Murakami [43] suggested. The resulting mean dimensionless wall distance y^+ is around 5 in all cases, while the maximum values are around 15. The final grid of the simulation has around 27 million cells. The percentage of hexahedrons in the grid is over 97%.

Atmospheric boundary layer flow is simulated similarly as in the wind tunnel, as computational domain explicitly considers turbulence generators, castellated barrier and roughness field, as shown in Fig. 4, Dirichlet conditions on the velocity field are specified at the inlet. 15 m/s is chosen as input value, as it produces a stream-wise wind speed of 14.3 m/s at building's height, similar to respective wind tunnel value of 16 m/s. The outlet is treated as a pressure-outlet with a constant relative pressure equal to zero and a zero-gradient boundary condition

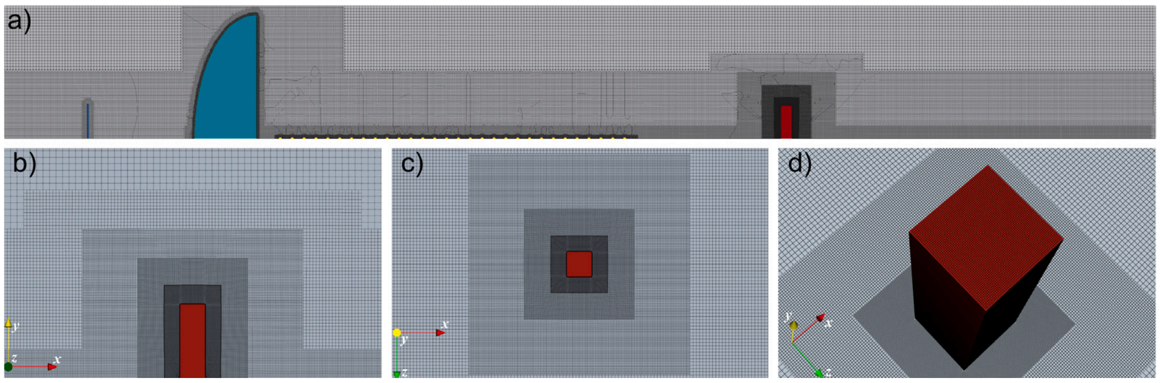

Fig. 5. (a) View of the LES grid through the vertical center plane; (b) close view of grid in vertical center plane; (c) top view of grid near building; (d) perspective view on grid on building surfaces and on ground plane near the building.

for the other flow variables. Other boundaries, floor, and sides of the tunnel, as well as the building surface, are modeled as a wall. To reduce computational time and speed up the convergence process, the initialization of the pressure and velocity fields is done by mapping the mean fields from coarser LES simulations.

All LES simulations are performed using the pisoFOAM solver of OpenFOAM, which uses the PISO velocity–pressure coupling scheme [44]. To mimic the interaction between subgrid-scale turbulence and resolved large-scale turbulent motion and overcome the closure problem, the Wall-Adapting Local Eddy-viscosity (WALE) SGS model by Nicoud and Ducros [45] is used. Time discretization is performed using an implicit, second-order backward scheme. The non-dimensional time step, based on the edge length B and velocity at the building height $U_{ref,NUM}$, is set to $\Delta t^* = \Delta t \cdot U_{ref,NUM}/B = 4.3\times10^{-3}$ leading to the maximum and mean Courant–Friedrichs–Lewy number in all simulations equal to 3 and 0.05, respectively. As far as the spatial discretization of the LES equations is concerned, discretization of the convection terms is performed by applying the second-order accurate linear upwind stabilized transport (LUST) scheme. All the other terms of the equations are discretized using a centered second-order differentiation scheme. All simulations are run for $1000t^* = 1000\cdot(t\cdot U_{ref,NUM}/B)$. Data has been sampled from $200t^*$, representing two flow cycles through the domain, as it is considered a minimum to develop the turbulent flow [46] fully. Time histories of the velocity and pressure are monitored to evaluate the convergence carefully in addition to monitoring equation residuals. More details on the LES setup can be found in the previous work [35].

The validation of the LES simulations is conducted using available experimental data [31]. First and second-order statistics of velocity and pressure data considering two wind directions 0° and 45° are validated and presented in [35]. Here only validation results related to the flow at the building location and surface pressures are summarized.

To check whether the incident flow is adequately modeled at the building location (i.e. at the center of the turntable of the empty wind tunnel), an additional LES simulation using the same domain and grid as presented in Fig. 4(b) is performed without the building model. Fig. 2 shows the resulting profiles of the mean streamwise velocity and the turbulence intensities from this numerical simulation, compared to profiles obtained in the experimental measurements at the same location. Graphs in Fig. 2 show a good agreement between data. In addition, the power spectrums at building height from numerical simulation and experiment fit well the Von Karman spectrum [47] shown in Fig. 2(c). In the high-frequency range, energy in the LES drops as expected due to the filtering. However, more than 80% of turbulence kinetic energy is resolved, which is a generally adopted threshold defined for a well-resolved LES [48].

Validation of the LES simulation results is performed by comparing with the experiments of the mean $\bar{C}_p$ and fluctuating C'_p values and presented in Table 1. The experimental data from all pressure taps marked in Fig. 1 are considered (light gray dots at the roof and

Table 1
Performance metrics in terms of percentages for the pressure coefficient for 0° and 45° degrees wind direction.

Performance	Flat roof building			
metrics	0°		45°	
[%]	$\bar{C}_p$	C'_p	$\bar{C}_p$	C'_p
10% tolerance	77.33	46.67	45.33	45.33
20% tolerance	97.33	90.67	84.00	60.00
30% tolerance	100.00	100.00	86.67	64.00

facades), and the percentage of data points with deviations less than 10%, 20%, and 30% are presented in Table 1. The simulated mean and rms pressure coefficients are, in general, in good agreement with the experimental results, that can be also observed in Fig. 6. As expected, Table 1 also confirms that second-order statistics are more difficult to predict, in particular in the case of wind direction of 45°. Similar values are reported in [2].

Fig. 7 shows the results of these validated LES simulations. Namely, contours of the mean and rms pressure coefficient over the roof and facades of the building are presented for angles 0°, 15°, 45°.

3. Machine learning methodology

Section 3.1 first discusses a wide range of features used to predict mean and rms pressure coefficients. This is followed by a short introduction of feature selection techniques used to determine the dominant ones that contribute the most to the prediction of the machine learning model in Section 3.2. Furthermore, Sections 3.3 and 3.4 briefly describe applied machine learning models as well as hyperparameter optimization techniques for improving the model performance. Finally, in Section 3.5 the training strategy is presented related to the choice of computationally costly LES simulations needed to provide accurate predictions of mean and rms pressure coefficients with respect to the entire wind rose.

3.1. Considered features

In machine learning, features are the independent variables that represent and transform input measurements in order to improve the predictive capability of the model. The features presented in [15] are used as a starting point. This includes the mean pressure coefficient $\bar{C}_{p,\mathrm{RANS}}$, the normalized local turbulence kinetic energy in the center of the wall-adjacent cell k/U_H^2, and the normalized height-dependent incident velocity magnitude U_{in}/U_H. These variables are imposed as important ones as they are used in the standard empirical models for rms pressure coefficient evaluation. One such model was

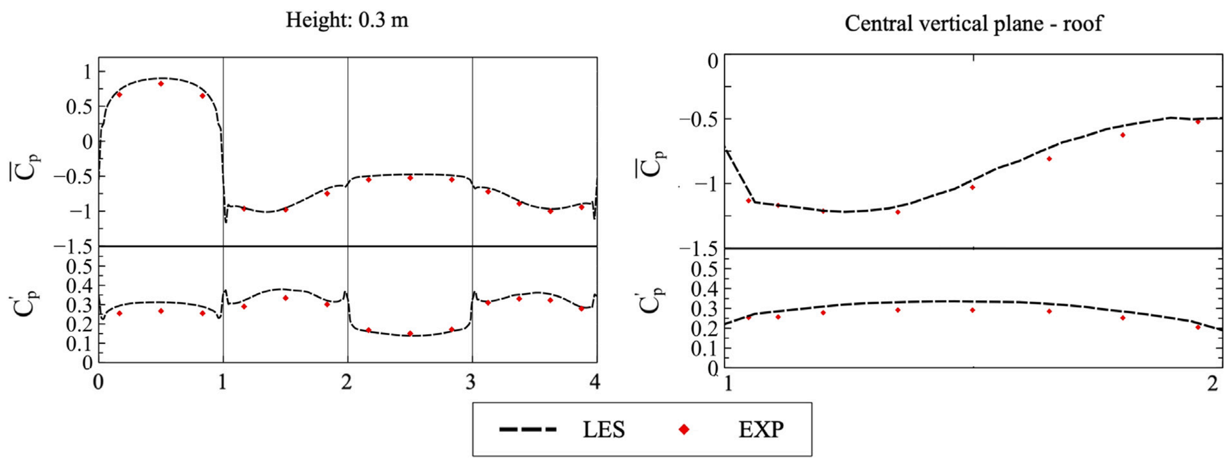

Fig. 6. Comparison between results from LES and wind tunnel measurements for the 0° wind direction. The left plot shows the $\bar{C}_p$ and C'_p results at 0.3 m height across the perimeter of the building (Fig. 1c) and the right plot along the middle roof line in the central vertical plane (Fig. 1d).

proposed by Peterson [49] that calculates the rms value of the pressure coefficient as

$$C'_{p,\text{RANS}} = \frac{k/3 + 0.816|C_p|U_{in}\sqrt{k_{in}}}{0.5U_H^2}, \quad (4)$$

where k_{in} is the incident turbulence kinetic energy, U_{in} mean velocity magnitude of the incident atmospheric boundary layer flow and U_H the reference wind velocity at building height. Note that $\bar{C}_{p,\text{RANS}}$ and $C'_{p,\text{RANS}}$ obtained using RANS are denoted with subscript RANS throughout the paper.

Mean and rms pressure coefficients generally show a different behavior in different flow regimes, i.e. separation, reattachment and fully attached flow, as observed in [35]. For example, Fig. 7 shows that the rms pressure coefficient is higher inside of flow separation. Thus, Lamberti and Gorle [15] additionally included features to provide further information on those flow regimes, such as: the friction coefficient $C_f = \|\tau_w\|/0.5\rho U_H^2$ and the non-dimensional norm of the pressure gradient $H\|\nabla P\|/0.5\rho U_H^2$. Here τ_w is the wall shear stress vector, H is the building height, and ρ is the air density.

Apart from these flow variables, other studies [6,26,50] considered as well spatial position on the building surface, which in this study corresponds to the coordinates of the faces. In addition, in [26], the wind direction was also used as an additional feature.

Thus, this leads to a total number of 9 features (inputs) listed below:

- Mean pressure coefficient $\bar{C}_{p,\text{RANS}}$: $\frac{P}{0.5\rho\ U_H^2}$
- Non-dimensional turbulence kinetic energy: $\frac{k}{U_H^2}$
- Non-dimensional inflow velocity magnitude: $\frac{U_{in}}{U_H}$
- Non-dimensional pressure gradient: $\frac{H\|\nabla P\|}{0.5\rho U_H^2}$
- Friction coefficient C_f: $\frac{\|\tau_w\|}{0.5\rho U_H^2}$
- X coordinate: x
- Y coordinate: y
- Z coordinate: z
- Wind direction: α.

These are used to predict two output parameters, the mean and the rms value of pressure coefficient, $\bar{C}_p$ and C'_p, respectively, based on the ML model trained with the training LES simulations.

3.2. Feature selection

Feature selection is the process of reducing the number of input variables in the model and represents an essential tool in machine learning, which helps to improve the model performance [51]. In addition, it deals with overfitting. Overfitting may occur if the model adapts too well to the training data, but does not generalize to any external set of data. Redundant or irrelevant features may be one of the causes of overfitting; therefore, eliminating ambiguous features from the training set ensures less noise during the training phase and improves the quality of the model. Furthermore, less data demands less storage space and eases the curse of dimensionality, i.e., eases the difficulty of optimizing a function with too many input variables. Thus, for a well-defined model, feature engineering can reduce the error drastically and additionally can achieve faster training since the inferior features are eliminated.

Thus, feature selection techniques are used to select only relevant features to reduce the computational cost of modeling and improve model performance. The aim is to identify the most dominant features and potentially exclude the least important ones that can be excluded without compromising the accuracy of the model. The end goal is to determine such features that must be included in the training dataset while solving similar problems, i.e. determination of the mean and rms pressure coefficient values on the surfaces of the building.

To facilitate this goal, two feature selection techniques are used: F-statistics [51], which considers the linear correlation between two variables, and mutual information [52], which considers both linear and nonlinear dependencies. The end goal is to rank the features by their impact on the prediction based on obtained results. This ranking is then used to determine the most dominant features.

- *F-statistics*
  The F-statistics method determines a linear correlation between features and the output variable and selects the best features based on univariate statistical tests [51]. The method computes the cross-correlation between each feature and the output and converts it into an F score. The F score based on the coefficient of determination is given as

$$F = \frac{R^2}{1 - R^2}\frac{n - p}{p - 1}, \quad (5)$$

  where R^2 is the coefficient of determination, n is the number of observations, and p is the number of features. The coefficient of determination (R^2) is the square of the coefficient of correlation, which provides information about the strength and direction of the relationship between the two variables. Thus, the method first computes the correlation coefficient and obtains the F score using Eq. (5). A high value of the F score indicates a better dependency between those two variables.
- *Mutual information*

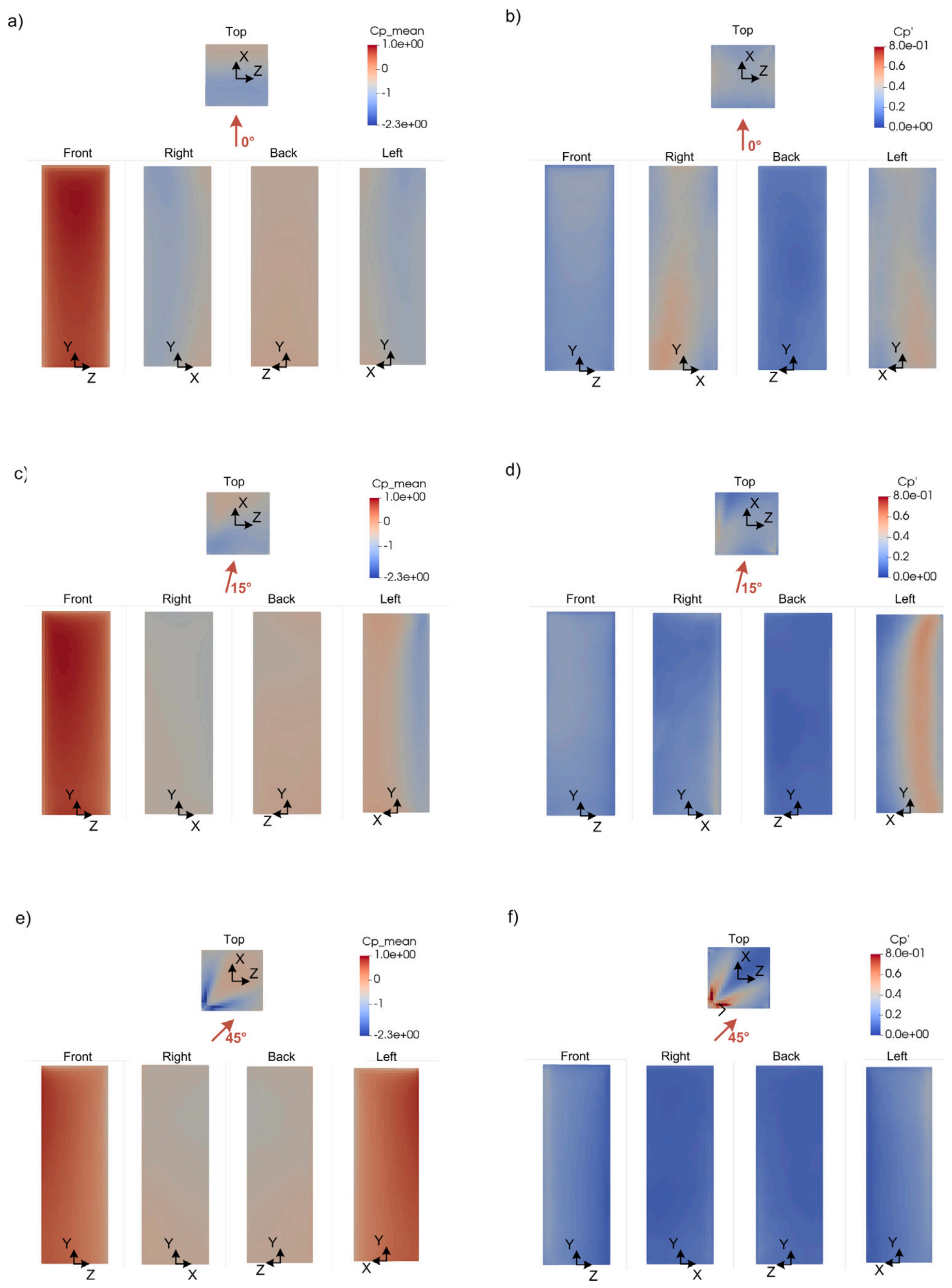

Fig. 7. Contours of mean pressure coefficient for angles: (a) 0°, (c) 15°, (e) 45°; and same for rms pressure coefficient for angles: (b) 0°, (d) 15°, (f) 45°.

Mutual information is a method that can capture any kind of dependency between variables [53], not only linear as in the F-statistics. The mutual information between two variables is given by Ross [52]:

$$I(x,y)=\sum_{i,j}P_{xy}(i,j)\log\frac{P_{xy}(i,j)}{P_x(i)P_y(j)}, \tag{6}$$

where P_x and P_y are the marginals of the variables X and Y, respectively and $P_{xy}(i,j)$ is the joint probability distribution. Mutual information between two variables is a non-negative value that measures the dependency between the variables. High values indicate high dependency. Intuitively, I measures how much knowing the value of one variable reduces the uncertainty of the other. For example, if x and y are two independent variables, knowing x does not give any information about y and vice versa, so their mutual information will be zero. Thus, the dominant feature can be determined by indicating a strong dependency on the target variable.

3.3. Machine learning models

The success of the algorithm is judged by reporting the root mean square error (RMSE) and the coefficient of determination (R^2). For the RMSE value to be used as a measure, it is also necessary to relate to the scale of the variable of interest, e.g., the mean pressure coefficient. On the other hand, R^2 measures how well the regression model approximates the given data, and $R^2 = 1$ indicates that the model fits the given data perfectly. A variety of machine learning algorithms are considered:

- *Support vector machine*
  A support vector machine (SVM) is a supervised learning algorithm initially designed for solving binary classification tasks but could also be extended for solving regression problems [54]. The objective of the support vector machine algorithm is to find a hyperplane in a high dimensional feature space that distinctly regresses the data points. The dimension of this hyperplane depends on the number of features, meaning that for two features, the hyperplane is a line, and for three features, it is a plane, etc. The problem is that one can define many possible equations of a line (hyperplanes) for the same data points. Thus, most regression models minimize the sum of squared errors in order to obtain the best hyperplane. In contrast, the objective function of SVM minimizes the L2-norm of the unknown coefficient vector and not the squared error [55], i.e.,

  $$\min \|w\|_2^2,$$

  where w represents the vector of coefficients of regression. The error term is handled by constraints, where it is less than or equal to a specified margin, called the maximum error as

  $$y_i - wx_i \leq \epsilon,$$
  $$wx_i - y_i \leq \epsilon,$$

  where ϵ is the maximum error, also called margin of error. The data points that are closest to the hyperplane are called support vectors, hence the name support vector machine, which influence the position and orientation of the hyperplane. These support vectors are then used to maximize the margin of error. Maximizing the margin distance provides some reinforcement and avoids overfitting. Additionally, the SVM uses a Kernel trick to work well with the nonlinear data [55]. The idea is to transform the training data into a high dimensional space where it becomes linear and then use a simple linear SVM. The most common choices of the kernel functions are linear, polynomial, radial basis function, and sigmoid.
- *Random forest*
  In machine learning, instead of using only one model, an ensemble of models can be used to improve the model's performance. Ensemble methods combine several models into a single machine learning model to increase the performance and robustness [56]. Random forest is a type of model that combines decision trees [57,58]. A decision tree is a tree structure model that breaks down a dataset into smaller and smaller subsets having similar values. A new branch of the tree is created by splitting the data into two subsets. One of the choices to split the data into two groups is variance [59]. Initially, the algorithm generates various splits, and the variance of each subset is computed. It is known that the dataset having similar values has low variance. Thus, to combine similar values, the algorithm selects the split that gives the smallest variance. This split process is repeated until the last node of the tree is pure, i.e., either has all homogeneous values or only one value. For prediction, the test data are fed to the trained tree and sorted into the branches having similar values until the last nodes. The output is then the average of the last nodes in which the test data is sorted. To avoid overfitting and increase the performance, the random forest combines several decision trees and takes the average of their outputs.
- *Gradient boosting*
  The gradient boosting algorithm is another ensemble algorithm considered in this study [60]. It is used for reducing bias and variance in supervised learning that combines several models (weak learners) into a single robust model in an iterative fashion [61]. The algorithm starts with a simple model (first weak learner) and uses it to predict some output. As a next step, in order to improve this output, the algorithm fits a new model (next weak learner) to the residual errors made by the previous model. This process is then repeated for certain iterations until the residual is minimized. Adding a new model to improve the previous model iteratively (boosting stages) is called boosting. In gradient boosting, these models are the decision trees with only a few branches. On the other hand, the random forest constructs full-grown decision trees. Also, the random forest algorithm builds trees in parallel and thus faster than the gradient boosting, which builds trees sequentially. However, gradient boosting sometimes gives better results than the random forest. The reason being each tree in the gradient boosting is grown using information from previously grown trees, which minimizes the overall error.
- *Neural Network*
  An artificial neural network (ANN) has also been used for regression in this work. It is comprised of a large number of connected nodes called neurons, each of which performs a simple mathematical operation [62,63]. They are organized in several layers, including one input layer, one output layer and one or more hidden layers. For example, it has features described in Section 3.1 as input and output as mean and/or rms pressure coefficient in our case. Neurons within two successive hidden layers are cross-connected with varying weights and biases. The computation results from the previous neurons are sent to the neurons of the next layer in a feedforward manner. This process from the input to the output layer is called the feedforward procedure, which can be explained using a single neuron readily. As shown in Fig. 8(a), the input values x_i are feedforwarded into a neuron through some connections. Each of these connections is characterized by some weight w_i. The output of the neuron is then computed directly from the sum of the product of connection weights plus some bias b as

  $$b_i + \sum_i^n x_i w_i, \qquad i = 1, \ldots, n.$$

  An undesirable property of this formula is that the output prediction is a linear function. Thus, to get nonlinear characteristics, some nonlinear function f, also known as the activation function is applied to this weighted sum as

  $$z_i = f(b_i + \sum_i^n x_i w_i), \qquad i = 1, \ldots, n.$$

  This nonlinear function assures the nonlinear output of the neurons. There are different choices for the activation function, namely the sigmoid function $f(x) = (1 + e^{-x})^{-1}$, the hyperbolic tangent function $f(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}}$, or the rectified linear unit (ReLu) $f(x) = \max(0, x)$ [64]. The selection of the appropriate activation function is later discussed in Section 3.4.
  Each layer of the neural network contains such single neurons, and the feedforward neural network is the combination of such layers, as shown in Fig. 8(b). Thus, the output of a layer is the combined output of each neuron within that layer. In order to improve this output, the neural network has to adjust the weights and biases of these layers. This adjustment is made by finding the weights and biases that minimize the error between truth and the output iteratively during the training process. A loss function can be defined to measure this difference between the predicted and the actual output. The most commonly used loss function is the mean square error, defined for N input–output training pairs (x_i, y_i) is

  $$E = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2,$$

  where $\hat{y}_i$ denotes the predicted output. The loss function is usually minimized using a gradient descent algorithm, which adjusts the weights and biases in the direction of the negative gradient of the loss function. The new weights and biases are determined

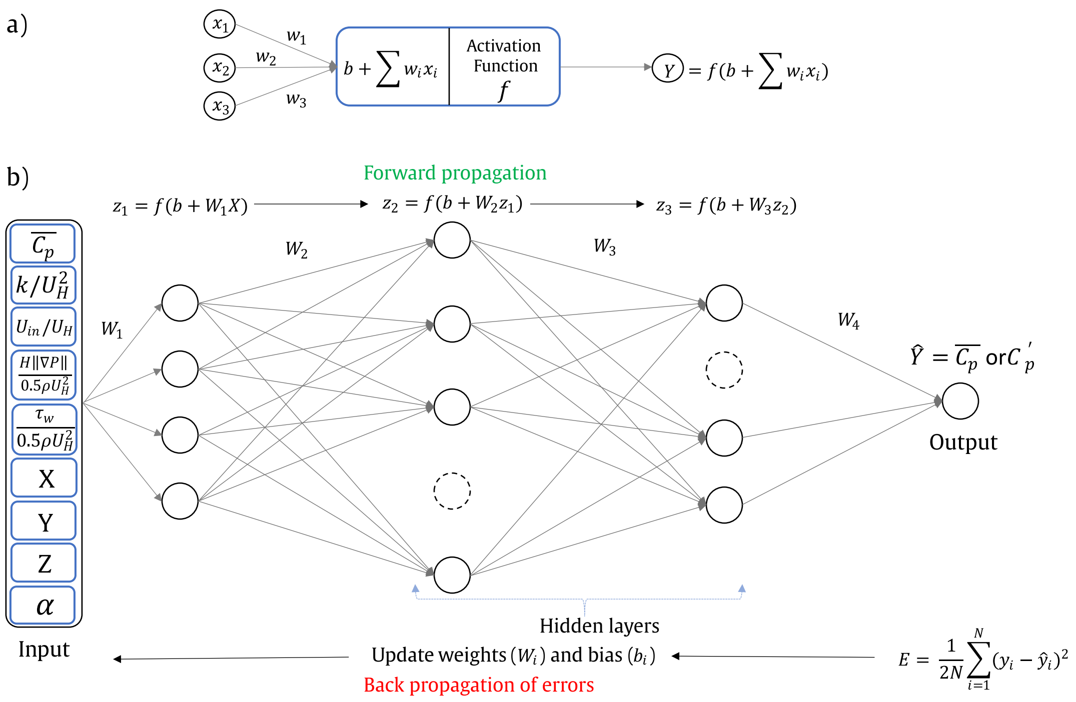

Fig. 8. (a) Single-layer perceptron; (b) feedforward neural network. The dotted neurons represents dropout.

iteratively until convergence (back propagation of errors). There are different modifications of gradient descent algorithms called optimizers, e.g., stochastic gradient descent, RMSProp [65] or Adam [66]. See [63] for more details. In short, training can be defined as the process of finding the optimal weights/bias, which is achieved by minimizing the error between ANN output and truth [26,67]. When attempting to train the neural network using these optimization techniques, overfitting might be a problem. The dropout technique proposed in [68] avoids this overfitting problem in neural networks. The basic idea is to randomly drop the neurons along with their connections from the neural network during training, as shown in Fig. 8 with the dotted circles. This process repeats itself for every epoch, i.e., iteration, sampling only a thinned neural network at each iteration. Thus, the dropout technique avoids the co-adaptation making the model less prone to overfitting.

3.4. Hyperparameter optimization

The above machine learning algorithms aim to train a model that minimizes a loss function over a given data. In addition to it, these models are governed by some underlying parameters called hyperparameters presented in Table 3. For example, in the case of neural networks, these hyperparameters are the number of hidden layers, activation functions, the number of neurons, or the algorithm used for reducing the loss function called optimizer. The problem is that it is not a priori known how many layers are the most suitable, how many neurons are the best, or which optimizer will give the best results. Hence, the choice and the combination of these parameters significantly affect the model's performance. Consequently, it is necessary to define an efficient method to determine the optimal combinations of these hyperparameters that will provide the best results. This problem of determining optimal hyperparameters can be formulated as an optimization problem, which in this study is solved using the random search technique [69,70]. See [71,72] for the implementation and applications of random search hyperparameter optimization.

Also, an adaptive learning rate is considered. When learning stagnates, and a metric has stopped improving, a model often benefits from reducing the learning rate by some factor. Thus, if no improvement is noticed for a few epochs, the learning rate is reduced. Finally, the hyperparameter optimization for the ML models discussed above is performed, and the considered hyperparameters for each model are presented in Table 3.

3.5. Full dataset and adaptive training strategy

The full dataset consists of the RANS and LES simulations for 7 different wind directions: 0°, 7.5°, 15°, 22.5°, 30°, 37.5°, 45°. Given the symmetry of the building, these wind directions cover the entire wind rose. The considered RANS grid has 7153 faces on the building surface. Thus, for each wind direction, there is a set of 7153 data points for each considered feature, i.e. in the general case, nine features defined in Section 3.1 and two outputs, $\bar{C}_p$ and C'_p.

Note that this number of faces on the building surface is a consequence of the adopted RANS grid, yet it strongly influences the machine learning output as it defines the size of the dataset. Thus, this number should be large enough to provide enough data for the machine learning predictions. Therefore, the optimization of the size of the grid in terms of its influence on the machine learning output could be of interest in future studies.

The main aim is to optimize the machine learning framework not just in terms of the model but as well as training datasets that can capture the entire wind rose, i.e. the whole range of wind directions. This leads to optimizing, i.e. reducing the number of computationally costly LES simulations that the machine learning algorithm uses to improve the prediction of $\bar{C}_p$ and C'_p coefficients. For that purpose, this paper presents an adaptive training strategy that gradually increases the number of selected training datasets, as:

- Baseline model: that exploits as training data the LES datasets of only two wind directions, i.e., 0° and 45°. This means that for training, the ML model takes features (RANS-based data) as inputs and the corresponding $\bar{C}_p$ or C'_p LES values as outputs for those two wind directions. As the goal is the adequate prediction for all wind directions, it is reasonable to consider a combination of those two extreme wind directions as the first training set. Those

Table 2
Test RMSE and R^2 values of the selected models (with or without hyperparameter optimization) trained on $\{0°, 45°\}$ wind direction datasets and tested on the 15° wind direction dataset.

Model	$\bar{C}_p$				C'_p			
	Non-optimized model		Optimized model		Non-optimized model		Optimized model	
Support vector machine	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE
5 flow features	0.9293	0.1464	0.9293	0.1464	−0.2264	0.1176	−0.2264	0.1176
5 flow features + Angle	0.9352	0.1431	0.9352	0.1431	0.2589	0.0914	0.2589	0.0914
5 flow features + Coords	0.9253	0.1512	0.9253	0.1512	0.1805	0.0961	0.1805	0.0961
5 flow features + Coords + Angle	0.9511	0.1224	0.9511	0.1224	0.4006	0.0822	0.4006	0.0822
Random forest	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE
5 flow features	0.9295	0.1421	0.9381	0.1377	−0.2123	0.1169	−0.1887	0.1158
5 flow features + Angle	0.9368	0.1399	0.9445	0.1304	0.2830	0.0908	0.2959	0.0891
5 flow features + Coords	0.9301	0.1502	0.9416	0.1337	0.1846	0.0936	0.2097	0.0944
5 flow features + Coords + Angle	0.9532	0.1200	0.9581	0.1131	0.3840	0.0882	0.4223	0.0807
Gradient boosting	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE
5 flow features	0.9321	0.1408	0.9501	0.1237	−0.1188	0.1123	−0.0855	0.1107
5 flow features + Angle	0.9415	0.1313	0.9585	0.1127	0.3131	0.0880	0.3738	0.0841
5 flow features + Coords	0.9403	0.1380	0.9512	0.1217	0.2908	0.1084	0.3573	0.0851
5 flow features + Coords + Angle	0.9524	0.1199	0.9613	0.1088	0.3936	0.0827	0.4403	0.0794
Artificial neural network	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE
5 flow features	0.9469	0.1297	0.9515	0.1201	−0.1012	0.1078	0.0702	0.1024
5 flow features + Angle	0.9588	0.111	0.9668	0.1008	0.3994	0.0823	0.5046	0.0759
5 flow features + Coords	0.9606	0.1098	0.9633	0.1051	0.3524	0.0978	0.4322	0.0800
5 flow features + Coords + Angle	0.9656	0.1002	0.9705	0.0951	0.4358	0.0803	0.5745	0.0693

incident wind directions are extreme in terms of flow patterns, as 0° wind direction creates a large separation bubble on top of the building and symmetrical separations on the sides, whereas the 45° angle produces two symmetrical conical separations on top of the building and separation of the flow behind the leeward sides of the building. More details, including the visualizations of those flow structures, over the same high-rise building, are presented in [35]. More importantly, those flow features are affecting the pressure distribution in a very different way, as shown in Fig. 7. The pressure distribution pattern created due to the wind direction of 15°, as well as the others, can be seen as "in between" above mentioned extreme wind directions, thus, making 0° and 45° incident angles good candidates for the initial training set.

- Universal model: In order to improve the accuracy of the machine learning model, a need for an additional LES dataset at some intermediate wind direction may arise. Consequently, an additional wind direction dataset is added to the initial training dataset by including the relevant features as inputs and LES outputs for that specific wind direction. This process is controlled by monitoring R^2 and RMSE values. R^2 and RMSE values are estimated for each wind direction by comparing the ML model results against LES data on the test wind directions (the training wind directions are excluded). If necessary, this procedure can be repeated until a satisfactory prediction is obtained. To achieve this, an important criterion is followed. Namely, the machine learning model should provide an R^2 value of at least 0.8 for each tested wind direction, as the regression models with $R^2 \geq 0.8$ are considered satisfactory [73]. Section 4.2 will further explore the inclusion of the additional training set and, finally, the formation of the optimal training set.

4. Results

4.1. Initial ML model, preliminary feature selection and hyper-parameter optimization

Based on the initial training dataset $\{0°, 45°\}$, all machine learning models are tested for all considered wind directions, and the worse results are obtained for the angle of attack of 15°. Thus, Table 2 reports R^2 and RMSE values for each machine learning model for the 15° case. It is noticed that the artificial neural network performs best among all considered approaches. Also, the gradient boosting regression gives slightly better results than the random forest. While in comparison, the random forest regression outperforms support vector regression.

Fig. 9 illustrates these results more clearly. The three plots show the $\bar{C}_p$ and C'_p results at three different heights along the perimeter of the building and one plot in the central vertical plane, as illustrated in Fig. 1(c) and (d). It can be observed that the artificial neural network follows the trend of $\bar{C}_p$ and C'_p values comparatively better than the other models. In particular, this is related to the second and third horizontal slices, i.e., at heights 0.2 m and 0.3 m, where all other models fail to predict C'_p compared to the ANN. A similar phenomenon is clearly observed in the vertical center plane plot, where the ANN mimics the LES results quite well. Note that the difference between RMSEs in Table 2 for $\bar{C}_p$ and C'_p is due to the fact that RMSE is a scale-dependent metric, and $\bar{C}_p$ and C'_p have different scales, as demonstrated in Fig. 7.

Regardless of the model, all developed models perform comparatively better for $\bar{C}_p$ than C'_p, as can be seen from Table 2 and Fig. 9. The reason can be related to the fact that $\bar{C}_{p,\text{RANS}}$ obtained using RANS is used as a feature, that is not the case for C'_p.

Fig. 9 shows another interesting phenomenon, where the C'_p results in zone 1–2 are overpredicted. The reason is the difference between the range of the 0° and 45° training dataset C'_p values. For example, Fig. 7(b) and (f) indicate that the C'_p values for the 0° and 45° wind directions in zone 1–2 range from 0.123 to 0.458 and 0.089 to 0.286, respectively. Since the machine learning models interpolate between these values to predict the results for the 15° test case, they get influenced by these values. Therefore, the machine learning models deliver C'_p results in the range of 0.125 to 0.398 for the 15° test case, like a weighted average of 0° and 45° C'_p results. Here the weighted average means that the machine learning algorithms put more weight on the 0° as the 15° is closer to the 0° than the 45° during testing. Therefore, the predicted values are more influenced by high 0° C'_p values, and are overpredicted.

In addition, Table 2 demonstrates the benefit of carefully selecting input features, as it compares models using different feature sets. In all presented cases, feature sets include the RANS five features, introduced in 3.1 and in addition, their combination with coordinates and wind direction. Table 2 reveals the significant effect of different feature sets

Table 3
Column 2: The type of the hyperparameter and its default/initial value. Column 3: the list of all hyperparameter values tested during the hyperparameter optimization. Column 4: The optimized hyperparameters obtained for each model.

Model	Hyperparameter: Default value	Hyperparameter values	Optimized value
Support vector regression	Kernel: Radial basis function	Linear, Polynomial, Radial basis function, Sigmoid	Radial basis function
	Regularization parameter: 1	1, 10, 100, 1000, 10 000	1
	Kernel coefficient: 1/(no. features × Variance)	1/(no. features × Variance), 1/no. features	1/(no. features × Variance)
Random forest	Number of trees: 100	20 evenly spaced values from 100 to 2000	1100
	Maximum depth of a tree: pure tree	10 evenly spaced values from 10 to 100	50
	Minimum number of samples at leaf node: 1	1, 2, 4	1
	Minimum number of samples required to split an internal node: 2	2, 5, 10	2
	Bootstrap: True	True, False	True
Gradient boosting	Number of boosting stages: 100	Range from 1 to 2000	1217
	Learning rate: 0.1	0.01, 0.1, 0.05, 0.2, 0.5	0.15
	Minimum number of samples at leaf node: 1	1, 2, 4: *1*	1
	Maximum depth of the individual tree: 3	1 to 10	3
Artificial neural network	Number of hidden layers: 5	1, 2, 3, 4, 5, 6	4
	Number of neurons: 10	5, 10, 16, 32, 64, 128, 256	32, 64, 64, 32
	Learning rate: 0.001	0.0005, 0.001, 0.005, 0.01, 0.1, adaptive	Adaptive
	Activation function: relu, final layer: linear	tanh, relu, linear	relu, final layer: linear
	Dropout rate: -	0.1, 0.2, 0.3, 0.4, 0.5	0.15
	optimizer: Adam	Adam, Adamax, RMSprop	Adamax
	Batch size: 64	8, 16, 32	32
	Number of epochs: 100	100, 200, 300, 400	300

on the model's output. It can be noticed that the results obtained with only five flow features are not satisfactory enough for some models as the R^2 values are negative, meaning that the model fails to follow the trend of the data. On the other hand, it is apparent that the inclusion of coordinates and/or wind direction improves the model performance remarkably. Table 2 demonstrates that the model trained with five flow features and wind direction performs slightly better than the model trained with five flow features and coordinates. This is because the wind direction is a dominant feature due to its strong correlation with the outputs. The feature selection and the dominant features are later discussed in Section 4.3. Nonetheless, the results obtained with all nine features are superior.

Table 2 also highlights the benefits of hyperparameter optimization. It can be seen that the optimal choice of hyperparameters significantly improves the model performance. In the case of the artificial neural network and C_p', the R^2 value increases by 60%, and RMSE decreases by 30% after hyperparameter optimization. Table 3 lists all tested hyperparameter values and shows the default/initial as well as the final optimized hyperparameter values that are used for each model. The initial hyperparameters for the support vector regression, random forest and gradient boosting are as defined in the scikit-learn library [74]. There is no change in support vector regression results before and after hyperparameter optimization as the default parameters give the best results. Additionally, as discussed in Section 3.4, an adaptive learning rate has been used for the neural network. The algorithm reduces the learning rate by 0.2 if no improvement is noticed in the validation loss for two consecutive epochs.

Therefore, as the optimized artificial neural network obtained the best results, the remainder of the paper will focus on this specific methodology. In addition, it can be concluded that even though it is evident that the quality of the results improves noticeably after the hyperparameter optimization, the results obtained for the 15° wind direction case with $\{0°, 45°\}$ wind direction datasets are not yet satisfactory.

4.2. Universal ML model

Results for prediction of C_p' using the baseline model reveal that an additional wind direction dataset is required to reach satisfactory performance, as only two extreme wind directions $\{0°, 45°\}$ cannot model multi-directional wind loads properly. However, the criterion for selecting the intermediate wind direction is not clear. Intuitively, selecting the mid wind direction of 22.5° may be the obvious choice since the predictability of the model drops further away from the nearest neighbor points [15]. Moreover, this can be useful, as it is not known in which case the ML model will perform best in advance. Nevertheless, given that the prediction of 15° was the worst result with the baseline model, one could argue that including this specific wind direction might provide superior performance. Therefore, both approaches are tested, i.e. the optimized ANN model is trained with $\{0°, 15°, 45°\}$ and $\{0°, 22.5°, 45°\}$ wind directions, whereas the trained models are tested for the 30° wind direction. The obtained R^2 values in the case of C_p' are 0.8824 and 0.8063 when the ANN models trained with $\{0°, 22.5°, 45°\}$ and $\{0°, 15°, 45°\}$ datasets are used for testing the 30° wind direction test case, respectively. The results clearly show superior performance when 22.5° wind direction is included.

Table 4
Test average R^2 values for the optimized ANN trained on different training datasets with nine features.

Training data	$\bar{C}_p$ Average R^2	C_p' Average R^2
$0°, 7.5°, 45°$	0.9566	0.6657
$0°, 15°, 45°$	0.9655	0.8296
$0°, 22.5°, 45°$	0.9832	0.8531
$0°, 30°, 45°$	0.9613	0.7856
$0°, 37.5°, 45°$	0.9577	0.6779

Considering that the universal training dataset should provide optimal performance over the entire wind rose, the average R^2 values obtained over all test wind directions are compared. Furthermore, all possible training sets are considered by adding a new dataset for one wind direction between two extremes with the stepsize of 7.5°, as shown in Table 4. The testing is performed on all wind directions that are not included in the training set, e.g., the wind directions 7.5°, 22.5°, 30° and 37.5° are the test cases for the training dataset $\{0°, 15°, 45°\}$. Since R^2 is a scale-independent metric, it is used as a performance measure. The averaged R^2 results over all test wind directions confirm the initial hypothesis that the mid wind direction of 22.5° provides the most informative outcomes; therefore, the training set $\{0°, 22.5°, 45°\}$ is used in all further experiments.

The benefits of adding the intermediate wind direction can be estimated by comparing the results obtained using the universal model in Table 5 to the baseline model in Table 2 with the complete feature set, i.e., all nine features. The predictions obtained for all tested wind

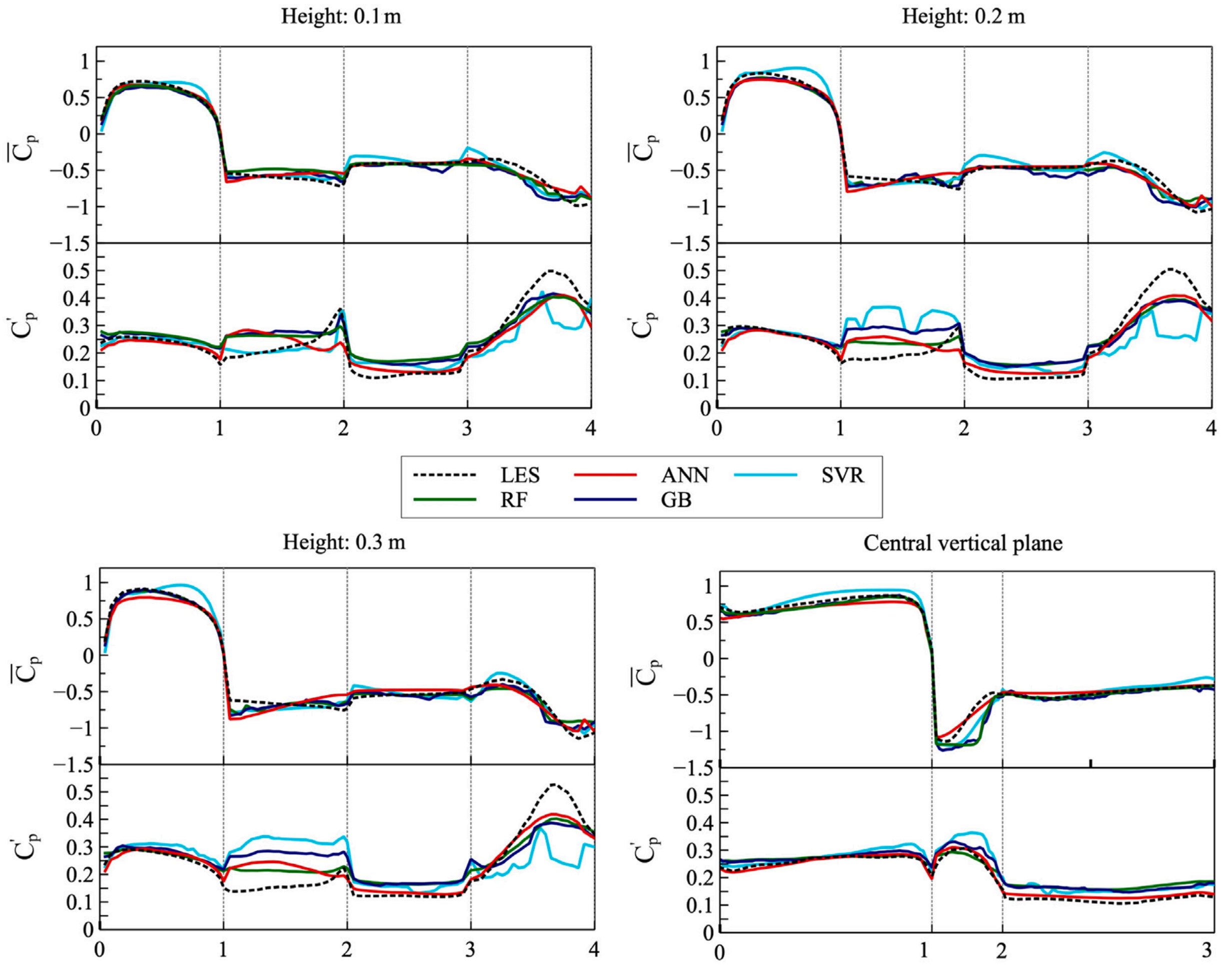

Fig. 9. Comparison between results from LES and different ML models trained on $\{0°, 45°\}$ wind direction datasets with nine features and tested for the 15° wind direction. The three plots show the $\bar{C}_p$ and C'_p results at three different heights across the perimeter of the building (Fig. 1c) and one plot along the lines in the central vertical plane (Fig. 1d).

Table 5
Test RMSE and R^2 values of the optimized ANN trained on $\{0°, 22.5°, 45°\}$ wind direction datasets and tested for 15° and 30° wind direction datasets.

Model	$\bar{C}_p$		C'_p		$\bar{C}_p$		C'_p	
	15°				30°			
Artificial neural network	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE
5 flow features	0.9653	0.1030	0.3672	0.0822	0.9672	0.0907	0.4769	0.0749
5 flow features + Angle	0.9717	0.0930	0.7985	0.0478	0.9759	0.0777	0.8611	0.0412
5 flow features + Coords	0.9708	0.0945	0.7849	0.0501	0.9796	0.0890	0.8308	0.0485
5 flow features + Coords + Angle	0.9764	0.0871	0.8294	0.0438	0.9814	0.0701	0.8824	0.0355
6 dominant features	0.9768	0.0850	0.8256	0.0439	0.9812	0.0702	0.8861	0.0352

directions are substantially improved, as in all cases, R^2 values are above 0.8 when considering the complete feature set. Again, the worst results are obtained for the 15° wind direction. Nevertheless, Table 5 reports that the R^2 value for the 15° case has increased by 43%, and the RMSE has decreased by 37% for the complete feature set case. These significant improvements are also well depicted in Fig. 10. A similar improvement is observed for all other tested angles, and as an example, results obtained for 30° are presented as well in Table 5. R^2 and RMSE values for 30° with all features demonstrate a very good fit, as the R^2 value has been improved by approximately 50% after considering all features.

Table 5 again highlights the necessity of defining the most important features, as it indicates that the inclusion of the wind angle or coordinates as additional features significantly improves the model's performance. This is well observed in Fig. 11, where instability is obvious when an initial subset of 5 flow features is used and successfully, this is overcome by simply including the wind direction or coordinates in the feature set.

Since the observed training dataset $\{0°, 22.5°, 45°\}$ provides satisfactory results for all tested wind directions, this dataset is considered as the universal training dataset.

4.3. *Dominant features*

So far, the obtained results are discussed based on five flow features and their combinations with additional features such as wind direction and coordinates. However, some features might not contribute much to the final output among these nine features. Thus, feature selection techniques are utilized to determine the most dominant features and omit the irrelevant information in the training datasets.

The F-statistics and mutual information techniques explained in Section 3.2 are used for feature selection. The results are presented

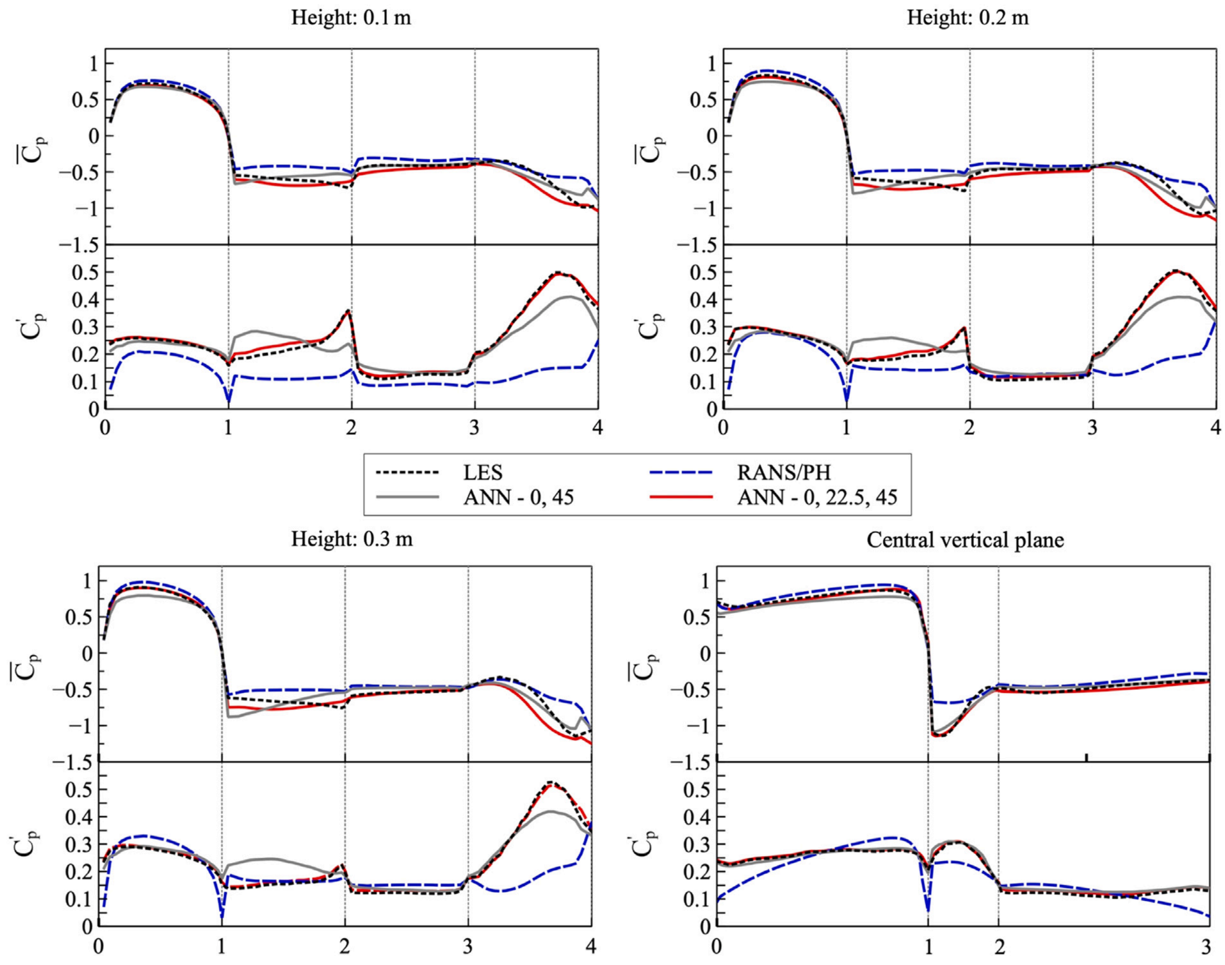

Fig. 10. Comparison between results from LES, RANS (or Paterson-Holmes (PH) model for C_p') and the ANN trained on $\{0°, 45°\}$ and $\{0°, 22.5°, 45°\}$ wind direction datasets with nine features and tested for the 15° wind direction. The three plots show the $\bar{C}_p$ and C_p' results at three different heights across the perimeter of the building (Fig. 1c) and one plot along the lines in the central vertical plane (Fig. 1d).

in Fig. 12, where a high value depicts a strong correlation between that feature and the output. The barplot also shows the ranking of the features based on obtained results at the bottom of the respective bar. The F-statistics technique captures the linear dependency between the output and features, whereas mutual information can consider nonlinear dependencies. When modeling $\bar{C}_p$ both techniques depict $\bar{C}_{p,\mathrm{RANS}}$ as the most dominant attribute, as expected. Besides, both techniques in modeling $\bar{C}_p$ and C_p' exhibit mean pressure coefficient, wind direction, x and z coordinates as the most prominent features with only slight variation in their ranks. The only exception is related to the mutual information result for $\bar{C}_p$ where the non-dimensional pressure gradient is ranked as the fourth feature followed by the z coordinate. Nonetheless, it can be concluded that both methods show that the most dominant features are the mean pressure coefficient, wind direction and coordinates.

In contrast, both methods infer that the non-dimensional inflow velocity magnitude, friction coefficient, and y coordinate are the least important attributes. The lesser importance of the y coordinate can be attributed to its lower variance compared to the other two coordinates, which was noticed in [50] as well. Similarly, as the inflow velocity magnitude does not provide significant variation in the data and information on the different flow regimes, it is one of the least important features.

From Fig. 12, it can be stated that ranks after the first four most dominant features do not change much considering both techniques and both outputs. Thus, to determine the relevant features, the ANN model is tested for the 15° wind direction test case based on the $\{0°, 22.5°, 45°\}$ training dataset by increasing the number of features sequentially, starting with the first four. It is observed that the six leading features (mean pressure coefficient, x and z coordinates, non-dimensional turbulence kinetic energy, non-dimensional pressure gradient) deliver the maximum R^2 value for both $\bar{C}_p$ and C_p' predictions, and perform as good as the complete feature set. Thus, they form the optimal feature subset. This is confirmed in Fig. 13. Improvement in computational cost is also witnessed: the model with the optimal feature subset trains approximately two times faster than the model with the complete feature set.

Lamberti and Gorle [15] have employed principal component analysis to determine which flow features contribute most to the final output. Thus, to verify and confirm the results, in the present study the principal component analysis is performed with only five features, similar to their study. It is observed that the first two principal components comprise 99% of the total variance, and the mean pressure coefficient and pressure gradient are the most dominant flow features. However, the results show that considering features like the coordinates and the wind direction improves the model performance significantly, and thus they are included in the analysis. Again, with nine features, it is noticed that the first two principal components comprise 99% of the total variance and received dominant features similar to the F-statistics results.

Therefore, based on the presented analysis, it can be concluded that the most optimal features are the first six features highlighted in Fig. 12.

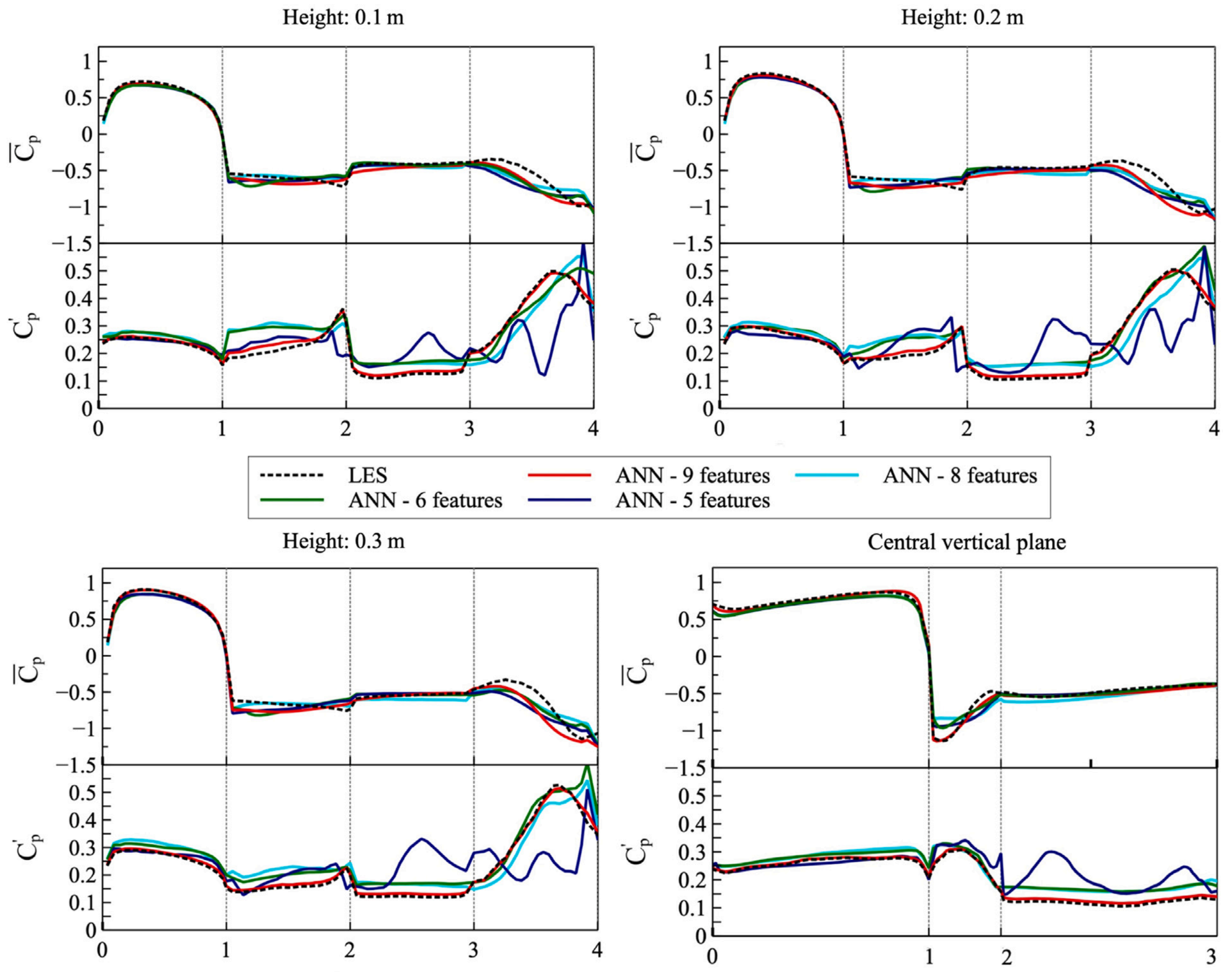

Fig. 11. Comparison between results from LES and ANN trained on {0°, 22.5° 45°} wind direction datasets with different input features and tested for the 15° wind direction. Considered features are: five flow features, five flow features plus wind direction or coordinates (6 or 8 features), and five flow features with both wind direction and coordinates (9 features). The three plots show the $\bar{C}_p$ and C'_p results at three different heights across the perimeter of the building (Fig. 1c) and one plot along the lines in the central vertical plane (Fig. 1d).

4.4. Confidence intervals

A confidence interval can be used to measure the degree of uncertainty of the developed machine learning model [75]. A way to estimate this uncertainty related to the machine learning framework is to use bootstrap as done in [15]. Bootstrap is a famous resampling method used to estimate the statistics of machine learning algorithms. Also, bootstrapping might help to reduce variance and, by extension, prevent overfitting. However, the need for recursion in order to produce N samples complicates implementation and requires much more computational power. Thus, this overall increases the time required to train the machine learning framework. See [76] for more details on the bootstrap method.

In this work, to assess the quality of the developed feedforward neural network, 100 training samples by sampling the original dataset with replacement are generated. The results of the ANN trained on {0°, 22.5°, 45°} datasets with six optimal features and five flow features and tested for 15° wind direction are discussed here. Those cases are selected as they represent the optimal case and the case with the worst prediction, as shown in Table 5.

It is observed that 92.04% of LES $\bar{C}_p$ and 68.37% of LES C'_p data lie in the confidence interval of the model trained with five flow features. On the other hand, a considerable improvement is observed in the results for the model trained with six optimal features, where 97.29% of LES $\bar{C}_p$ and 94.48% of LES C'_p data lie in the confidence interval. This improvement is also clearly illustrated in Fig. 14, which shows that most of the ANN results fall within the confidence interval. Also, it can be concluded that the developed model demonstrates great precision as the confidence is narrow.

Another phenomenon that is observed is the smoothing of results due to bootstrapping. For example, in Fig. 11, one can see that the results obtained with five features have some fluctuations in the last section. However, by applying bootstrapping and due to averaging, these oscillations are damped. Anyhow, Fig. 13 show that the model with six optimal features outperforms the five flow features by a considerable margin, clearly highlighting the advantages of feature engineering suggested in this paper. Thus, feature engineering combined with hyperparameter optimization is suggested as a reliable approach to tackle similar problems.

4.5. Extrapolation problem and transfer learning application

Here the developed framework is tested for extrapolation problems. Unfortunately, such problems are inherently hard to solve using regression-based techniques. This is because the trend in the data, as summarized by the trained model, does not necessarily hold outside the model's scope. Consequently, extrapolation beyond the range of the training data must be treated skeptically.

In particular, of interest are predictions for the 0° wind direction test case when the model has been trained on {15°, 30°, 45°} wind

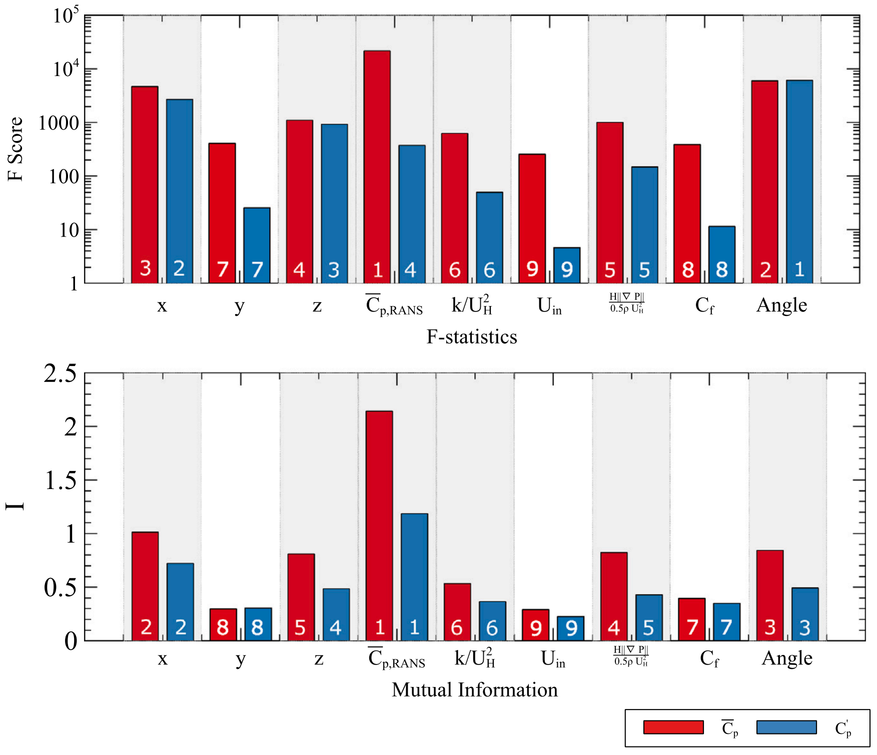

Fig. 12. Feature selection results using F-statistics and mutual information. The optimal six features are highlighted in gray color, and the rank of each feature is mentioned at the bottom of the respective bar. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

direction datasets and 45° when the model is trained on $\{0°, 15°, 30°\}$ wind direction datasets. For the 0° test case, the ANN model gives $R^2 = 0.9113$ and RMSE = 0.1791 for $\bar{C}_p$, while $R^2 = 0.1291$ and RMSE = 0.0811 for C_p'. Similarly, for the 45° test case, the ANN model offers $R^2 = 0.9308$ and RMSE = 0.1073 for $\bar{C}_p$, while $R^2 = 0.2608$ and RMSE = 0.0541 for C_p'. Although the results for $\bar{C}_p$ are satisfactory, the neural network fails to accurately predict extrapolation results of C_p', as they have R^2 values below 0.30. This can also be witnessed in Fig. 15, where C_p' values around the buildings are compared with the LES data. Thus, the 0° case is discussed here as it provides severely underpredicted results. This is mainly prominent in zones 1–2 and 3–4 of the height plots in Fig. 15, where the predicted C_p' values are nowhere near the LES values. Also, between two considered extrapolation cases, the results for 0° are worst than 45°.

Another interesting problem with the extrapolation is that the predicted C_p' results lack symmetry, which is expected for the 0° case, as evident from the LES results. This is because the training dataset contains simulations for 15°, 30°, and 45° wind directions, which do not hold such inherent symmetry of $\bar{C}_p$ and C_p' in their flow fields, as demonstrated in Fig. 7. Thus, the ANN could not learn this symmetry during the training phase and failed to capture it for the test problem.

To tackle this extrapolation problem and for plausible improvements of C_p' predictions, a transfer learning technique is considered in this work, which has been used to solve similar problems in the literature before [77–79]. Transfer learning is a technique that uses a pre-trained model that has already been learned on a wide variety of data to solve a similar problem that lacks data. It improves the learning of a new task by utilizing the knowledge from a pre-trained model. The idea is to reuse partly or wholly the pre-trained model on a similar problem to accelerate training and improve performance.

The following workflow is pursued to construct the transfer learning framework. Since RANS results for the 0° case are already available, this information can be used to train the model and construct the pre-trained model. Basically, the pre-trained model is obtained by learning on the data comprised of only RANS features for all wind directions and corresponding $C'_{p,\mathrm{RANS}}$ values obtained using the empirical model. Therefore, the developed model has already learned the features that correspond to the 0° case and has some information related to the distribution of C_p' values, which makes it robust during the re-training process. Furthermore, a few layers of this pre-trained model are taken during the transfer learning process and frozen to avoid destroying any of the vital information they contain during the fine-tuning process. Also, if necessary, some new trainable layers are added to the existing model. Finally, the model is trained on the new dataset having RANS and LES results at 15°, 30°, and 45° wind directions and six optimal features. Additionally, the learning rate is reduced to prevent overfitting. To solve the problem at hand, two extra layers are added to the pre-trained model with 32 neurons in each layer. The pre-trained model has a total of 8705 parameters, and the transfer learning model has a total of 9761 parameters, out of which 1409 are trainable.

The results obtained using the transfer learning model are also shown in Fig. 15. The trained transfer learning model delivers $R^2 = 0.5310$ and RMSE = 0.0648 for the 0° test case, which evidently shows better results than the results obtained using the traditional model trained that gives $R^2 = 0.1291$. Even though the model performance is still not comparable to the interpolation cases considered in this study, the transfer learning approach has opened a new doorway to tackle similar problems.

4.6. Computational efficiency

Machine learning models can also be compared based on the computational efficiency. Computational times required to perform all mentioned machine learning tasks are summarized below.

The training time is measured as based on the C_p' as an output and considering all 9 features. The training times for all cases are relatively low. Support vector regression, random forest, gradient boosting and artificial neural network are trained in 0.13, 0.24, 0.32, 0.4 CPU hours, respectively. For that purpose, a regular personal computer is used (8 cores Intel i9 11th gen. at 2.90 GHz). A more computationally

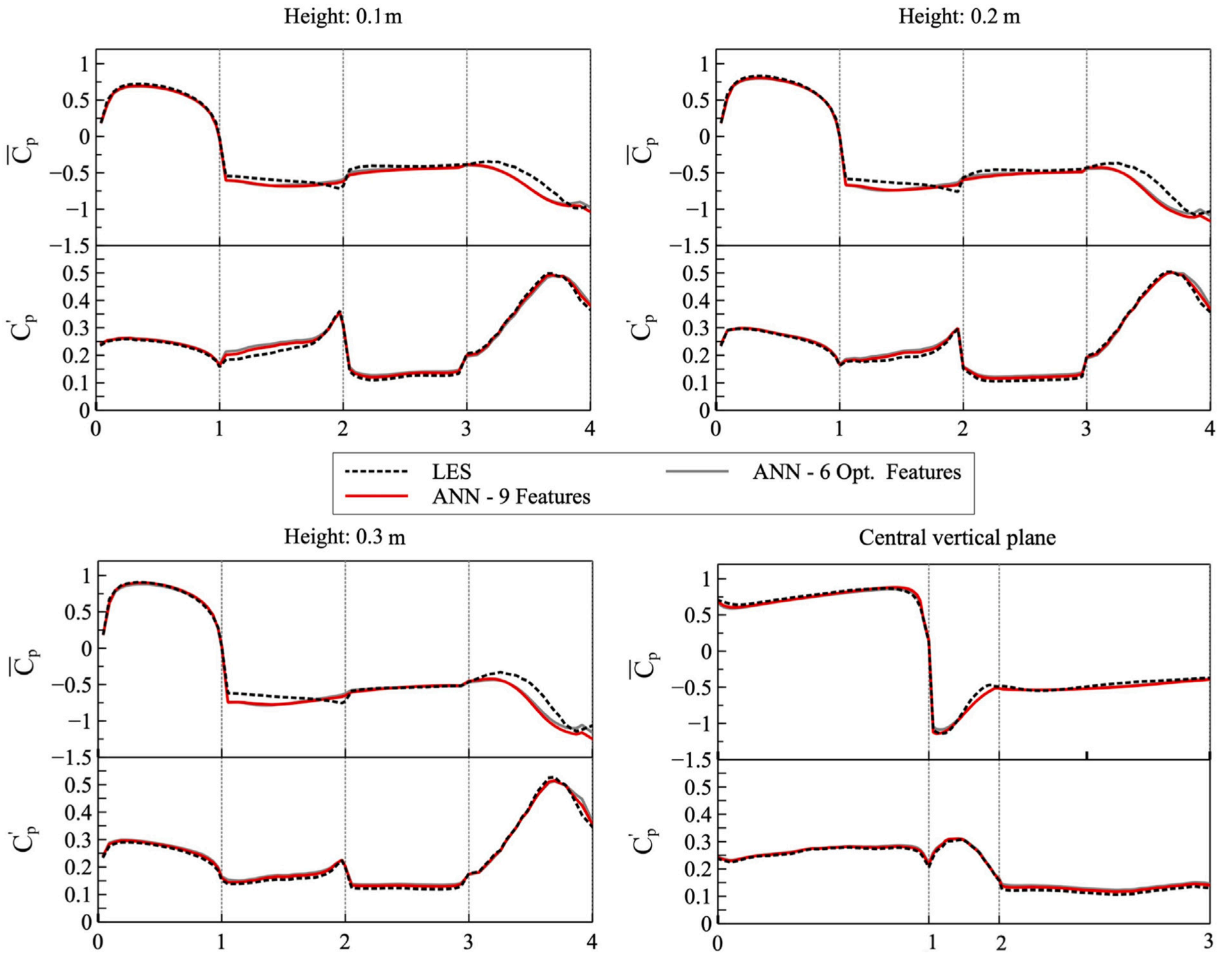

Fig. 13. Comparison between results from LES and ANN trained with nine features and six optimal features on {0°, 22.5° 45°} wind direction datasets and tested for the 15° wind direction. The three plots show the $\bar{C}_p$ and C_p' results at three different heights across the perimeter of the building (Fig. 1c) and one plot along the lines in the central vertical plane (Fig. 1d).

demanding task is related to the hyperparameter optimization. For example, hyperparameter optimization of support vector regression, random forest, gradient boosting, and artificial neural network requires approximately 4.2, 12.3, 12.5, and 14.7 CPU hours, respectively. Hyperparameter optimization is run for all mentioned cases on the University of Luxembourg high performance computing facility (2 nodes with 2 cores Skylake at 4.5 GHz). It is apparent that hyperparameter optimization of the artificial neural network is the most time-consuming option, as it is more than three times slower than the support vector regression. However, from Table 2, it is clear that the results obtained with the artificial neural network are significantly superior to the support vector regression. Regarding the bootstrapping process, for 100 training samples it takes around 16.8 CPU hours on a personal computer (8 cores Intel i9 11th gen. at 2.90 GHz).

On the other hand, numerical methods, in particular LES are much more computationally time consuming. For each wind direction, an LES simulation needs about 5.65×10^4 CPU hours on the University of Luxembourg high performance computing facility (7 nodes with 28 cores Skylake at 4.5 GHz). Each RANS simulation took around 20 CPU hours (6 core at Intel Core i-7 at 2.6 GHz).

5. Limitations of the study and future perspectives

Some limitations of the study are mentioned below:

- The present study is focused on an isolated high-rise building without considering the impact of the surrounding buildings in an urban setting. Moreover, the building is centrosymmetric. As the machine learning models are data-driven, any change (considering more complex building geometry or neighborhood) might need to retrain these models again. Therefore, future work will explore the ways of increasing this challenged universality by applying techniques like transfer learning.
- The considered neural network is a black box that does not give any insights into the structure of the function being approximated. Thus, in the future, techniques that can provide a better understanding of how a neural network makes predictions based on interpretable models can be explored.
- The analysis encompasses mean and rms pressure coefficient to assess the wind load on structures. As peak pressure is another particularly relevant quantity of interest when assessing the wind load on the facades, further development can focus on this specific parameter.

6. Conclusions

A machine learning framework is proposed that uses a larger number of computationally affordable RANS simulations with an optimized number of computationally costly LES simulations aiming to accurately predict mean and rms pressure coefficients on high-rise building, covering the whole wind rose. The full dataset consists of the RANS and LES simulations for 7 different wind directions: 0°, 7.5°, 15°, 22.5°, 30°, 37.5°, 45°. To optimize the needed number of costly LES simulations, an adaptive training strategy is proposed. It starts with a baseline model

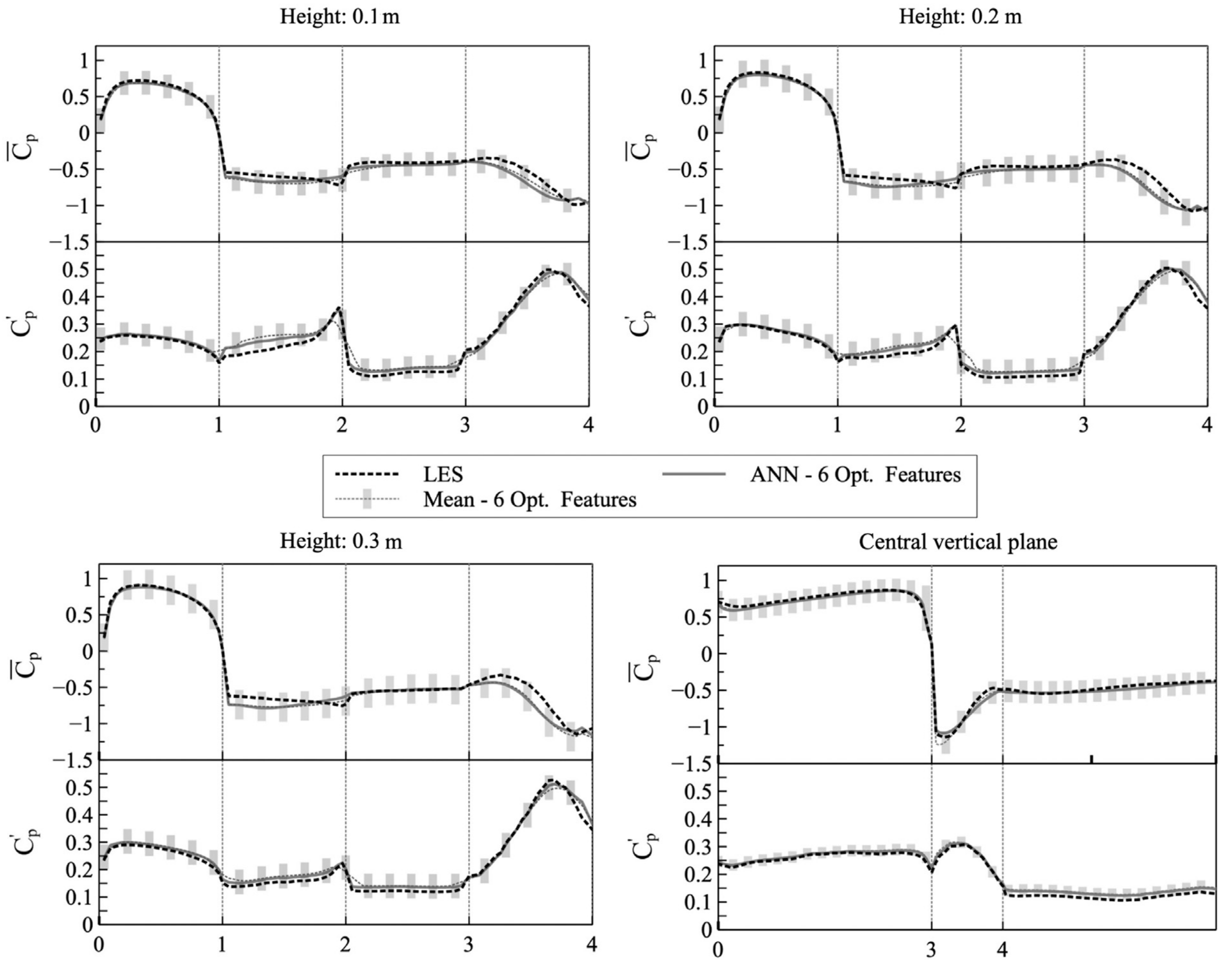

Fig. 14. Comparison between results from LES and ANN trained on $\{0°, 22.5° 45°\}$ wind direction datasets with six optimal features with bootstrap mean and confidence interval and tested for the 15° wind direction. The three plots show the $\bar{C}_p$ and C'_p results at three different heights across the perimeter of the building (Fig. 1c) and one plot along the lines in the central vertical plane (Fig. 1d).

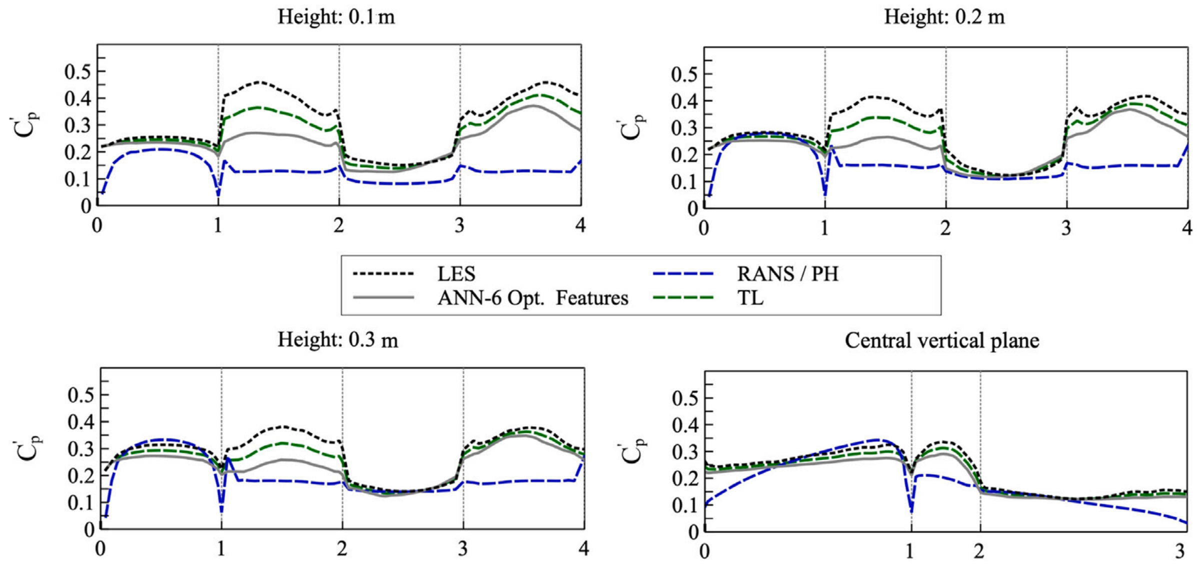

Fig. 15. Comparison among results from LES, RANS, ANN, and transfer learning model trained on $\{15°, 30° 45°\}$ wind direction datasets with six optimal features and tested for the 0° extrapolation case. The three plots show the C'_p results at three different heights across the perimeter of the building (Fig. 1c) and one plot along the lines in the central vertical plane (Fig. 1d).

that considers only two LES simulations of the most extreme wind direction cases as a training set, i.e. the approaching wind directions of 0° and 45°. This is followed by determining a supplementary dataset that increases the accuracy to an acceptable level. The accuracy of the model is judged by the scale-dependent RMSE metric and, in addition, by the R^2 value, where a value above 0.8 is considered satisfactory. Different machine learning algorithms are considered, including support vector machine, random forest, gradient boosting and artificial neural networks.

The main conclusions are summarized below:

- The study shows that the set of only 3 LES simulations for 3 wind directions is sufficient to define the universal training dataset that can accurately predict distributions of $\bar{C}_p$ and C'_p over the high-rise building for the entire wind rose. These wind directions comprise two extreme cases in terms of flow pattern: 0° and 45°; and the median wind direction of 22.5°. In addition, significant improvement is observed in the case of C'_p values when compared to the standard empirical model, even when a dataset of $\{0°, 45°\}$ is used.
- The best-performed machine learning model is based on the artificial neural network approach. Moreover, hyperparameter optimization is shown to significantly improve the model predictions. In the case of the neural network, the R^2 value related to C'_p has increased by 60% and the RMSE decreased by 30% after applying hyperparameter optimization.
- All developed machine learning models perform comparatively better for $\bar{C}_p$ than C'_p, due to the use of RANS $\bar{C}_p$ as an input feature.
- The prediction can be improved even further by carefully selecting the input features. After applying feature selection techniques over a wider range of 9 input features, 6 dominant features are defined. These are: mean pressure coefficient, x (streamwise) and z (lateral) coordinates, nondimensional turbulence kinetic energy, nondimensional pressure gradient and wind direction. It is observed that those dominant features provide the same level of accuracy as the complete feature set. Moreover, the same 6 features are considered to be optimal for both outputs, $\bar{C}_p$ and C'_p.
- The confidence intervals are evaluated by utilizing the bootstrap method. It is observed that the confidence is very narrow for the model with the dominant features, confirming its high accuracy.
- Other training datasets are explored, in particular related to extrapolation problem. The test case with the worst results is related to the 0° wind direction and prediction of C'_p. The transfer learning technique brought a noticeable improvement of the R^2 values by increasing them from 0.13 to 0.53.
- The developed multi-fidelity framework is 2.3 times computationally more efficient in terms of CPU time than performing LES simulations for all 7 wind directions.

The findings of this study have broader relevance for the use of machine learning in wind engineering problems, as they provide important methodological steps that should be considered when similar problems are treated.

CRediT authorship contribution statement

Anina Šarkić Glumac: Writing – original draft, Visualization, Validation, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Onkar Jadhav:** Writing – original draft, Visualization, Validation, Resources, Methodology, Investigation, Formal analysis, Data curation. **Vladimir Despotović:** Writing – review & editing, Methodology. **Bert Blocken:** Writing – review & editing, Conceptualization. **Stephane P.A. Bordas:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Anina Glumac and Onkar Jadhav reports financial support was provided by Luxembourg National Research Fund (FNR). Co-author Bert Blocken is acting as one of the editors for Building and Environment journal.

Data availability

Data will be made available on request.

Acknowledgments

The authors A.S.G., O.J. and S.P.A.B. would like to acknowledge the support of the "Fonds National de la Recherche, Luxembourg" (FNR) for funding the CORE Junior project DATA4WIND - "Data-Driven Approach for Urban Wind Energy Harvesting", C19/SR/13639741. The co-author S.P.A.B. received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 811099 TWINNING Project DRIVEN for the University of Luxembourg. Numerical simulations were run on the HPC centre of University of Luxembourg.

References

[1] T. Tamura, K. Nozawa, K. Kondo, AIJ guide for numerical prediction of wind loads on buildings, J. Wind Eng. Ind. Aerodyn. 96 (2008) 1974–1984.
[2] M. Ricci, L. Patruno, I. Kalkman, S. de Miranda, B. Blocken, Towards LES as a design tool: Wind loads assessment on a high-rise building, J. Wind Eng. Ind. Aerodyn. 180 (2018) 1–18.
[3] T. Stathopoulos, H. Alrawashdeh, Wind loads on buildings: A code of practice perspective, J. Wind Eng. Ind. Aerodyn. 206 (2020) 104338.
[4] G. Pomaranzi, L. Amerio, P. Schito, G. Lamberti, C. Gorlé, A. Zasso, Wind tunnel pressure data analysis for peak cladding load estimation on a high-rise building, J. Wind Eng. Ind. Aerodyn. 220 (2022) 104855.
[5] P. Irwin, Wind engineering challenges of the new generation of super-tall buildings, J. Wind Eng. Ind. Aerodyn. 97 (2009) 328–334.
[6] H. Dongmei, H. Shiqing, H. Xuhui, Z. Xue, Prediction of wind loads on high-rise building using a BP neural network combined with POD, J. Pf Wind Eng. Ind. Aerodyn. 170 (2017) 1–17.
[7] J. Zhang, Q. Li, Wind tunnel test and field measurement study of wind effects on a 600-m-high super-tall building, Struct. Des. Tall Spec. Build. 26 (17) (2017) e1385.
[8] M. Mooneghi, R. Kargarmoakhar, Aerodynamic mitigation and shape optimization of buildings, J. Build. Eng. 6 (2016) 225–235.
[9] M.M. Salehinejad, R.G.J. Flay, A review of approaches to generate equivalent static and synthetic wind loads on tall buildings for the preliminary stage of design, J. Pf Wind Eng. Ind. Aerodyn. 219 (2021) 104823.
[10] T. Stathopoulos, Computational wind engineering: Past achievements and future challenges, J. Wind Eng. Ind. Aerodyn. 67–68 (1997) 509–532.
[11] Y. Tominaga, T. Stathopoulos, CFD simulation of near-field pollutant dispersion in the urban environment: A review of current modeling techniques, Atmos. Environ. 79 (2013) 716–730.
[12] B. Blocken, LES over RANS in building simulation for outdoor and indoor applications: A foregone conclusion? Build. Simul. 11 (5) (2018) 821–870.
[13] B. Blocken, 50 Years of computational wind engineering: Past, present and future, J. Wind Eng. Ind. Aerodyn. 129 (2014) 69–102.
[14] T. Stathopoulos, The numerical wind tunnel for industrial aerodynamics: Real or virtual in the new millennium? Wind Struct. 5 (2002) 193–208.
[15] G. Lamberti, C. Gorle, A multi-fidelity machine learning framework to predict wind loads on buildings, J. Wind Eng. Ind. Aerodyn. 214 (2021) 104647.
[16] A. Kareem, Emerging frontiers in wind engineering: Computing, stochastics, machine learning and beyond, J. Pf Wind Eng. Ind. Aerodyn. 206 (2020) 104320.
[17] S. Murakami, Comparison of various turbulence models applied to a bluff body, J. Wind Eng. Ind. Aerodyn. 46–47 (1993) 21–36.
[18] S. Murakami, A. Mochida, Y. Hayashi, S. Sakamoto, Numerical study on velocity-pressure field and wind forces for bluff bodies by k-e, ASM and LES, J. Wind Eng. Ind. Aerodyn. 44 (1992) 2841–2852.
[19] S. Murakami, Computational wind engineering, J. Wind Eng. Ind. Aerodyn. 36 (1990) 517–538.
[20] S. Murakami, Numerical simulation of turbulent flowfield around cubic model: Current status and applications of k-e model and LES, J. Wind Eng. Ind. Aerodyn. 33 (1990) 139–152.

[21] A. Forrester, A. Sobester, A. Keane, Multi-fidelity optimization via surrogate modelling, in: Proceedings of Proceedings of the Royal Society of London a: Mathematical, Physical and Engineering Sciences, 2007, pp. 3251–3269.
[22] K. Collins, A Multi-Fidelity Framework for Physics Based Rotor Blade Simulation and Optimization, (Ph.D. Dissertation), Georgia Institute of Technology, 2008.
[23] F. Ding, A. Kareem, A multi-fidelity shape optimization via surrogate modeling for civil structures, J. Pf Wind Eng. Ind. Aerodyn. 178 (2018) 49–56.
[24] J. Fu, S. Liang, Q. Li, Prediction of wind-induced pressures on a large gymnasium roof using artificial neural networks, Comput. Struct. 85 (2007) 179–192.
[25] Y. Chen, G. Kopp, D. Surry, Prediction of pressure coefficients on roofs of low buildings using artificial neural networks, J. Pf Wind Eng. Ind. Aerodyn. 91 (2003) 423–441.
[26] J. Tian, K.G. Gurley, M.T. Diaz, P.L. Fernandez-Caban, J. Forrest, R. Fang, Low-rise gable roof buildings pressure prediction using deep neural networks, J. Pf Wind Eng. Ind. Aerodyn. 196 (2020) 104026.
[27] Y. Chen, G. Kopp, D. Surry, Interpolation of wind-induced pressure time series with an artificial neural network, J. Pf Wind Eng. Ind. Aerodyn. 90 (2002) 589–615.
[28] G. Hu, L. Liu, D. Tao, J. Song, K. Tse, K. Kwok, Deep learning-based investigation of wind pressures on tall building under interference effects, J. Pf Wind Eng. Ind. Aerodyn. 201 (2020) 104138.
[29] J. Huang, Q. Li, X. Han, Recovery of missing field measured wind pressures on a supertall building based on correlation analysis and machine learning, J. Pf Wind Eng. Ind. Aerodyn. 231 (2022) 105237.
[30] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, Annu. Rev. Fluid Mech. 51 (2019) 4357–4377.
[31] H. Hemida, A.Š. Glumac, G. Vita, K.K. Vranešević, R. Höffer, On the flow over high-rise building for wind energy harvesting: An experimental investigation of wind speed and surface pressure, Appl. Sci. 10 (2020) 5283.
[32] A.Š. Glumac, H. Hemida, G. Vita, K.K.V. c, R. Höffer, Wind tunnel experimental data for flow characteristics above the roof of isolated high-rise building for wind energy harvesting considering two shapes of the roof, flat roof and deck roof, mendeley data v1, 2020, https://doi.org/10.17632/jp7vc8tf7w.1.
[33] B. Blocken, T. Stathopoulos, J. Carmeliet, Wind environmental conditions in passages between two long narrow perpendicular buildings, J. Aerosp. Eng. 21 (2008) 280–287.
[34] Eurocode 1, EN 1991-1-4: Actions on structures part 1-4: General actions - wind actions, Eur. Comm. Stand. (2005) https://doi.org/10.17632/jp7vc8tf7w.1.
[35] K. Kostadinović-Vranešević, G. Vita, S.P.A. Bordas, A.Š. Glumac, Furthering knowledge on the flow pattern around high-rise buildings: LES investigation of the wind energy potential, J. Wind Eng. Ind. Aerodyn. 226 (2022) 105029.
[36] Y. Tominaga, A. Mochida, R. Yoshie, H. Kataoka, T. Nozu, M. Yoshikawa, T. Shirasawa, AIJ guidelines for practical applications of CFD to pedestrian wind environment around buildings, J. Wind Eng. Ind. Aerodyn. 96 (2008) 1749–1761.
[37] T. Norton, J. Grant, R. Fallon, D.-W. Sun, Optimizing the ventilation configuration of naturally ventilated livestock buildings for improved indoor environmental homogeneity, Build Environ. 45 (2010) 983e95.
[38] R. Ramponi, B. Blocken, CFD simulation of cross-ventilation for a generic isolated building: impact of computational parameters, Build. Environ. 53 (2012) 34–48.
[39] B. Blocken, T. Stathopoulos, J. Carmeliet, CFD simulation of the atmospheric boundary layer: wall function problems, Proc. Natl. Acad. Sci. USA 41 (2007) 238–252.
[40] B. Blocken, Computational fluid dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations, Build. Environ. 91 (2015) 219–245.
[41] F. Menter, Two-equation eddy-viscosity turbulence models for engineering applications, AIAA J. 32 (1994) 1598–1605.
[42] M. Thordal, J. Bennetsen, S. Capra, H. Koss, Engineering approach for a CFD inflow condition using the precursor database method, J. Wind Eng. Ind. Aerodyn. 203 (2020) 104210.
[43] S. Murakami, Overview of turbulence models applied in CWE-1997, J. Wind Eng. Ind. Aerodyn. 74–76 (1998) 1–24.
[44] R. Issa, Solution of the implicitly discretised fluid flow equations by operator-splitting, J. Comput. Phys. 62 (1986) 40–65.
[45] F. Nicoud, F. Ducros, Subgrid-scale stress modelling based on the square of the velocity gradient tensor, Flow Turbul. Combust. 62 (1999) 183–200.
[46] M. Thordal, J. Bennetsen, H. Koss, Review for practical application of CFD for the determination of wind load on high-rise buildings, J. Wind Eng. Ind. Aerodyn. 186 (2019) 155–168.
[47] T.V. Karman, Progress in the statistical theory of turbulence, in: Proceedings of the National Academy of Sciences, 1948, pp. 530–539.
[48] S. Pope, Turbulent Flows, Cambridge University Press, Cambridge; New York, 2000.
[49] D. Peterson, Predicting rms pressures from computed velocities and mean pressures, Comput. Wind Eng. 1 (1994) 431–437.
[50] D. Meddage, I. Ekanayake, A. Weerasuriya, C. Lewangamage, K. Tse, T. Miyanawala, C. Ramanayaka, Explainable machine learning (XML) to predict external wind pressure of a low-rise building in urban-like settings, J. Pf Wind Eng. Ind. Aerodyn. 226 (2022) 1–20.
[51] M. Kuhn, K. Johnson, Feature Engineering and Selection: A Practical Approach for Predictive Models, CRC Press, 2019.
[52] B.C. Ross, Mutual information between discrete and continuous data sets, PLoS One 9 (2) (2014) 1–5.
[53] R. Smith, A mutual information approach to calculating nonlinearity, Stat 4 (1) (2015) 291–303.
[54] M. Awad, R. Khanna, Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers, A Press, 2015.
[55] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (3) (2004) 199–222.
[56] T.G. Dietterich, Ensemble methods in machine learning, in: Multiple Classifier Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 1–15.
[57] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[58] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer, New York, NY, 2001.
[59] R. Genuer, Variance reduction in purely random forests, J. Nonparametr. Stat. 24 (3) (2012) 543–562.
[60] A. Natekin, A. Knoll, Gradient boosting machines, A tutorial, Front. Neurorobot. 7 (2013) 1662–5218.
[61] J.H. Friedman, Stochastic gradient boosting, Comput. Statist. Data Anal. 38 (4) (2002) 367–378.
[62] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[63] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media, Inc, 2019.
[64] S. Raschka, Python Machine Learning, Packt Publishing Ltd., 2015.
[65] T. Tieleman, G. Hinton, Stochastic gradient boosting, COURSERA: Neural Netw. Mach. Learn. 4 (2) (2012) 26–31.
[66] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv:10.48550/ARXIV.1412.6980.
[67] S. Deshpande, J. Lengiewicz, S. Bordas, Probabilistic deep learning for real-time large deformation simulations, Comput. Methods Appl. Mech. Engrg. 398 (2022) 115307.
[68] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (56) (2014) 1929–1958.
[69] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 24, Curran Associates, Inc., 2011, pp. 1–9.
[70] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.
[71] H.F.S. Lui, W.R. Wolf, Construction of reduced-order models for fluid flows using deep feedforward neural networks, J. Fluid Mech. 872 (2019) 963–994.
[72] T. Yu, H. Zhu, Hyper-parameter optimization: A review of algorithms and applications, 2020, arXiv:10.48550/ARXIV.2003.05689.
[73] D. Chicco, M.J. Warrens, G. Jurman, The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation, PeerJ. Comput. Sci. 7 (2021) e623.
[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[75] T.J. DiCiccio, B. Efron, Bootstrap confidence intervals, Stat. Sci. 11 (3) (1996) 189–228.
[76] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, Springer, New York, NY, 2013.
[77] K. Weiss, T.M. Khoshgoftaar, D. Wang, A survey of transfer learning, J. Big Data 3 (9) (2016) 1–40.
[78] M. Inubushi, S. Goto, Transfer learning for nonlinear dynamics and its application to fluid turbulence, Phys. Rev. E 102 (2020) 1–8.
[79] A. Subel, A. Chattopadhyay, Y. Guan, P. Hassanzadeh, Data-driven subgrid-scale modeling of forced Burgers turbulence using deep learning with generalization to higher Reynolds numbers via transfer learning, Phys. Fluids 33 (3) (2021) 031702.