Prediction of 2D steady flow velocity fields with a CNN surrogate trained on RANS simulations

Michala Jakubcová[1,2,*], *Hana* Chaloupecká[1], and *Štěpán* Nosek[1]

[1]Institute of Thermomechanics of the Czech Academy of Sciences, Prague, Czech Republic
[2]Department of Water Resources and Environmental Modeling, Faculty of Environmental Sciences, Czech University of Life Sciences Prague, Prague, Czech Republic

Abstract. We present a data-driven surrogate modeling framework based on convolutional neural networks (CNNs) for predicting steady-state two-dimensional velocity fields in incompressible flows. The model was trained on a dataset of 120 Reynolds-averaged Navier–Stokes (RANS) simulations of flow past a rectangular obstacle, with systematic variation in inlet velocity, turbulence intensity, surface roughness, and obstacle orientation. Time-averaged velocity fields were extracted at $z = 2$ m, and subsequently interpolated onto a regular structured grid of 339×374 points. Only the horizontal velocity component U_x was retained for training the CNN. The surrogate model achieved a median MSE of 0.07 $(\mathrm{m/s})^2$ and R^2 of 0.75 on the test set, with most prediction errors localized in wake regions behind the obstacle. Cross-sectional velocity profiles and full-field error analyses confirmed high predictive accuracy across diverse flow configurations. Once trained, the CNN produces velocity field predictions within milliseconds, providing speed-ups of several orders of magnitude compared to RANS simulations and enabling rapid parametric exploration, design pre-screening, and real-time decision support.

1 Introduction

The accurate prediction of fluid flow velocity fields is central to a wide range of engineering and environmental applications, including aerodynamics, urban ventilation, groundwater transport, and atmospheric boundary layer (ABL) dynamics. High-fidelity computational fluid dynamics (CFD) simulations such as direct numerical simulation (DNS) or large eddy simulation (LES) provide detailed insights into these flows, but their high computational cost severely limits their use for design optimization, uncertainty quantification, and real-time decision-making [1]. Even Reynolds-averaged Navier–Stokes (RANS) models, although significantly cheaper than LES or DNS, remain too costly when thousands of scenarios must be evaluated within limited timeframes [2].

Recent advances in machine learning (ML) and deep learning (DL) offer a promising alternative by enabling the construction of surrogate models, i.e., data-driven emulators trained on CFD or experimental datasets, that can approximate flow solutions with orders-of-magnitude lower computational cost [3, 4]. Once trained, such models are capable of predicting velocity, pressure or concentration fields in milliseconds, compared to the hours required by conventional solvers, thereby facilitating rapid exploration of design spaces, near real-time forecasting, and integration into optimization and control frameworks. The growing availability of simulation data and the success of DL architectures in handling high-dimensional spatial information further accelerate the adoption of ML surrogates in fluid dynamics [3].

Among the various approaches, convolutional neural networks (CNNs) have emerged as a particularly effective tool. Their ability to extract local and global spatial features from structured data makes them well suited for predicting flow fields, which can be represented on regular grids analogous to images. CNN-based surrogates have been successfully applied to canonical problems such as flow around airfoils [5, 6] and cylinders [7], indoor airflow [8, 9], and environmental wind prediction [10], consistently achieving high accuracy and significant speed-up compared to CFD solvers. Variants of encoder–decoder architectures (e.g., U-Nets) have proven capable of reproducing complex flow patterns, including shock positions and recirculation zones, even in high-Reynolds-number regimes [2].

Beyond CNNs, other ML and DL architectures have been investigated to address challenges such as irregular geometries, unsteady flows, and long-range dependencies. Graph neural networks (GNNs) operate directly on unstructured meshes, offering greater geometric flexibility [2]. Recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) models, capture temporal dependencies in unsteady flow evolution, while autoencoders provide efficient dimensionality reduction for reduced-order modeling [4, 11]. More recently, transformer-based models and neural operators, such as the transformer-based neural operator (TNO), have

* Corresponding author: michala.jakubcova@it.cas.cz

demonstrated strong potential in capturing global interactions in turbulent and multiphase flows [12].

Environmental applications have become a particularly active domain for ML-based modeling, reflecting the urgency of problems requiring rapid flow and environmental process prediction. Examples include urban wind comfort analysis [4, 13], pollutant dispersion modeling [14], or hydro-meteorological forecasting using data-driven approaches [15]. In these contexts, ML models, including surrogate models trained on RANS, LES, or DNS data, have enabled real-time scenario evaluation and decision support that would otherwise be infeasible with conventional CFD alone.

This study situates itself within this growing body of work by focusing on the prediction of 2D steady flow velocity fields using CNN-based surrogates trained on RANS simulations. We aim to develop and validate a CNN surrogate model capable of predicting spatially resolved velocity fields around a rectangular obstacle in a 3D atmospheric boundary layer flow, using only a limited set of input parameters such as inlet velocity, turbulence intensity, surface roughness, and obstacle orientation. The goal is to enable rapid estimation of flow fields in urban-like configurations while preserving spatial accuracy and capturing key flow structures downstream of obstacles. Such surrogates can significantly accelerate parametric studies, support real-time decision-making in environmental modeling, and serve as efficient alternatives to full CFD in applications such as pollutant dispersion, urban microclimate assessment, or data-driven flow control.

2 Methodology

This study presents a data-driven surrogate modeling approach for predicting horizontal velocity fields (U_x) around a rectangular obstacle in a three-dimensional ABL flow. The methodology integrates CFD simulations, systematic data processing, and the training of CNN in an encoder-like architecture. The overall workflow consists of four stages: (1) generation of CFD data using OpenFOAM, (2) extraction and standardization of 2D velocity fields using Python programming language, (3) surrogate model training using DL, and (4) model evaluation and validation.

The overall pipeline from CFD simulation, post-processing, and data preparation to surrogate model training and evaluation is summarized schematically in Figure 1.

2.1 CFD data generation

All CFD simulations were conducted in OpenFOAM v12, within a three-dimensional domain that includes a centrally placed rectangular obstacle. A total of 120 unique scenarios were simulated based on a full factorial combination of input parameters:

- five uniform inlet velocities (U_{in}) → 0.5, 1.0, 2.0, 4.0, or 8.0 m/s,
- three surface roughness (k_s) → 0.001, 0.05, or 0.1 m,

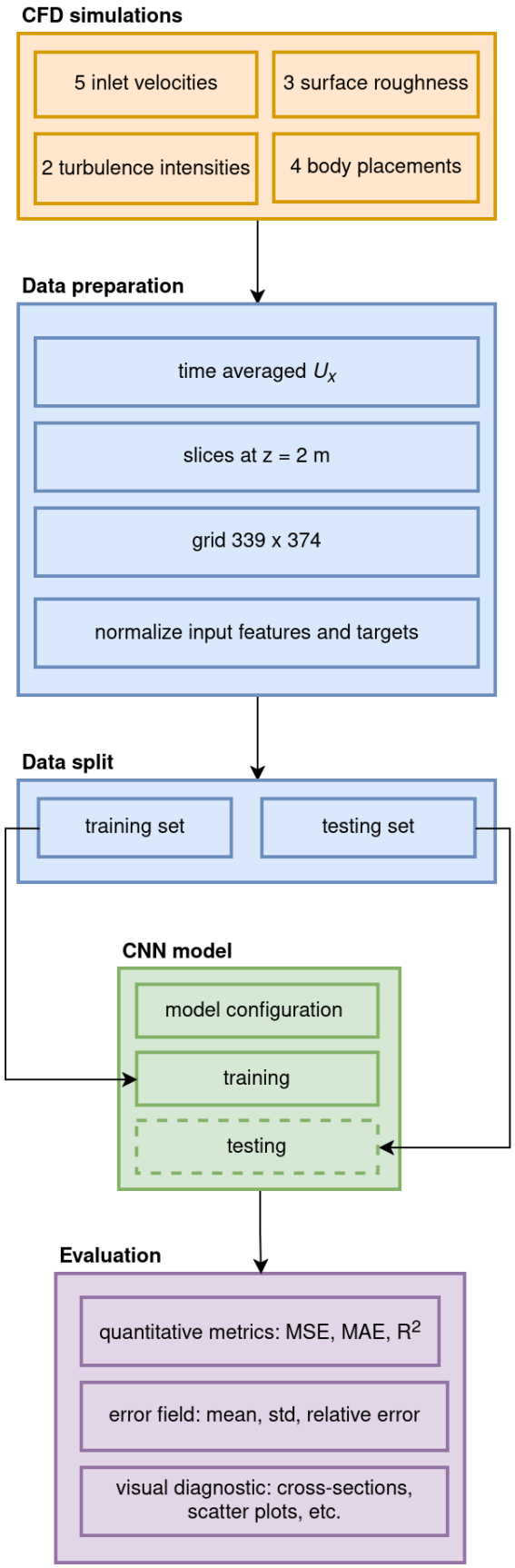

Figure 1. Overview of the surrogate modeling workflow. The pipeline consists of CFD simulations (orange), data preparation (blue), CNN model training and testing (green), and model evaluation (purple).

- two turbulence intensities (TI) → 5% or 15%,
- four obstacle orientations (θ_{rot}) → 0°, 15°, 30°, or 45°.

The selected range of inlet velocities represents typical near-surface wind speeds under moderate to strong ventilation conditions. This range ensures coverage of both low- and high-Reynolds-number regimes relevant for urban and environmental flows, while maintaining numerical stability and steady-state convergence in RANS simulations. Three surface roughness values were prescribed to represent smooth, moderately rough, and highly rough wall conditions, thus allowing evaluation of the surrogate model's sensitivity to near-wall boundary effects. The two turbulence intensities correspond to low- and high-turbulence inflow conditions, characteristic of open terrain and built-up or rough-surface environments, respectively. Finally, four obstacle orientations were included to capture the influence of body rotation on wake development and to introduce geometric variability in the training data without altering the overall domain configuration. These discrete parameter levels provide sufficient diversity for model generalization while keeping the total number of CFD simulations computationally tractable.

All simulations were carried out with the support of the MetaCentrum computing infrastructure, provided under the CESNET project.

2.1.1 Computational domain and mesh

The domain was a three-dimensional channel of dimensions $L_x \times L_y \times L_z = 200 \times 100 \times 50$m. A rectangular obstacle was positioned on the ground plane, centered laterally in the y-direction and offset in the streamwise direction, with a $5H$ buffer upstream and a $20H$ buffer downstream to ensure full flow development and minimize boundary effects. The obstacle had a fixed size of $10\,\text{m} \times 20\,\text{m}$ in the x, and y directions, respectively, but was variably rotated according to the specified setup (Fig. 2). The height of the obstacle was $H = 10$m.

The mesh was generated using the `snappyHexMesh` utility. A base hexahedral mesh was refined around the obstacle using three levels of local refinement to capture the steep gradients in velocity and turbulence quantities near solid surfaces. Boundary layers were resolved using a layered mesh with appropriate wall function compatibility. The total mesh count varied slightly across configurations but typically consisted of approximately one million cells. Mesh quality was assessed using the OpenFOAM built-in diagnostics.

2.1.2 RANS setup

Steady-state RANS simulations were performed using the `simpleFoam` solver in OpenFOAM v12. The standard two-equation k–ε turbulence model was employed to represent turbulent transport, solving separate transport equations for the turbulent kinetic energy k and its dissipation rate ε. The model assumes isotropic turbulence and applies wall functions to resolve near-wall regions efficiently.

The inlet boundary conditions were prescribed to impose the desired mean velocity magnitude and turbulence intensity, which were internally converted to k and ε fields. A uniform velocity profile was applied at the domain inlet, while no-slip boundary conditions were used on the obstacle and ground surfaces, and a symmetry plane was imposed at the domain top. Surface roughness effects were represented through a modified wall function.

Each simulation was iterated until the normalized residuals of all solved variables dropped below 10^{-6} and the velocity and pressure fields exhibited negligible changes between iterations, ensuring fully converged steady-state solutions.

2.2 Data processing

Once converged, each 3D CFD solution was post-processed in Python to extract a 2D horizontal slice at the height of $z = 2$ m. To obtain a uniform representation suitable for CNN input, the extracted fields were interpolated onto a regular structured grid covering the physical domain. The resulting slices contained 339×374 grid points spanning $x \in [-50, 150]$ m and $y \in [-50, 50]$ m. Only the horizontal velocity component U_x was retained for this study, while metadata such as TI, k_s, and θ_{rot} were embedded in the filenames for parsing.

The resulting slices were saved and loaded into Python using the `NumPy` library. Each scenario was represented by a three-channel input tensor containing the three scalar input parameters (U_{in}, TI, and k_s) spatially broadcasted over the domain, yielding the input array $X \in \mathbb{R}^{120\times339\times374\times3}$. The target output was the corresponding horizontal velocity field U_x, forming $y \in \mathbb{R}^{120\times339\times374}$.

All input and output fields were normalized to the range $[0, 1]$ to ensure stable and efficient CNN training. This normalization was applied globally using min–max scaling, where each value x in a given field was transformed into x^T according to

$$x^T = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{1}$$

where $x_{\min}$ and $x_{\max}$ denote the minimum and maximum values of the field across all scenarios, respectively. This global normalization ensured that the model learned relative spatial patterns consistently across different flow regimes and boundary conditions.

To evaluate generalization performance, the dataset of 120 scenarios was randomly split into two subsets, with 80% of the samples (i.e., 96 scenarios) used for training and the remaining 20% (i.e., 24 scenarios) reserved for independent testing. No overlap occurred between training and testing subsets, preventing information leakage or overfitting.

2.3 CNN surrogate model

To approximate the spatial distribution of the velocity component U_x from a small set of input parameters, CNN was implemented as a surrogate model in `Keras` using the

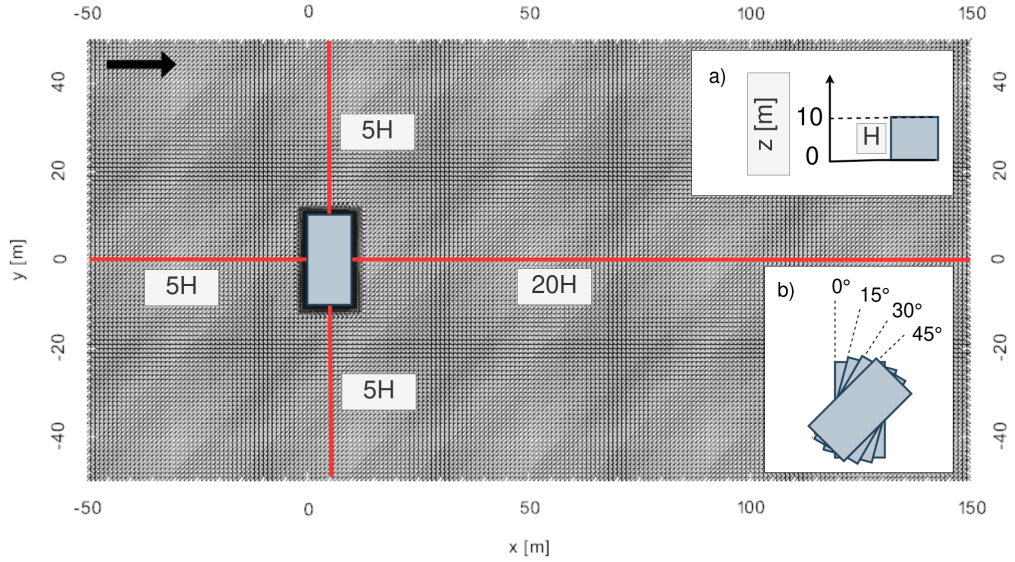

Figure 2. Schematic representation of the computational domain used in the RANS simulations, shown here as a horizontal 2D slice at $z = 2$ m for clarity, with a rectangular obstacle. Subfigure (a) shows a vertical side view of the obstacle. Subfigure (b) illustrates the four tested obstacle orientations relative to the incoming flow: 0°, 15°, 30°, and 45°. Buffer zones upstream (5H), downstream (20H), and laterally (5H) were used to minimize boundary effects.

`TensorFlow` backend. The adopted architecture follows an encoder-like structure, where convolutional and pooling layers progressively extract spatial features from the input representation.

Each input sample to the network is a three-channel 2D tensor of shape $(H_{\text{img}}, W_{\text{img}}, C_{\text{img}})$, where H_{img} and W_{img} denote the height and width of the structured slice, and C_{img} the number of input channels. In this study, $H_{\text{img}} = 339$, $W_{\text{img}} = 374$, and $C_{\text{img}} = 3$.

The CNN consisted of three convolutional layers with ReLU activations, defined as

$$\text{ReLU}(x) = \begin{cases} x, & x > 0, \\ 0, & x \leq 0. \end{cases} \tag{2}$$

Each convolutional layer used `same` padding and was followed by a 2×2 max-pooling operation to progressively reduce the spatial resolution while increasing the receptive field. A fully connected layer with 512 neurons and ReLU activation followed the convolutional stack, together with a dropout layer (rate = 0.3) to mitigate overfitting.

The final dense output layer contained $H_{\text{img}} \times W_{\text{img}}$ neurons and employed a linear activation

$$\text{Linear}(x) = x, \tag{3}$$

to regress the velocity field. The model was trained using the mean squared error (MSE, Eq. 4) as the loss function, with the mean absolute error (MAE, Eq. 5) used as an auxiliary metric.

The CNN was compiled using the Adam optimizer with a learning rate of 10^{-4} and trained for 100 epochs with a batch size of 4. During training, the output targets originally shaped as $(H_{\text{img}}, W_{\text{img}})$ were flattened into vectors for training.

The final model was saved in the `.keras` format, and the training loss history was recorded and visualized. The full pipeline is fully reproducible and designed for extensibility toward automated hyperparameter optimization or architecture search. In this study, hyperparameters such as learning rate, number of filters, and dropout rate were selected empirically based on preliminary experiments.

Although the adopted CNN architecture is deliberately simpler than more advanced architectures such as U-Nets or neural operators, this choice was made to prioritize computational efficiency and fast convergence. The encoder-like structure provided sufficient representational capacity for the studied parameter space while maintaining low computational cost. More complex architectures could potentially improve performance but at the expense of significantly higher training time and implementation complexity.

The corresponding `Keras` implementation of the model is shown below:

```
model = Sequential([
    InputLayer(input_shape=(height, width,
        channels)),
    Conv2D(16, (3, 3), activation='relu',
        padding='same'),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu',
        padding='same'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu',
        padding='same'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.3),
```

```
    Dense(output_size, activation='linear')
])

model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='mse',
    metrics=['mae']
)

history = model.fit(
    X_train, y_train_flat,
    validation_data=(X_test, y_test_flat),
    epochs=100,
    batch_size=4,
    verbose=1
)
```

Here, `height`, `width`, and `channels` are dynamically inferred from the data. The `output_size` corresponds to the total number of spatial points (i.e., $339 \times 374 = 126{,}786$) in the flattened output field.

The trained model serves as a differentiable surrogate of the CFD simulation, capable of predicting the spatial velocity field U_x for unseen combinations of input parameters within the training domain. Performance evaluation and visualization of the surrogate predictions are presented in Section 3.

2.4 Model Evaluation

The predicted velocity fields were quantitatively compared against the ground truth RANS solutions using three commonly adopted regression metrics: MSE, MAE, and the coefficient of determination (R^2). These metrics were computed for each test case to assess the accuracy and robustness of the CNN surrogate model as:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2\,, \tag{4}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|\,, \tag{5}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}\,, \tag{6}$$

where y_i and $\hat{y}_i$ denote the true and predicted velocity values at point i, respectively, $\bar{y}$ is the mean of the true values, and N is the total number of points.

In addition to global metrics, spatial patterns of prediction errors were analyzed using error field visualizations. Specifically, the mean error (ME) and standard deviation of the error (STD) across all test cases were computed as:

$$\mathrm{ME}(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M}\left(\hat{y}_j(\mathbf{x}) - y_j(\mathbf{x})\right)\,, \tag{7}$$

$$\mathrm{STD}(\mathbf{x}) = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\left(\hat{y}_j(\mathbf{x}) - y_j(\mathbf{x}) - \mathrm{ME}(\mathbf{x})\right)^2}\,, \tag{8}$$

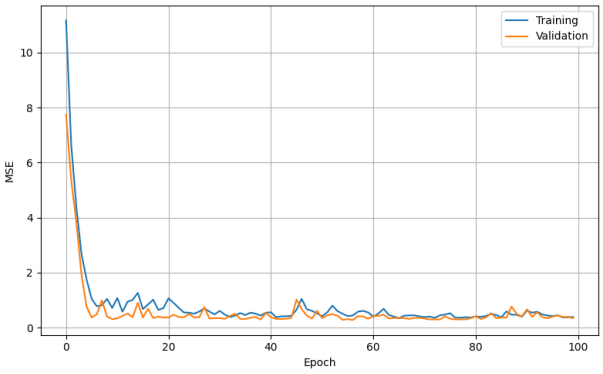

Figure 3. Training and validation loss (MSE) over epochs.

where $\mathbf{x}$ denotes the spatial location and M is the number of test cases. These fields highlight regions with systematic under- or over-prediction and spatial variability in model performance.

The point-wise prediction error (PE) used in the evaluation was defined as

$$\mathrm{PE_i} = \hat{y}_i - y_i\,, \tag{9}$$

where PE_i represents the local difference between predicted and true velocity values at point i. Positive values indicate overestimation, while negative values correspond to underestimation by the CNN.

To further assess the quality of the model, horizontal and vertical cross-sections were extracted from representative test cases and compared against the corresponding CFD results. A Gaussian filter was applied to the predicted fields to remove high-frequency noise where appropriate. A scatter plot of all predicted vs. true pixel values was used to assess regression accuracy and bias.

All evaluation and visualization procedures were implemented in Python using the `matplotlib`, `scikit-learn`, and `SciPy` libraries.

3 Results

The training process of the CNN was monitored using the MSE loss for both the training and validation datasets (Fig. 3). A rapid decrease in MSE is observed during the first 10 epochs, followed by a gradual convergence for both curves. The smooth and parallel evolution of the training and validation losses indicates stable training without signs of overfitting.

A qualitative comparison between the CNN-predicted velocity field and the RANS solution for a representative test case (i.e., $U_{in} = 4.0$ m/s, $k_s = 0.05$ m, TI = 15%, and $\theta_{\mathrm{rot}} = 30°$) is presented in Fig. 4. The predicted velocity field (middle panel) closely captures key flow features from the RANS ground truth (left panel), including the wake region and recirculation zones behind the obstacle. The prediction error (right panel) remains predominantly within the range of ±1 m/s, and is localized primarily in regions characterized by complex flow structures, such as vortex shedding or sharp velocity gradients.

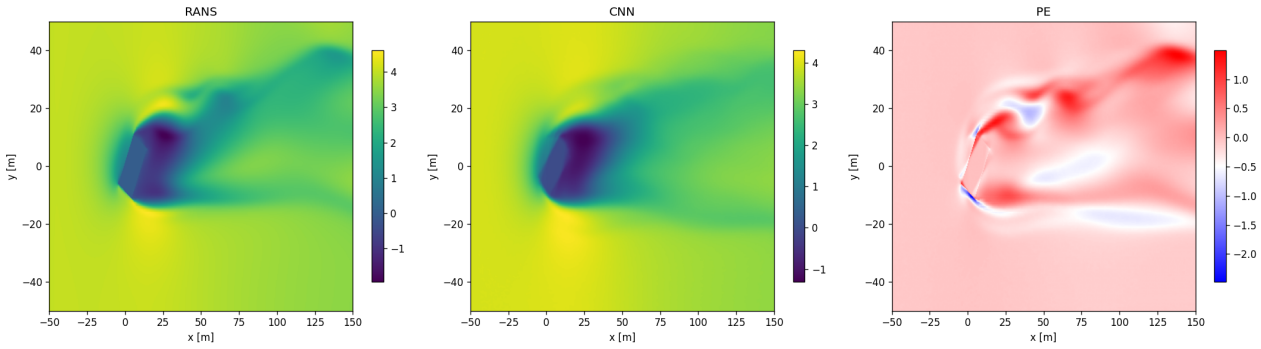

Figure 4. Comparison of RANS ground truth velocity field U_x[m/s] (left), CNN predicted velocity field U_x[m/s] (middle), and PE metric (right) for a representative test case.

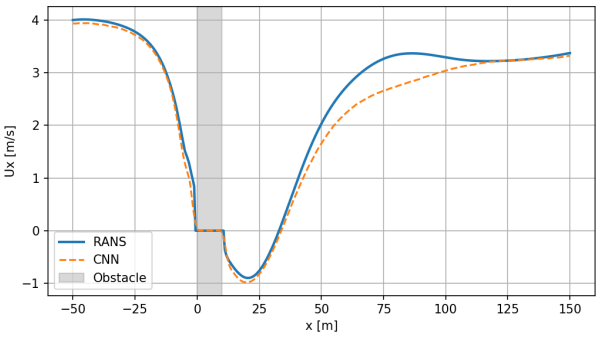

Figure 5. Horizontal velocity profile, U_x, at $y = 0$ m for a representative test case. The shaded region denotes the position of the obstacle.

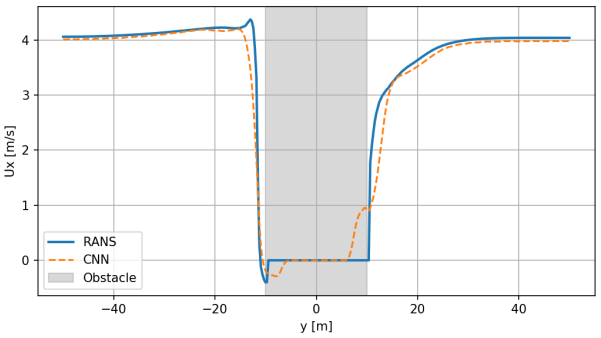

Figure 6. Vertical velocity profile, U_x, at $x = 5.23$ m for a representative test case. The shaded region denotes the position of the obstacle.

To further assess the spatial fidelity of the predictions, horizontal and vertical velocity profiles were extracted along the symmetry axes passing through the obstacle center (Figs. 5 and 6). Both sections demonstrate strong agreement between the CNN and RANS velocity profiles, including in the wake region behind the obstacle. Small deviations are observed near steep velocity gradients at the obstacle edges, which are expected due to local sharp transitions.

The global spatial distribution of model performance was analyzed using statistical error fields aggregated over all test cases (Fig. 7). The ME field remains close to zero across the majority of the domain, suggesting the absence of consistent bias. The STD and MSE fields show localized peaks in the wake region downstream of the obstacle, where flow features are most complex and less represented in the training data. Nonetheless, these elevated errors are confined to small regions, indicating that the model generalizes well across the domain.

Table 1 summarizes the statistical performance of the model across both training and test datasets. On the training data, the model achieves a mean MSE of 0.22, a mean MAE of 0.26, and a mean R^2 of 0.74. The corresponding medians (MSE = 0.06, MAE = 0.22, R^2 = 0.80) indicate that the majority of training cases were predicted with higher accuracy, while the larger standard deviations suggest the presence of a few outliers.

On the test set, the model maintains strong performance with a mean MSE of 0.34, mean MAE of 0.31, and mean R^2 of 0.67. Again, the median values reflect improved performance (MSE = 0.07, MAE = 0.23, R^2 = 0.75), further indicating that the model generalizes well across unseen configurations. The slight decrease in test accuracy is expected due to the increased variability and unseen placements of obstacles, but the results remain competitive for surrogate modeling applications.

To evaluate overall accuracy, a scatter plot of all predicted vs. true pixel values across the test set is shown in Fig. 8. The predictions align closely with the ideal $y = x$ reference line, yielding a high R^2 value of 0.95, which demonstrates the surrogate model's ability to reliably capture the range and magnitude of the RANS-computed velocities.

Additionally, the histogram of prediction errors (Fig. 9) reveals a tight, symmetric distribution centered near zero, with most errors within ±1 m/s. This further confirms that the model does not systematically over- or under-predict the velocity field, and that large errors are rare.

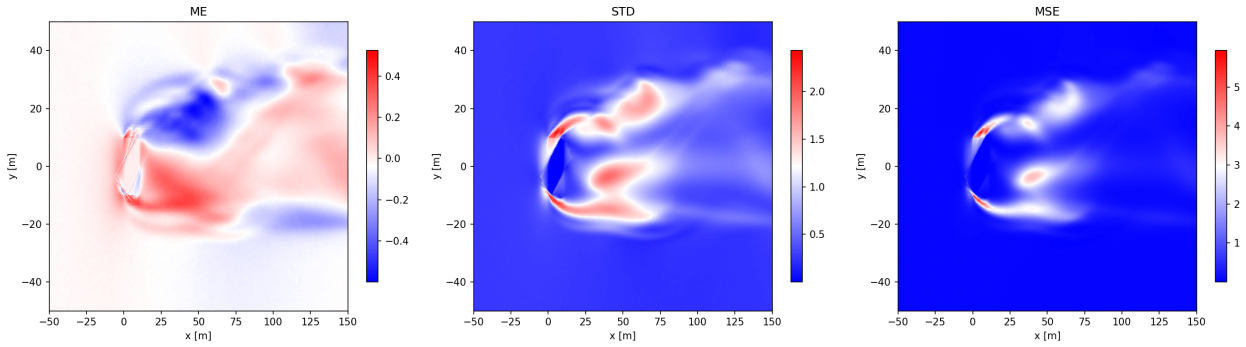

Figure 7. Spatial distribution of ME (left), STD (middle), and MSE (right) across all test cases

Table 1. Summary of CNN model performance on the training and testing datasets. Reported values include the mean, median, and standard deviation (stdev) across all cases for MSE, MAE, and R^2.

Statistic	Train			Test		
	MSE	MAE	R^2	MSE	MAE	R^2
Mean	0.2160	0.2582	0.7402	0.3428	0.3076	0.6728
Median	0.0625	0.2169	0.7995	0.0675	0.2301	0.7483
Stdev	0.3315	0.2038	0.2066	0.6564	0.2715	0.2424

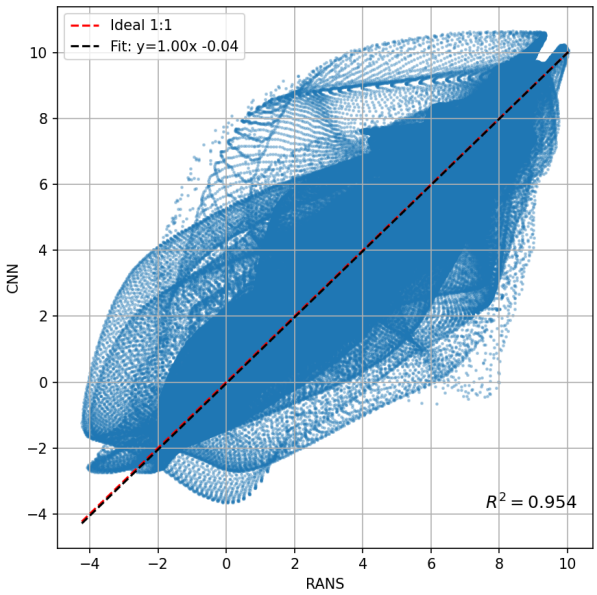

Figure 8. Scatter plot comparing RANS ground truth and CNN predicted values of U_x across all test pixels. The red line indicates the ideal 1:1 match.

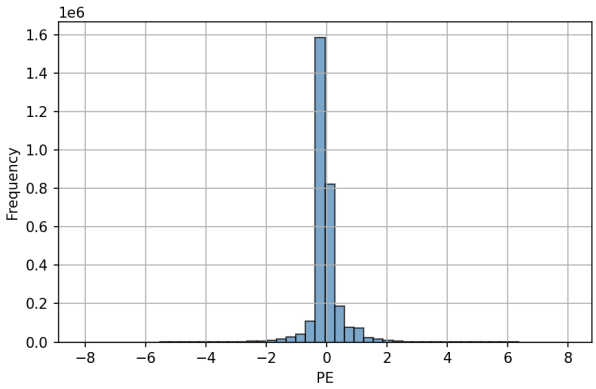

Figure 9. Histogram of PE across all test pixels.

4 Conclusions

This study demonstrated the feasibility and effectiveness of using a convolutional neural network (CNN) as a surrogate model for predicting two-dimensional steady-state velocity fields around a rectangular obstacle. The CNN was trained on a dataset of RANS simulations covering a range of inflow velocities, roughness classes, turbulence intensities, and obstacle orientations. By learning the mapping from flow parameters to the corresponding velocity fields, the surrogate model achieved high prediction accuracy while significantly reducing computational cost compared to traditional CFD simulations.

The performance of the surrogate model was validated using both statistical metrics and spatial error analysis. Across all test cases, the CNN achieved median R^2 values above 0.74, with error distributions tightly centered around zero. Visual comparisons of predicted velocity fields with RANS solutions confirmed that the model reliably captured key flow features, including separation and wake regions. Cross-sectional velocity profiles and aggregated error maps further highlighted the model's capacity to generalize across varied geometries, with most discrepancies localized to complex near-wake regions.

The present surrogate model is limited to predicting two-dimensional steady-state velocity fields at a fixed height above the surface. While this allows for rapid

proof-of-concept development, future work will focus on extending the approach to three-dimensional domains, transient flow conditions, and more complex geometries such as non-rectangular buildings or building clusters. Such extensions would broaden the model's applicability to more realistic environmental and engineering scenarios, while also providing opportunities to explore hybrid or physics-informed architectures.

Acknowledgment

The computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic. The authors would like to acknowledge the support of the Technology Agency of the Czech Republic (TAČR), grant SQ01010181. We further appreciate the institutional sup-port from the Institute of Thermomechanics of the Czech Academy of Sciences for its long-term conceptual devel-opment, RVO: 61388998.

References

[1] R. Garcia-Fernandez, K. Portal-Porras, O. Irigaray, Z. Ansa, U. Fernandez-Gamiz, Cnn-based flow field prediction for bus aerodynamics analysis, Scientific Reports **13**, 21213 (2023). https://doi.org/10.1038/s41598-023-48419-4

[2] S.J. Jacob, M. Mrosek, C. Othmer, H. Köstler, Benchmarking convolutional neural network and graph neural network based surrogate models on a real-world car external aerodynamics dataset, Computers Fluids **300**, 106760 (2025). https://doi.org/10.1016/j.compfluid.2025.106760

[3] T. Chen, R. Li, X. Hu, B. Zhang, Y. Liu, L.L. Wang, N. Gao, Machine learning as cfd surrogate models for rapid prediction of building-related physical fields: A review of methods and state-of-the-art, Building and Environment p. 113667 (2025). https://doi.org/10.1016/j.buildenv.2025.113667

[4] R. Mao, Y. Lan, L. Liang, T. Yu, M. Mu, W. Leng, Z. Long, Rapid cfd prediction based on machine learning surrogate model in built environment: A review, Fluids **10**, 193 (2025). https://doi.org/10.3390/fluids10080193

[5] M.Y. Wu, Y. Wu, X.Y. Yuan, Z.H. Chen, W.T. Wu, N. Aubry, Fast prediction of flow field around airfoils based on deep convolutional neural network, Applied Sciences **12**, 12075 (2022). https://doi.org/10.3390/app122312075

[6] J. Hu, W. Zhang, Flow field modeling of airfoil based on convolutional neural networks from transform domain perspective, Aerospace science and technology **136**, 108198 (2023). https://doi.org/10.1016/j.ast.2023.108198

[7] H. Ozaki, T. Aoyagi, Prediction of steady flows passing fixed cylinders using deep learning, Scientific Reports **12**, 447 (2022). https://doi.org/10.1038/s41598-021-03651-8

[8] H. Gao, W. Qian, J. Dong, J. Liu, Rapid prediction of indoor airflow field using operator neural network with small dataset, Building and Environment **251**, 111175 (2024). https://doi.org/10.1016/j.buildenv.2024.111175

[9] C. Wei, R. Ooka, Applying a physics-informed neural network to an indoor airflow time-extrapolation prediction, Building and Environment p. 113246 (2025). https://doi.org/10.1016/j.buildenv.2025.113246

[10] T. Bashir, H. Wang, M. Tahir, Y. Zhang, Wind and solar power forecasting based on hybrid cnn-abilstm, cnn-transformer-mlp models, Renewable Energy **239**, 122055 (2025). https://doi.org/10.1016/j.renene.2024.122055

[11] X. Li, W. Xu, M. Ren, Y. Jiang, G. Fu, Hybrid cnn-lstm models for river flow prediction, Water Supply **22**, 4902 (2022). https://doi.org/10.2166/ws.2022.170

[12] Z. Li, T. Liu, W. Peng, Z. Yuan, J. Wang, A transformer-based neural operator for large-eddy simulation of turbulence, Physics of Fluids **36** (2024). https://doi.org/10.1063/5.0210493

[13] Z. Liu, S. Zhang, X. Shao, Z. Wu, Accurate and efficient urban wind prediction at city-scale with memory-scalable graph neural network, Sustainable Cities and Society **99**, 104935 (2023). https://doi.org/10.1016/j.scs.2023.104935

[14] G. Chen, S. Chen, D. Li, C. Chen, A hybrid deep learning air pollution prediction approach based on neighborhood selection and spatio-temporal attention, Scientific Reports **15**, 3685 (2025). https://doi.org/10.1038/s41598-025-88086-1

[15] M. Jakubcová, P. Máca, M. Hanel, P. Pech, Combination of hybrid artificial neural networks with particle swarm optimization algorithm for spei forecasting, Journal of Hydrometeorology (2025). https://doi.org/10.1175/JHM-D-25-0034.1